**网络工程实验系列之**

**网络协议教学实验教程**

Teaching ExperimentalSystemfor Network Protocol

**（TCP/IP协议——IPv4网络协议篇）**

吉林中软吉大信息技术有限公司

**目录**

[第1部分 IPv4网络协议](#_Toc327522618)

[实验1 IEEE802标准和以太网 1](#_Toc327522619)

[实验2 地址解析协议（ARP） 18](#_Toc327522620)

[实验3 网际协议（IP） 36](#_Toc327522621)

[实验4 Internet组控报文协议（ICMP） 66](#_Toc327522622)

[实验5 Internet组管理协议（IGMP） 76](#_Toc327522623)

[实验6 用户数据报协议（UDP） 88](#_Toc327522624)

[实验7 传输控制协议（TCP） 112](#_Toc327522625)

[实验8 ICP协议的高级特性 123](#_Toc327522626)

[实验9 简单网络管理协议(SNMP) 136](#_Toc327522627)

[实验10 动态主机配置协议(DHCP) 158](#_Toc327522628)

[实验11 域名服务(DNS) 166](#_Toc327522629)

[实验12 网络地址转换(NAT) 181](#_Toc327522630)

[实验13 超文本传输协议(HTTP) 186](#_Toc327522631)

[实验14 远程登录与文件传送协议(TELNET与FTP) 193](#_Toc327522632)

[实验15 电子邮件协议(SMTP、POP3和IMAP) 212](#_Toc327522633)

[实验16 NetBIOS应用及SMB/CIFS协议 225](#_Toc327522634)

[实验17 路由信息协议(RIP) 232](#_Toc327522635)

[实验18 开放式最短路径优先协议(OSPF)Ⅰ 244](#_Toc327522636)

[实验19 开放式最短路径优先协议(OSPF)Ⅱ 257](#_Toc327522637)

[实验20 代理 267](#_Toc327522638)

[实验21 IPv4综合实验 273](#_Toc327522639)

[第2部分网络攻防与故障](#_Toc327522640)

[实验22 ARP地址欺骗 275](#_Toc327522641)

[实验23 ICMP重定向 281](#_Toc327522642)

[实验24 TCP与UDP端口扫描 288](#_Toc327522643)

[实验25 路由欺骗 292](#_Toc327522644)

[实验26 冲突与网络广播风暴 295](#_Toc327522645)

[实验27 路由环与网络回路 297](#_Toc327522646)

**附录**

[附录A 网络结构 299](#_Toc327522647)

[附录B Outlook Express的使用方法 305](#_Toc327522648)

[附录C Windows 2003下SNMP服务的安装 306](#_Toc327522649)

[附录D 协议编辑器使用说明 310](#_Toc327522650)

[附录E 协议分析器使用说明 319](#_Toc327522651)

[附录F 工具使用说明 323](#_Toc327522652)

[附录G 常见问题 328](#_Toc327522653)

第1部分IPv4网络协议

## 实验1IEEE802标准和以太网

【实验目的】

1.掌握以太网的报文格式

2.掌握MAC地址的作用

3.掌握MAC广播地址的作用

4.掌握LLC帧报文格式

5.掌握协议编辑器和协议分析器的使用方法

6.掌握协议栈发送和接收以太网数据帧的过程

【学时分配】

4学时

【实验环境】

该实验采用网络结构一

【实验原理】

#### 一.OSI模型和TCP/IP协议族

##### 1.OSI简介

国际标准化组织（ISO）成立于1947年，它是个多国团体，专门就一些国际标准达成世界范围的一致。网络方面的ISO标准就是OSI（开放系统互连）模型。它是在20世纪70年代后期问世的。

在不需要改变底层硬件或软件逻辑的情况下，OSI模型使两个不同的系统能够较容易地通信。OSI模型并不是协议，它是个灵活的、稳健的和可互操作的模型，用来设计网络体系结构，它使得所有类型的计算机系统可以通信。OSI模型包括7个层次，每一层都定义了通过网络传送信息的一些过程，如下图所示。掌握了OSI模型的基本概念后，就有了学习数据通信较牢固的基础。

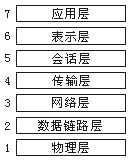


图1-1OSI模型

##### 2.OSI模型中的层次

(1)物理层

物理层协调在物理媒体中传送比特流所需的各种功能。物理层涉及到接口和传输媒体的机械的和电气的规约。它还定义了这些物理设备和接口在传输过程中所必须完成的任务。

(2)数据链路层

数据链路层把物理层（即原始的传输设施）转换为可靠的链路。

(3)网络层

网络层负责把数据包从源点交付到终点，这可能要跨越多个网络。数据链路层是监督在同一个链路上的两个相邻节点之间数据包的交付，而网络层则确保每一个数据包能够从它的源点到达终点。

如果两个节点连接到同一条链路上，那么通常就不需要网络层。但是，如果两个节点连接在不同的网络上，而这些网络是由一些连接的设备连接起来的，那么通常是需要网络层来完成从源点到终点的交付。

(4)传输层

传输层负责把报文进行端到端的交付。网络层虽然负责单个数据包的端到端交付，但它并不考虑这些数据包之间的关系。传输层要确保整个报文原封不动地按序到达，负责从源点到终点这一级的差错控制和流量控制。

(5)会话层

会话层是网络的对话控制器。它建立并维持通信系统之间的交换，并使这些通信系统同步。会话层完成以下任务：

●对话控制：会话层允许两个节点进行对话状态控制。它允许两个进程之间的通信按半双工或全双工的方式进行。

●同步：会话层允许进程将若干个同步点插入到数据流中，以完成传输的同步。

(6)表示层

表示层负责两个系统所交换的信息的语法和语义。表示层完成以下任务：

●转换：在两个系统中的进程所交换的信息的形式通常都是字符串，数字等等。这些信息在传输之前必须变为比特流。由于不同的计算机使用不同的编码系统，所以表示层负责在这些不同的编码方法之间提供互操作性。

●加密：为了携带敏感信息，一个系统必须确保能够进行保密。加密就是发送端把原始信息转换为另一种形式，然后再把这种形式的报文发送出去。

●压缩：数据压缩减少了信息中所包含的比特数。在传输多媒体信息时，数据压缩是特别重要的。

(7)应用层

应用层使用户（不管是人还是软件）接入网络，给用户提供了接口，也提供了对多种服务的支持。

##### 3.TCP/IP协议族

TCP/IP协议族是在OSI模型出现之前出现的。因此TCP/IP协议族的层次无法准确地和OSI模型对应起来。TCP/IP协议族由5层组成：物理层、数据链路层、网络层、传输层和应用层。前四层与OSI的前四层相对应，提供物理标准、网络接口、网际互连、以及传输功能。然而OSI的高三层在TCP/IP中则叫做应用层。如下图所示：

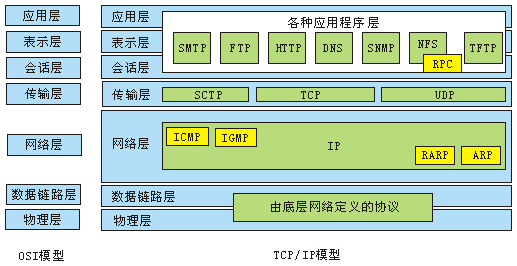


图1-2TCP/IP和OSI模型

TCP/IP是由一些交互的模块组成的分层次的协议，每个模块提供特定的功能。TCP/IP协议族中的各层包含了一些相对独立的协议，可以根据系统的需要把这些协议混合和配套使用。

在传输层中，TCP/IP定义了3个协议：传输控制协议（TCP）、用户数据报协议（UDP）和流控制协议（SCTP）。在网络层中，TCP/IP定义的主要协议是网际协议（IP）。

**二.IEEE802参考模型**

1980年2月IEEE（电气和电子工程师协会）成立了802局域网标准委员会，开始了有关局域网标准化的工作。IEEE802局域网参考模型中只定义了物理层和数据链路层，在模型中较高层次，IEEE802参照OSI模型，尽可能与其相符合。IEEE标准模型与OSI模型的比较如下图所示：

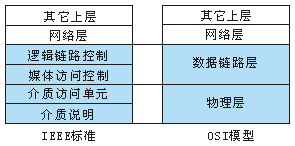


图1-3IEEE标准模型与OSI模型比较

在OSI模型中，物理层的任务是为上一层提供一个物理连接，以透明的方式传送比特流。而在IEEE802局域网参考模型中，物理层被分为上下两个子层，分别为：

●下面的子层是对电缆介质的说明；

●上面的子层是介质访问单元（MAU）。

电缆可以是各种介质，如双绞线，同轴电缆等。MAU的主要作用是信息编码、信号发送和介质处理等。

在OSI模型中，数据链路层的任务是把物理层转化成可靠的链路，使物理层对上层（网络层）看起来好像是不产生差错的。而IEEE802标准的数据链路层被分为两个子层：

●下面的子层是媒体访问控制子层（MAC）；

●上面的子层是逻辑链路控制子层（LLC）。

LLC子层的功能是实现有效的数据传输，负责数据链路层的流量控制和差错控制。MAC子层的功能是保证物理功能和逻辑功能的连续性，还把从LLC子层收到的数据组装成帧，并把帧交给物理层进行编码。

#### 三.以太网简介

IEEE802.3所支持的局域网标准最早是由Xerox开发的，后来通过Digital公司、Intel公司和Xerox公司联合扩展为以太网标准，符合以太网标准的局域网络称为以太网。

##### 1.以太网的分类

数据速率为10Mbps的以太网称为标准以太网，数据速率为100Mbps的以太网称为快速以太网，数据速率为1000Mbps的以太网称为千兆以太网。目前10G以太网的标准也已正式制定。

##### 2.以太网的物理地址

以太网上的每一个主机都有自己的网络接口卡（NIC）。网络接口卡通常安装在主机内部，并为主机提供一个6字节的物理地址，如：44-45-53-54-00-00。在遵循IEEE802标准的以太网络中，将这个物理地址称作“MAC地址”。MAC地址是惟一的，任意两个不同的网络接口卡都具有不同的MAC地址。MAC地址中的某些位具有特殊的意义，如下图所示：

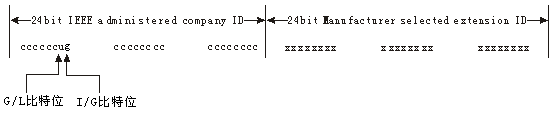


图1-4以太网的物理地址

I/G比特位表示Individual/Group，当I/G位为0时，地址字段表示单个站地址（单播），为1时表示组地址，用来进行多播。

G/L比特位表示Global/Local，当G/L位为1时是全球管理，保证在全球没有相同的地址，当为0时是本地管理，这时用户可任意分配网络上的地址。

以太网MAC地址可分为三类：单播地址、广播地址和多播地址。单播地址（unicast）是一对一的，该地址是特定主机的MAC地址；广播地址（broadcast）：广播地址是一对全体的，该地址为全1，指明数据帧是发送给所有主机的。多播地址（multicast）：多播地址是一对多的，指明数据帧是发送给一部分主机的。

#### 四.以太网访问模式

当多个节点被连接到一条链路上时，叫做多点链路或广播链路。这时就需要一个协议来协调链路的访问，使得同一时刻只有一个节点访问链路。如果发生同一时刻多个节点使用链路的情况，则称为链路发生了冲突。带有冲突检测的载波侦听多路访问（CSMA/CD）是这样一种方案。发送主机在传输过程中仍继续监听信道，以检测是否存在冲突。如果发生冲突，信道上可以检测到超过发送主机本身发送的载波信号的幅度，由此判断出冲突的存在。一旦检测到冲突，就立即停止发送，并向总线上发一串阻塞信号，用以通知总线上其它各有关主机。这样，通道容量就不致因白白传送已受损的帧而浪费，可以提高总线的利用率。以太网使用CSMA/CD作为其访问模式。

#### 五.以太网的帧格式

##### 1.以太网的MAC帧格式

以太网的MAC帧格式有两种标准，一种是DIXEthernetV2标准，另一种是IEEE的802.3标准。但两种帧格式可以在同一以太网络共存。两种帧格式都具有7个域：前导码、帧首定界符、目的MAC地址、源MAC地址、协议类型或数据长度、数据、帧校验序列。如下图所示：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 前导码  （7字节） | 帧首定界符（1字节） | 目的MAC地址（6字节） | 源MAC地址（6字节） | 协议类型或数据长度  （2字节） | 数据 | 帧校验序列  （4字节） |

图1-5以太网的MAC帧格式

两种格式的帧可以依据协议类型或数据长度字段的值进行区分。如果此帧是DIXEthernetV2标准格式帧，则协议类型或数据长度字段的值大于1536；如果此帧是IEEE802.3标准格式的帧，则协议类型或数据长度字段的值小于1518。对DIXEthernetV2帧来说，此字段的值代表了高层协议的类型；对IEEE802.3帧来说，它的高层协议一定是LLC，此字段的值代表了数据的长度。

在以太网的MAC帧格式中，各个字段的含义如下：

●前导码：这是以太网MAC帧的第一个域，包含了7个字节的二进制“1”和“0”间隔的代码，即“10101010……10”共56位，提示接收方一个数据帧即将到来，同时使接收系统建立起同步时钟。

●帧首定界符：帧首定界符标记了帧的开始。它是一个字节的“10101011”二进制序列，帧首定界符通知接收方后面所有的内容都是数据，以便接收方对数据帧进行定位。

●目的MAC地址：目的MAC地址为6个字节，标记了数据帧下一个主机的物理地址。如果数据包的目的地址必须从一个网络穿越到另一个网络，那么目的MAC地址所包含的是连接当前网络和下一个网络的路由器地址。当数据包到达目标网络后，目的MAC地址域换成目的主机的地址。

●源MAC地址：源MAC地址也是6个字节。它包含了最后一个转发此帧的设备的物理地址。该设备可以是发送此数据帧的主机，也可以是最近接收和转发此数据帧的路由器。

●协议类型或数据长度：如果该字段的值小于1518，它用于定义后面数据字段的长度；如果字段的值大于1536，它定义一个封装在帧中的数据包的类型。

●数据：它的长度范围是从46到1500字节之间。46是以太网MAC帧所封装的高层协议数据的最小长度。如果高层协议的数据包小于46字节，则填充到46字节。

●帧校验序列：最后一个域是帧校验序列，以太网采用32位冗余校验（CRC）。校验范围是除了前导码、帧首定界符和帧校验序列外的所有内容。

##### 2.LLC帧格式

LLC的帧格式如下图所示：

|  |  |  |  |
| --- | --- | --- | --- |
| DSAP | SSAP | 控制 | LLC数据 |

图1-6LLC帧格式

其中，DSAP（目的服务访问点）和SSAP（源服务访问点）是LLC所使用的地址，用来标识接收和发送数据的计算机上的用户实体。DSAP的第一个比特是用来指明帧是为单地址还是组地址，0表示单地址，1表示组地址。SSAP的第一个比特用来指明帧是命令帧还是响应帧。0表示命令帧，1表示响应帧。

LLC定义了三种帧：信息帧（I-帧）、监控帧（S-帧）和无编号帧（U-帧）。帧的类型可从控制字段识别。对于信息帧和监控帧，控制字段为2字节长，而对于无编号帧，控制字段为1字节长。

下图表示了LLC三类帧的控制字段的比较。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1位 | | 7位 | | | | | | 1位 | | | 7位 | | | | |
| 0 | | N(S) | | | | | | P/F | | | N(R) | | | | |
| I-帧的控制字段 | | | | | | | | | | | | | | | |
| 1位 | 1位 | | 2位 | | 4位 | | | | 1位 | | | 7位 | | | |
| 1 | 0 | | SS | | X | | | | P/F | | | N(R) | | | |
| S-帧的控制字段 | | | | | | | | | | | | | | | |
| 1 | | | | 1 | | M | M | | | P/F | | | M | M | M |
| U-帧的控制字段 | | | | | | | | | | | | | | | |

图1-7LLC三类帧的控制字段

●N(S)：发送序号。

●N(R)：接收序号。

●SS：监控功能位，00表示准备接收（RR）；10表示未准备接收（RNR）；01表示拒绝（REJ）。

●M：修正功能位。

●X：保留，设置为0。

●P/F：Poll/final位。命令LLCPDU传输/响应LLCPDU传输。

**3.LLC-PDU与相邻层的PDU之间的关系**

IEEE802标准为LLC和MAC子层的帧格式作了详细规定。下图描述了网络层PDU、LLC子层PDU和MAC子层PDU的关系。

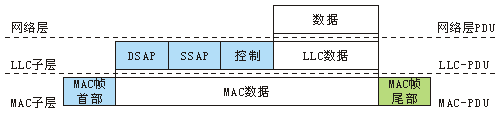


图1-8LLC-PDU与相邻层的PDU之间的关系

LLC帧（即LLC-PDU）与媒体无关，而MAC（即MAC-PDU）则与局域网的媒体访问方式有很大关系，不同的局域网有不同的MAC帧格式。

**4.LLC地址与MAC地址**

在MAC帧的帧首中，有目的MAC地址和源MAC地址，它们都是6字节长。在LLC帧的帧首中，则设有DSAP和SSAP，该地址是逻辑地址，表示的是数据链路层的不同访问服务点。LLC地址与MAC地址是两个不同的概念，在局域网中，一个主机上的多个服务访问点可以利用同一条数据链路。从这一点可以看出，LLC子层带有OSI网络层的某些功能。

#### 六.协议栈实现代码解析

本实验将通过对netproto\_eth\_student.h和netproto\_eth\_student.c两个文件进行编码，完成协议栈中以太网数据帧接收和发送的实现。

netproto\_eth\_student.h文件中定义了以太网数据帧中“协议类型与数据长度”字段值以及以太网数据帧的负载内容、负载长度，关键代码如下所示：

#definePAYLOAD\_DATA"Hello,World!"

#definePAYLOAD\_LENsizeof(PAYLOAD\_DATA)

#defineMAX\_PAYLOAD\_LEN1024

#defineTYPE\_LENGTH5893

这段代码定义了3个宏，他们代表的含义如下表所示：

表1-1netproto\_eth\_student.h中定义的宏

|  |  |  |
| --- | --- | --- |
| 宏 | 值 | 描述 |
| PAYLOAD\_DATA | "Hello,World!" | 定义以太网负载数据 |
| PAYLOAD\_LEN | sizeof(PAYLOAD\_DATA) | 定义以太网负载数据长度 |
| TYPE\_LENGTH | 5893 | 定义以太网帧中的“协议类型或数据长度”字段的值 |

在实验的编码过程中，应该使用这些宏对相应的变量进行赋值。学生也可以根据自己的需求修改这些宏定义的值。

netproto\_eth\_student.c文件是协议栈中以太网数据帧的实现部分，其中定义了2个函数。下面分别介绍这些协议栈的实现部分。

函数netp\_eth\_output\_student的功能是编辑并发送一个EthernetV2数据帧。这个函数的编码工作需要由学生完成。

当有数据到达本机网络接口时，函数netp\_eth\_input\_student将被调用，并传递给这个函数原始数据。该函数的返回值为push\_to\_lwip的枚举类型值，push\_to\_lwip的定义如下：

enumpush\_to\_lwip{

NETP\_PUSH\_TO\_LWIP,//数据处理完成后，交给lwIP继续处理

NETP\_NO\_PUSH\_LIWP//数据处理完成后，不交给lwIP继续处理，本层处理完毕以后

数据包被丢弃

};

返回NETP\_PUSH\_TO\_LWIP表示这个数据帧应该提交给协议栈上层继续处理，而返回NETP\_NO\_PUSH\_LIWP则表示不需要提交给协议栈上层处理，本层处理完毕后，这个数据帧将被丢弃。需要根据正确的逻辑关系返回适当的值，使协议栈正常工作。

在编码过程中可能会遇到一些结构体、宏和函数，下表是对他们进行介绍：

表1-2实验涉及的结构体和函数

| 结构体/函数 | 声明或定义 | 描述 |
| --- | --- | --- |
| structnetp\_eth\_header | structnetp\_eth\_header{  structnetp\_eth\_addrdest\_address;  structnetp\_eth\_addrsour\_address;  u16\_ttype;  }; | 以太网数据帧头结构 |
| structnetp\_eth\_addr | structnetp\_eth\_addr{  u8\_taddr[ETH\_ADDRESS\_LEN];  }; | 以太网地址 |
| ETH\_HEADER\_LEN | #defineETH\_HEADER\_LEN14 | 以太网帧头长度 |
| PAYLOAD\_DATA | #definePAYLOAD\_DATA"Hello,World!" | 以太网负载内容 |
| PAYLOAD\_LEN | #definePAYLOAD\_LENsizeof(PAYLOAD\_DATA) | 以太网负载长度 |
| MAX\_PAYLOAD\_LEN | #defineMAX\_PAYLOAD\_LEN1024 | 最大负载长度 |
| TYPE\_LENGTH | #defineTYPE\_LENGTH5893 | 协议类型与数据长度字段值 |
| netp\_set\_eth\_addr | voidnetp\_set\_eth\_addr(  structnetp\_eth\_addr\*p\_addr,  u8\_tv1,u8\_tv2,u8\_tv3,  u8\_tv4,u8\_tv5,u8\_tv6  ); | 设置以太网数据帧MAC地址 |
| netp\_current\_hw\_addr | intnetp\_current\_hw\_addr(  u8\_t\*hardware\_address  ); | 获取正在使用的网络适配器的物理地址 |
| netp\_packet\_send | Intnetp\_packet\_send(  void\*buffer,intlen  ); | 发送一个数据帧 |
| htons | u16\_thtons(u16\_tn); | 将16位数值由主机字节序转换为网络字节序 |
| netp\_is\_eth\_addr\_broadcast | intnetp\_is\_eth\_addr\_broadcast(  structnetp\_eth\_addr\*mac\_address  ); | 判断一个MAC地址是否为广播地址 |
| netp\_eth\_addr\_cmp | intnetp\_eth\_addr\_cmp(  structnetp\_eth\_addr\*mac\_address1,  structnetp\_eth\_addr\*mac\_address2  ); | 判断两个MAC地址是否相同 |

#### 七.各模块推荐流程

1.EthernetV2数据帧发送流程

编码实现EthernetV2数据帧发送推荐使用如下流程：

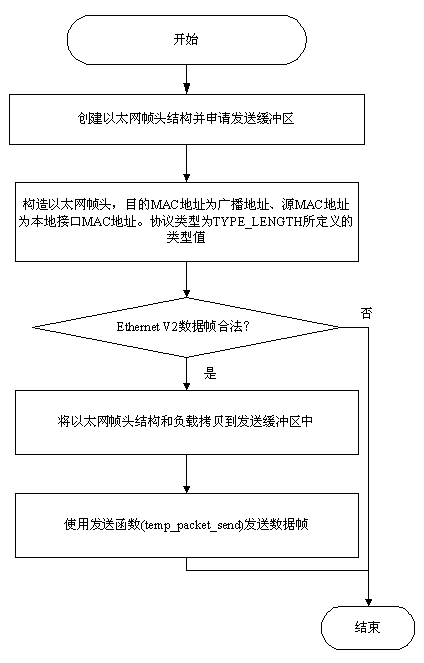


图1-9EthernetV2数据帧发送推荐流程

2.EthernetV2数据帧处理流程

编码实现处理EthernetV2输入数据帧推荐使用如下流程：

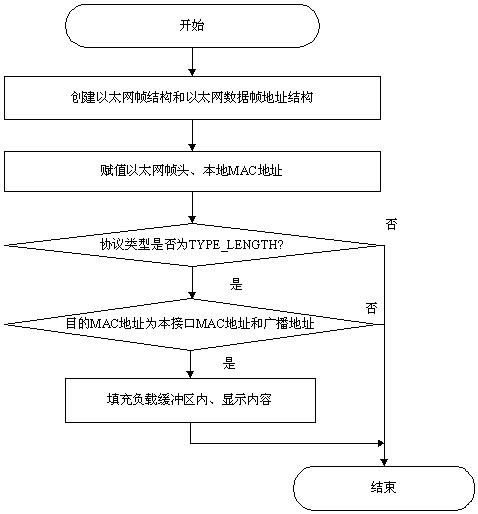


图1-10处理EthernetV2输入数据帧推荐流程

【实验步骤】

### 练习1领略真实的MAC帧

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。

1.主机B启动协议分析器，新建捕获窗口进行数据捕获并设置过滤条件（提取ICMP协议）。

2.主机Aping主机B，察看主机B协议分析器捕获的数据包，分析MAC帧格式。

3.将主机B的过滤器恢复为默认状态。

### 练习2理解MAC地址的作用

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B为例，其它组的操作参考主机A、B的操作。

1.主机B启动协议分析器，打开捕获窗口进行数据捕获并设置过滤条件（源MAC地址为主机A的MAC地址）。

2.主机Aping主机B。

3.主机B停止捕获数据，在捕获的数据中查找主机A所发送的ICMP数据帧，并分析该帧内容。

●记录实验结果

表1-3实验结果

|  |  |  |  |
| --- | --- | --- | --- |
|  | 本机MAC地址 | 源MAC地址 | 目的MAC地址 |
| 主机B |  |  |  |
| 主机D |  |  |  |
| 主机F |  |  |  |

思考问题：

1.MAC地址应用于TCP/IP协议模型的哪一层？

2.如何区分以太网的两种标准帧格式？

### 练习3编辑并发送MAC广播帧

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.主机E启动协议编辑器。

2.主机E编辑一个MAC帧：

目的MAC地址：FFFFFF-FFFFFF

源MAC地址：主机E的MAC地址

协议类型或数据长度：大于0x0600

数据字段：编辑长度在46—1500字节之间的数据

3.主机A、B、C、D、F启动协议分析器，打开捕获窗口进行数据捕获并设置过滤条件（源MAC地址为主机E的MAC地址）。

4.主机E发送已编辑好的数据帧。

5.主机A、B、C、D、F停止捕获数据，察看捕获到的数据中是否含有主机E所发送的数据帧。

●结合练习三的实验结果，简述FFFFFF-FFFFFF作为目的MAC地址的作用。

思考问题：

1.主机A、B、C、D、F是否可以收到主机E的广播帧？

2.说明MAC广播帧的范围？

### 练习4编辑并发送LLC帧

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。

1.主机A启动协议编辑器，并编写一个LLC帧。

目的MAC地址：主机B的MAC地址

源MAC地址：主机A的MAC地址

协议类型和数据长度：001F

控制字段：填写02（注：回车后变成0200，该帧变为信息帧，控制字段的长度变为2字节）

用户定义数据/数据字段：AAAAAAABBBBBBBCCCCCCCDDDDDD（注：长度为27个字节）

2.主机B启动协议分析器并开始捕获数据。

3.主机A发送编辑好的LLC帧。

4.主机B停止捕获数据，在捕获到的数据中查找主机A所发送的LLC帧，分析该帧内容。

●记录实验结果

表1-4实验结果

|  |  |  |
| --- | --- | --- |
| 帧类型 | 发送序号N（S） | 接受序号N（R） |
|  |  |  |

●简述“协议类型和数据长度”字段的两种含义。

5.将第1步中主机A已编辑好的数据帧修改为“无编号帧”(前两个比特位为1)，用户定义数据/数据字段修改为AAAAAAABBBBBBBCCCCCCCDDDDDDD（注：长度为28个字节），重做第2、3、4步。

思考问题：

1.如何编辑LLC无编号帧和LLC数据帧。

2.在协议分析端捕获到该帧，帧的长度是多少？由此理解以太网的最短帧长度。

3.为什么IEEE802标准将数据链路层分割为MAC子层和LLC子层？

4.为什么以太网有最短帧长度的要求？

### 练习5发送EthernetV2数据帧功能的实现

本练习将主机A、B、C、D、E、F作为一组进行实验。实验开始前，先单击“初始环境”。

在实验中主机的调试接口IP地址分别设为172.16.1.1n1、172.16.1.1n2、172.16.1.1n3、172.16.1.1n4、172.16.1.1n5、172.16.1.1n6(n为组别号，目的是为了避免IP地址冲突)，所有主机使用子网掩码255.255.255.0，默认网关设置为0.0.0.0。

以第一组为例：主机A将调试接口的IP地址设置为172.16.1.111、主机B使用处于连接状态的物理接口，将调试接口的IP设置为172.16.1.112、主机C将调试接口的IP地址设置为172.16.1.113、主机D使用处于连接状态的物理接口，将调试接口的IP地址设置为172.16.1.114、主机E使用处于连接状态的物理接口，将调试接口的IP地址设置为172.16.1.115、主机F将调试接口的IP地址设置为172.16.1.116。所有主机使用子网掩码255.255.255.0，默认网关设置为0.0.0.0。

1. 所有主机编码实现发送Ehternet V2数据帧

(1) 各主机使用VS2005软件打开实验平台安装目录下的JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_eth\_student\netproto\_eth\_student.sln文件，在源文件netproto\_eth\_student.c的函数netp\_eth\_output\_student内编写实现代码。

注：若实验平台安装在C盘，则目录为：C:\Program Files\JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_eth\_student\netproto\_eth\_student.sln。

(2) 参考实验原理Ethernet V2数据帧发送推荐流程图给出的流程，分析已经存在的代码。

已经存在的代码定义了一个以太网数据帧头部结构和一个能容纳以太网帧头和负载的发送缓冲区send\_buff，另外还实现了将以太网帧头结构和负载拷贝到发送缓冲区的过程。

(3) 构造、填充以太网数据帧头

构造并填充一个以太网数据帧头。目的MAC地址设置为广播地址即FF-FF-FF-FF-FF-FF，可以使用netp\_set\_eth\_addr函数设置MAC地址。源MAC地址设置为本接口的MAC地址，可以使用netp\_current\_hw\_addr函数获取本接口的MAC地址。协议类型或数据长度字段值应设置为0x0806，表示上层协议为arp协议，可以使用MAC\_PROTO\_ARP宏。

(4) 判断是否为合法的Ethernet V2数据帧

根据实验原理关于MAC帧格式的介绍，判断要发送的数据帧是不是合法的Ethernet V2数据帧，即“协议类型或数据长度”字段值是否大于1536。

(5) 使用发送函数netp\_packet\_send发送数据帧。

2. 当完成代码编写后，所有主机打开协议分析器，开始捕获数据。

3. 所有主机调试并运行程序。

4. 各主机停止数据捕获，观察实验现象。

● 捕获到的数据帧，“协议类型或数据长度”字段值是什么？

5.参考代码如下：

/\*\*  
\*\brief编辑并发送一个EthernetV2数据帧  
\*  
\*由学生完成这个函数，发送一个EthernetV2数据帧。主线程将会调用这个函数。  
\*/  
void  
netp\_eth\_output\_student()  
{  
//创建以太网帧头结构并申请发送缓冲区  
structnetp\_eth\_headereth\_header;  
u8\_tsend\_buff[ETH\_HEADER\_LEN+PAYLOAD\_LEN];  
  
//构造以太网帧头，目的MAC地址为广播地址、源MAC地址为本接口MAC地址  
//协议类型为TYPE\_LENGTH所定义的类型值  
netp\_set\_eth\_addr(&eth\_header.dest\_address,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF);  
netp\_current\_hw\_addr(&(eth\_header.sour\_address));  
eth\_header.type=htons(TYPE\_LENGTH);  
  
//判断是否为合法的EthernetV2数据帧  
if(ntohs(eth\_header.type)<=1536){  
printf("Ethernetv2数据帧的协议类型或数据长度字段值要求大于1536，发送失败！\n");  
return;  
}

//将以太网帧头结构和负载拷贝到发送缓冲区中  
memcpy(send\_buff,&eth\_header,sizeof(structnetp\_eth\_header));  
memcpy(send\_buff+ETH\_HEADER\_LEN,PAYLOAD\_DATA,PAYLOAD\_LEN);  
  
//使用发送函数(netp\_packet\_send)发送数据帧  
if(!netp\_packet\_send(send\_buff,sizeof(send\_buff))){  
printf("发送MAC数据帧----成功！\n");  
}else{  
printf("发送MAC数据帧----失败！\n");  
}  
}

### 练习6处理EthernetV2输入数据帧功能的实现

本练习将主机A、B、C、D、E、F作为一组进行实验。实验开始前，先单击“初始环境”。

该练习需要在前一个练习的基础上进行。

在实验中主机的调试接口IP地址分别设为172.16.0.1n1、172.16.0.1n2、172.16.0.1n3、172.16.0.1n4、172.16.0.1n5、172.16.0.1n6(n为组别号，目的是为了避免IP地址冲突)，所有主机使用子网掩码255.255.255.0，默认网关设置为0.0.0.0。

以第一组为例，主机A将新接口的IP地址设置为172.16.0.111、主机B使用处于连接状态的物理接口，将新接口的IP设置为172.16.0.112、主机C将新接口的IP地址设置为172.16.0.113、主机D使用处于连接状态的物理接口，将新接口的IP地址设置为172.16.0.114、主机E使用处于连接状态的物理接口，将新接口的IP地址设置为172.16.0.115、主机F将新接口的IP地址设置为172.16.0.116。所有主机使用子网掩码255.255.255.0，默认网关设置为0.0.0.0。

1. 所有主机编码实现处理Ethernet V2输入数据帧功能

(1) 各主机使用VS2005软件打开实验平台安装目录下的JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_eth\_student\netproto\_eth\_student.sln文件，在源文件netproto\_eth\_student.c的函数netp\_eth\_input\_student内编写实现代码，参考实验原理处理Ethernet V2输入数据帧推荐流程图给出的流程，思考代码编写方案。首先分析netproto\_eth\_student.c文件中已经给出的代码。

注：若实验平台安装在C盘，则目录为：C:\Program Files\JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_eth\_student\netproto\_eth\_student.sln。

已经存在的代码定义了一个以太网数据帧头部结构和以太网数据帧地址结构，实现了从输入缓冲区中拷贝以太网帧头结构，获取本接口MAC地址的过程。最后的几行代码填充了负载缓冲区，显示其内容。

(2) 提取“协议类型或数据长度”字段值为TYPE\_LENGTH的数据帧

通过判断以太网帧中的“协议类型或数据长度”字段值是否为TYPE\_LENGTH（自定义上层协议）来过滤以太网数据帧。如果接收到的数据包不满足条件，则应该返回NETP\_PUSH\_TO\_LWIP交给协议栈处理。

(3) 提取“目的MAC地址”字段值为本接口地址或广播地址的数据帧

通过判断以太网帧中的“目的MAC地址”字段值是否为本接口MAC地址或广播地址（目的MAC地址为全1）来过滤不是发送给本接口的数据帧。如果接收到的数据包不满足条件，则应该返回NETP\_PUSH\_TO\_LWIP交给协议栈处理。

2. 所有主机调试并运行程序

将在安装目录JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_eth\_student\netproto\_eth\_student.sln下的头文件netproto\_eth\_student.h文件中PAYLOAD\_DATA定义的内容修改为自己要发送的负载信息，例如“I am host A”。

● 你收到的负载内容是什么？

3.参考代码如下：

/\*\*  
\*\brief当有数据帧到达时，将调用这个函数  
\*  
\*\parampacket指向接收到的数据  
\*\parampacket\_len数据帧的长度  
\*  
\*\return一个put\_to\_lwip类型的返回值。返回NETP\_PUT\_LWIP表示处理数据后将数据帧交给上层协议栈继续处理  
\*返回NETP\_NO\_PUT\_LIWP表示不将数据帧交给上层协议栈处理。  
\*/  
enumpush\_to\_lwip  
netp\_eth\_input\_student(void\*packet,intpacket\_len)  
{  
//创建以太网帧头结构和以太网数据帧地址结构  
structnetp\_eth\_headereth\_header;  
structnetp\_eth\_addrmy\_mac\_address;  
charpayload\_buff[MAX\_PAYLOAD\_LEN]={0};  
  
//赋值以太网帧头、本机MAC地址  
memcpy(&eth\_header,packet,ETH\_HEADER\_LEN);  
netp\_current\_hw\_addr(&my\_mac\_address);  
  
//判断“协议类型与数据长度”值是否为TYPE\_LENGTH  
if(TYPE\_LENGTH!=ntohs(eth\_header.type)){  
returnNETP\_PUSH\_TO\_LWIP;  
}  
  
//判断目的MAC地址是否为本接口地址或者为广播地址  
if(!(netp\_is\_eth\_addr\_broadcast((structnetp\_eth\_addr\*)packet)||  
!netp\_eth\_addr\_cmp(&my\_mac\_address,(structnetp\_eth\_addr\*)packet))){  
returnNETP\_PUSH\_TO\_LWIP;  
}  
  
//填充负载缓冲区，显示内容  
memset(payload\_buff,0,sizeof(payload\_buff));  
memcpy(payload\_buff,(u8\_t\*)packet+ETH\_HEADER\_LEN,MAX\_PAYLOAD\_LEN-1);  
printf("收到协议类型为%d的以太网数据帧，负载内容为%s\n",  
TYPE\_LENGTH,(char\*)payload\_buff);  
returnNETP\_NO\_PUSH\_LIWP;  
}

## 实验2地址解析协议（ARP）

【实验目的】

1.掌握ARP协议的报文格式

2.掌握ARP协议的工作原理

3.理解ARP高速缓存的作用

4.掌握ARP请求和应答的实现方法

5.掌握ARP缓存表的维护过程

【学时分配】

2学时

【实验环境】

该实验采用网络结构二

【实验原理】

#### 一.物理地址与逻辑地址

##### 1.物理地址

物理地址是节点的地址，由它所在的局域网或广域网定义。物理地址包含在数据链路层的帧中。物理地址是最低一级的地址。

物理地址的长度和格式是可变的，取决于具体的网络。以太网使用写在网络接口卡（NIC）上的6字节的标识作为物理地址。

物理地址可以是单播地址（一个接收者）、多播地址（一组接收者）或广播地址（由网络中的所有主机接收）。有些网络不支持多播或广播地址，当需要把帧发送给一组主机或所有主机时，多播地址或广播地址就需要用单播地址来模拟。

##### 2.逻辑地址

在互联网的环境中仅使用物理地址是不合适的，因为不同网络可以使用不同的地址格式。因此，需要一种通用的编址系统，用来惟一地标识每一台主机，而不管底层使用什么样的物理网络。

逻辑地址就是为此目的而设计的。目前Internet上的逻辑地址是32位地址，通常称为IP地址，可以用来标识连接在Internet上的每一台主机。在Internet上没有两个主机具有同样的IP地址。

逻辑地址可以是单播地址、多播地址和广播地址。其中广播地址有一些局限性。在实验三中将详细介绍这三种类型的地址。

#### 二.ARP协议简介

Internet是由各种各样的物理网络通过使用诸如路由器之类的设备连接在一起组成的。主机发送一个数据包到另一台主机时可能要经过多种不同的物理网络。主机和路由器都是在网络层通过逻辑地址来识别的，这个地址是在全世界范围内是惟一的。然而，数据包是通过物理网络传递的。在物理网络中，主机和路由器通过其物理地址来识别的，其范围限于本地网络中。物理地址和逻辑地址是两种不同的标识符。这就意味着将一个数据包传递到一个主机或路由器需要进行两级寻址：逻辑地址和物理地址。需要能将一个逻辑地址映射到相应的物理地址。

ARP协议（地址解析协议）是“AddressResolutionProtocol”的缩写。所谓“地址解析”就是主机在发送帧前将目的逻辑地址转换成目的物理地址的过程。在使用TCP/IP协议的以太网中，ARP协议完成将IP地址映射到MAC地址的过程。

#### 三.ARP报文格式

下图为ARP数据报的报文格式：

|  |  |  |
| --- | --- | --- |
| 硬件类型（16位） | | 协议类型（16位） |
| 硬件地址长度（8位） | 协议地址长度（8位） | 操作码（16位） |
| 发送端硬件地址  （例如，对以太网是6字节） | | |
| 发送端逻辑地址  （例如，对IP是4字节） | | |
| 目的端硬件地址  （例如，对以太网是6字节）  （在请求帧中不填入） | | |
| 目的端逻辑地址  （例如，对IP是4字节） | | |

图2-1ARP报文格式

ARP报文格式具有如下的一些字段：

●硬件类型：这是16位字段，用来定义运行ARP的网络的类型。每一个局域网基于其类型被指派给一个整数。例如，以太网的硬件类型是1。ARP可用在任何网络上。

●协议类型：这是16位字段。用来定义协议的类型。例如，对IPv4协议，这个字段的值是0x0800。ARP可用于任何高层协议。

●硬件地址长度：这是一个8位字段，用来定义以字节为单位的物理地址长度。例如，以太网物理地址为6字节，所对应的硬件地址长度值为6。

●协议地址长度：标识用于该数据包的逻辑地址的长度，用十进制标识，单位为一个字节，例如，IPv4为4个字节，所对应的协议地址长度值为4。

●操作码：这是16位字段，用来定义数据包的类型。已定义了两种类型：为1时表示ARP请求，为2时表示ARP应答。

●发送端硬件地址：这是可变长度字段，用来定义发送端的物理地址。对于以太网这个字段是6字节长。

●发送端逻辑地址：这是可变长度字段，用来定义发送端的逻辑地址。对于逻辑地址为IP地址的网络，该字段长度为4字节。

●目的端硬件地址：这是可变长度字段，用来定义目标的物理地址。对于ARP请求，字段是全0，因为发送端不知道目标的物理地址（该字段长度为可变，如以太网硬件地址为6个字节）。

●目的端逻辑地址：这是可变长度字段，用来定义目标的逻辑地址（该字段长度为可变，如IPv4协议的逻辑地址为4个字节）。

#### 四.ARP封装

ARP数据报直接封装在数据链路帧中。例如，在下图中，ARP数据包封装在以太网的帧中。类型字段值为0x0806指出了此帧所携带的数据是ARP数据包。

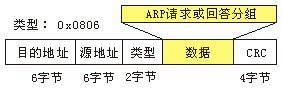


图2-2ARP数据包的封装

#### 五.ARP的运行过程

数据包传输过程可分为如下步骤：

1.发送端知道目的端的IP地址。

2.IP要求ARP创建一个ARP请求报文，其中包含了发送方的物理地址、发送方的IP地址和目的端的IP地址。目的端的物理地址用0填充。

3.将报文传递到数据链路层，并在该层中用发送方的物理地址作为源地址，用物理广播地址作为目的地址，将其封装在一个帧中。

4.因为该帧中包含了一个广播目的地址，所以同一链路中的每个主机或路由器都接收到这个帧。所有接收到该帧的主机都将其传递到ARP层进行处理。除了目的端主机以外的所有主机都丢弃该报文。

5.目的端主机用一个包含其物理地址的ARP应答报文做出响应，并对该报文进行单播。

6.发送方接收到这个应答报文，这样它就知道了目标主机的物理地址。

ARP地址解析过程如下图所示。

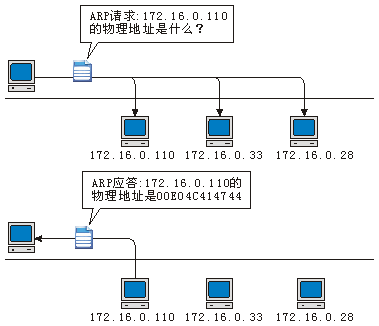


图2-3ARP地址解析过程

#### 六.ARP高速缓存

在真正的协议实现中，并不是每次发送IP报文前都需要发送ARP请求报文来获取目的MAC地址。在大多数的系统中都存在着一个ARP缓存表。记录着一段时间内曾经获取过的MAC地址和IP地址的映射关系，如下图所示：

|  |  |
| --- | --- |
| IP地址 | MAC地址 |
| 202.98.13.1  202.98.13.2  202.98.13.3  …… | 00-E0-4C-3D-89-76  00-E0-4C-3D-C5-03  00-E0-4C-4D-BA-92  …… |

图2-4ARP高速缓存

发送IP数据报前先对ARP缓存表进行查找，查看目的MAC地址是否存在于缓存表中，如果存在，则不需要发送ARP请求报文而直接使用此地址进行IP数据包的发送。如果不存在，则发送ARP请求报文，在收到ARP应答报文之后，使用应答报文中的目的MAC地址发送IP数据包，并将目的MAC地址存于ARP缓存表中供以后使用。

另外，ARP缓存表采用老化机制，在一段时间内如果表中的某一项没有使用，就会被删除，这样可以大大减少ARP缓存表的长度，加快查询速度。

下图描述了ARP高速缓存的使用与更新过程：

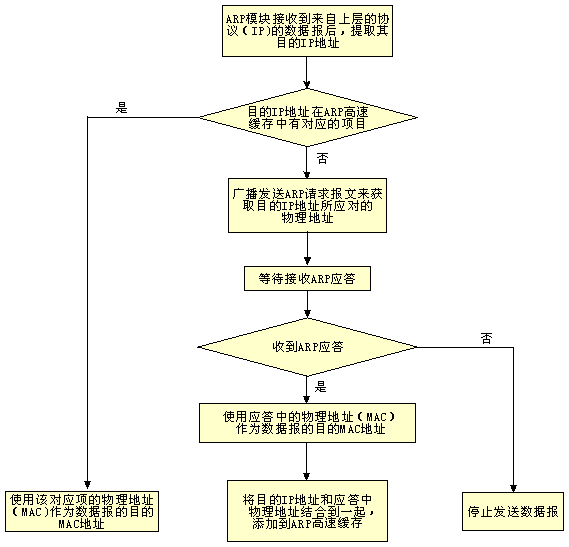


图2-5ARP高速缓存的使用与更新过程

#### 七.代理ARP

代理ARP可用来产生划分子网的效应。如果ARP请求是从一个网络中的主机发往另一个网络中的主机，那么连接这两个网络的路由器就可以回答该请求，当这个路由器收到真正的IP数据包时，它就把该数据包发送给相应的主机或路由器。

例如，在下图所示的网络中，安装在右边主机上的代理ARP应答对目标IP地址为141.23.56.23的ARP请求。

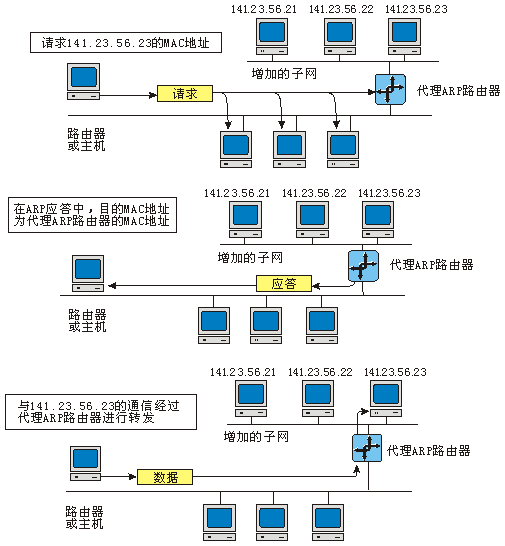


图2-6代理ARP

#### 八.协议栈实现代码解析

本实验将通过对安装目录

JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_arp\_student\netproto\_arp\_student下的netproto\_arp\_student.h和netproto\_arp\_student.c两个文件进行编码，完成协议栈中arp协议的实现。

netproto\_arp\_student.h文件中定义了arp协议实现相关数值，关键代码如下所示：

#defineMAC\_PROTO\_ARP 0x0806  
#defineARP\_HWTYPE\_ETH 0x0001  
#defineARP\_PROTOTYPE\_IP 0x0800  
#defineARP\_HWADDR\_LEN\_ETH 0x06  
#defineARP\_PROTOADDR\_LEN\_IP 0x04  
#defineARP\_OPCODE\_REQUEST 0x0001  
#defineARP\_OPCODE\_RESPONSE 0x0002

#defineARP\_DEST\_IP "0.0.0.0"

这段代码定义了8个宏，他们代表的含义如下表所示：

表2-1netproto\_arp\_student.h中定义的宏

|  |  |  |
| --- | --- | --- |
| 宏 | 值 | 描述 |
| MAC\_PROTO\_ARP | 0x0806 | 定义以太网帧中的“协议类型或数据长度”字段的值 |
| ARP\_HWTYPE\_ETH | 0x0001 | 定义arp包头中“硬件类型”字段值 |
| ARP\_PROTOTYPE\_IP | 0x0800 | 定义arp包头中“协议类型”字段值 |
| ARP\_HWADDR\_LEN\_ETH | 0x06 | 定义arp包头中“硬件地址长度”字段值 |
| ARP\_PROTOADDR\_LEN\_IP | 0x04 | 定义arp包头中“协议地址长度”字段值 |
| ARP\_OPCODE\_REQUEST | 0x0001 | 定义arp请求数据包头中“操作码”字段值 |
| ARP\_OPCODE\_RESPONSE | 0x0002 | 定义arp应答数据包头中“操作码”字段值 |
| ARP\_DEST\_IP | "0.0.0.0" | 用点分十进制表示的IP地址，定义arp包头中“目的端逻辑地址”字段值 |

在实验的编码过程中，应该使用这些宏对相应的变量进行赋值。

netproto\_arp\_student.c文件是协议栈中arp协议的实现部分，其中定义了1个全局数组以及3个函数。下面分别介绍这些协议栈的实现部分。

全局数组netp\_arp\_table是arp协议的缓存表，拥有NETP\_ARP\_TABLE\_SIZE个netp\_arp\_table\_item元素。其中NETP\_ARP\_TABLE\_SIZE是arp缓存表的条目数，默认值为10，同学们不需要修改。netp\_arp\_table\_item是一个结构体，代表了arp缓存表中的一个条目，包括一个物理地址hardware\_addr和一个IP地址ip\_address，其定义如下：

structnetp\_arp\_table\_item  
{  
u8\_thardware\_addr[ETH\_ADDRESS\_LEN];  
structip\_addrip\_address;  
};

需要根据arp协议的实现原理编写代码来维护arp缓存表。

函数display\_arp\_table的功能是显示arp缓存表中的条目，在实验中可以直接调用该函数，便于查看arp缓存表中的内容。

函数netp\_arp\_output\_student的功能是构造并发送一个arp请求数据包。这个函数的编码工作需要由学生完成。

当有数据到达本机网络接口时，函数netp\_arp\_input\_student将被调用，并传递给这个函数原始数据。在本实验中该函数需要完成两个功能，一是处理针对本机的arp请求数据包，二是处理针对本机的arp应答数据包。处理针对本机的arp请求数据包时，应该发送相应的arp应答数据包。处理针对本机的arp应答数据包时，应该更新arp缓存表。这个函数的编码工作需要由学生完成。该函数的返回值为push\_to\_lwip的枚举类型值，push\_to\_lwip的定义如下：

enumpush\_to\_lwip{  
NETP\_PUSH\_TO\_LWIP,//数据处理完成后，交给lwIP继续处理  
NETP\_NO\_PUSH\_LIWP//数据处理完成后，不交给lwIP继续处理  
//本层处理完毕以后数据包被丢弃  
};

返回NETP\_PUSH\_TO\_LWIP表示这个数据帧应该提交给协议栈上层继续处理，而返回NETP\_NO\_PUSH\_LIWP则表示不需要提交给协议栈上层处理，本层处理完毕后，这个数据帧将被丢弃。需要根据正确的逻辑关系返回适当的值，使协议栈正常工作。

在编码过程中可能会遇到一些结构体、宏和函数，下表对他们进行介绍：

表2-2实验涉及的结构体和函数

| 结构体/函数 | 声明或定义 | 描述 |
| --- | --- | --- |
| structnetp\_arp\_table\_item | structnetp\_arp\_table\_item{  u8\_thardware\_addr[ETH\_ADDRESS\_LEN];  structip\_addrip\_address;  }; | ARP缓存表中一个条目的结构 |
| structin\_addr | structin\_addr{  u32\_ts\_addr;  }; | 32位地址 |
| structnetp\_eth\_header | structnetp\_eth\_header{  u8\_tdest\_address[ETH\_ADDRESS\_LEN];  u8\_tsour\_address[ETH\_ADDRESS\_LEN];  u16\_ttype;  }; | 以太网帧头结构 |
| structnetp\_arp\_header | structnetp\_arp\_header{  u16\_thardware\_type;  u16\_tproto\_type;  u8\_thw\_addr\_len;  u8\_tproto\_addr\_len;  u16\_topcode;  u8\_tsrc\_hw\_addr[ETH\_ADDRESS\_LEN];  structip\_addrsrc\_ip\_addr;  u8\_tdest\_hw\_addr[ETH\_ADDRESS\_LEN];  structip\_addrdest\_ip\_addr;  }; | arp协议包头结构 |
| NETP\_ARP\_TABLE\_SIZE | #defineNETP\_ARP\_TABLE\_SIZE10 | ARP缓存表中条目的个数 |
| ETH\_ADDRESS\_LEN | #defineETH\_ADDRESS\_LEN6 | 以太网帧地址长度 |
| ETH\_HEADER\_LEN | #defineETH\_HEADER\_LEN14 | 以太网帧头长度 |
| ARP\_HEADER\_LEN | #defineARP\_HEADER\_LEN28 | ARP数据包头长度 |
| display\_arp\_table | voiddisplay\_arp\_table(); | 将ARP缓存表的内容显示到标准输出 |
| netp\_current\_hw\_addr | intnetp\_current\_hw\_addr(  u8\_t\*hardware\_address  ); | 获取正在使用的网络适配器的物理地址 |
| netp\_current\_ip\_addr | u32\_tnetp\_current\_ip\_addr(); | 获取当前正在使用的网络适配器的IP地址 |
| htons | u16\_thtons(u16\_tn); | 将16位数值由主机字节序转换为网络字节序 |
| inet\_addr | u32\_tinet\_addr(constchar\*cp); | 将ASCII编码的Internet地址转换成为网络字节序地址 |
| netp\_packet\_send | intnetp\_packet\_send(  void\*buffer,intlen  ); | 使用当前正在使用的网络接口发送一个数据帧 |

#### 九.各模块推荐流程

##### 1.arp请求发送流程

编码实现arp请求数据包发送推荐使用如下流程：

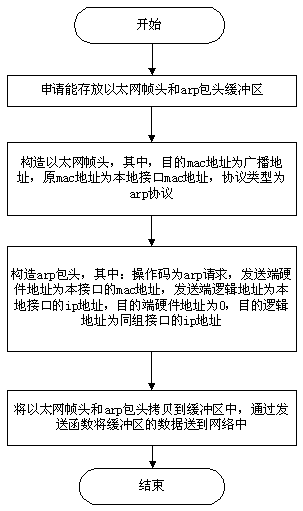


图2-7arp请求数据包发送推荐流程

##### 2.输入arp数据包处理流程

编码实现处理arp输入数据包推荐使用如下流程：

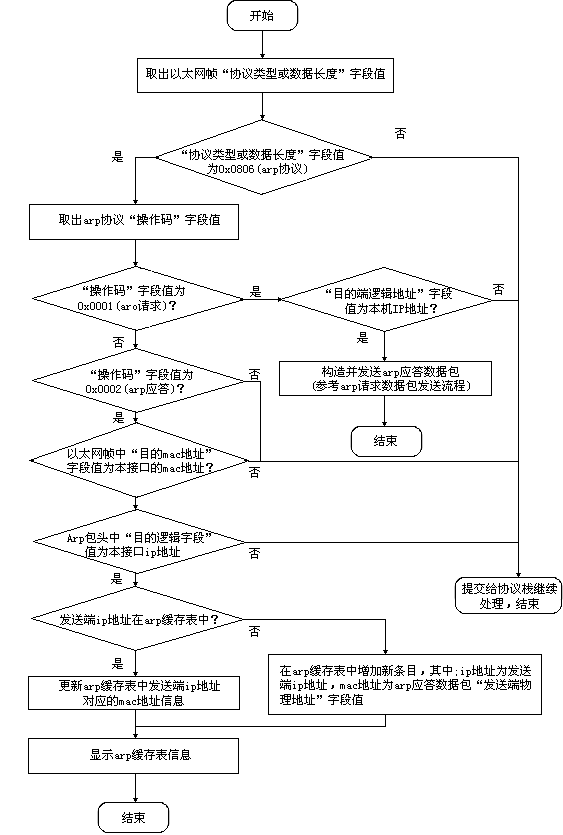


图2-8处理arp输入数据包推荐流程

【实验步骤】

### 练习1领略真实的ARP（同一子网）

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.主机A、B、C、D、E、F启动协议分析器，打开捕获窗口进行数据捕获并设置过滤条件（提取ARP、ICMP）。

2.主机A、B、C、D、E、F在命令行下运行“arp-d”命令，清空ARP高速缓存。

3.主机Aping主机D（172.16.1.4）。

主机Bping主机C（172.16.1.3）。

主机Eping主机F（172.16.0.3）。

4.主机A、B、C、D、E、F停止捕获数据，并立即在命令行下运行“arp-a”命令察看ARP高速缓存。

●ARP高速缓存表由哪几项组成？

●结合协议分析器上采集到的ARP报文和ARP高速缓存表中新增加的条目，简述ARP协议的报文交互过程以及ARP高速缓存表的更新过程。

### 练习2编辑并发送ARP报文（同一子网）

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.在主机E上启动协议编辑器，并编辑一个ARP请求报文。其中：

MAC层：

目的MAC地址：设置为FFFFFF-FFFFFF

源MAC地址：设置为主机E的MAC地址

协议类型或数据长度：0806

ARP层：

发送端硬件地址：设置为主机E的MAC地址

发送端逻辑地址：设置为主机E的IP地址（172.16.0.2）

目的端硬件地址：设置为000000-000000

目的端逻辑地址：设置为主机F的IP地址（172.16.0.3）

2.主机A、B、C、D、F启动协议分析器，打开捕获窗口进行数据捕获并设置过滤条件（提取ARP协议）。

3.主机B、E、F在命令行下运行“arp-d”命令，清空ARP高速缓存。主机E发送已编辑好的ARP报文。

4.主机A、B、C、D、F停止捕获数据，分析捕获到的数据，进一步体会ARP报文交互过程。

思考问题：

1.哪些主机收到了ARP请求包，哪个主机给出了ARP响应包？

2.主机A、C、D是否收到ARP请求包，为什么？

### 练习3跨路由地址解析（不同子网）

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.主机B在命令行方式下输入staticroute\_config命令，开启静态路由服务。

2.主机A、B、C、D、E、F在命令行下运行“arp-d”命令，清空ARP高速缓存。

3.主机A、B、C、D、E、F重新启动协议分析器，打开捕获窗口进行数据捕获并设置过滤条件（提取ARP、ICMP）。

4.主机Aping主机E（172.16.0.2）。

5.主机A、B、C、D、E、F停止数据捕获，察看协议分析器中采集到的ARP报文，并回答以下问题：

●单一ARP请求报文是否能够跨越子网进行地址解析？为什么？

●ARP地址解析在跨越子网的通信中所起到的作用？

6.主机B在命令行方式下输入recover\_config命令，停止静态路由服务。

思考问题：

1.哪些主机收到了ARP请求包，哪台主机给出了ARP响应包？

2.比较ARP协议在同网段内解析和跨网段的解析有何异同点？

3.ARP数据包的长度是固定的吗？试加以解释。

4.试解释为什么ARP高速缓存每存入一个项目就要设置10-20分钟的超时计时器。这个时间设置得太大或太小会出现什么问题？

5.至少举出两种不需要发送ARP请求数据包的情况。

### 练习4发送arp请求功能的实现

本练习将主机A、C和D作为一组，主机B、E和F作为一组。现仅以主机A、C、D所在组为例，其它组的操作参考主机A、C、D所在组的操作。实验开始前，先单击“初始环境”。

在实验中，主机A将调试接口的IP地址设置为172.16.1.12、主机B使用物理接口2，将调试接口的IP设置为172.16.0.11、主机C将调试接口的IP地址设置为172.16.1.13、主机D使用处于连接状态的物理接口，将调试接口的IP地址设置为172.16.1.14、主机E使用处于连接状态的物理接口，将调试接口的IP地址设置为172.16.0.12、主机F将调试接口的IP地址设置为172.16.0.13。所有主机使用子网掩码255.255.255.0，默认网关设置为0.0.0.0。

1. 所有主机编码实现发送arp请求数据包

(1) 各主机使用VS2005软件打开实验平台安装目录下的JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_arp\_student\netproto\_arp\_student.sln文件，在源文件netproto\_arp\_student.c的函数netp\_arp\_output\_student内编写实现代码。

注：若实验平台安装在C盘，则目录为：C:\Program Files\JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_arp\_student\netproto\_arp\_student.sln。

(2) 参考实验原理arp请求数据包发送推荐流程图给出的流程，分析已经存在的代码。

已经存在的代码定义了一个能容纳arp请求数据包的缓冲区arp\_buffer、一个以太网帧头结构变量eth\_header和一个arp包头结构变量arp\_header、以及将缓冲区数据发送到网络中的实现。

(3) 构造、填充以太网数据帧头

构造并填充一个以太网数据帧头。目的MAC地址设置为广播地址即FF-FF-FF-FF-FF-FF。源MAC地址设置为本接口的MAC地址，可以使用netp\_current\_hw\_addr函数获取本接口的MAC地址。协议类型或数据长度字段值应设置为0x0806，表示上层协议为arp协议，可以使用MAC\_PROTO\_ARP宏。

(4) 构造、填充ARP数据包头

构造并填充一个ARP数据包头。ARP数据包头中各字段值的填充如下：

硬件类型值为0x0001，可以使用ARP\_HWTYPE\_ETH宏。

协议类型值为0x0800，可以使用ARP\_PROTOTYPE\_IP宏。

硬件地址长度值为0x06，可以使用ARP\_HWADDR\_LEN\_ETH宏。

协议地址长度值为0x04，可以使用ARP\_PROTOADDR\_LEN\_IP宏。

操作码值为0x0001，可以使用ARP\_OPCODE\_REQUEST宏。

发送端硬件地址值为本接口的MAC地址，可以使用netp\_current\_hw\_addr函数获取本接口的MAC地址。

发送端逻辑地址值为本接口的IP地址，可以使用netp\_current\_ip\_addr函数获取本接口的IP地址。

目的端硬件地址值为0。

主机A使用172.16.1.3作为目的端逻辑地址、主机C使用172.16.1.4作为目的端逻辑地址、主机D使用172.16.1.2作为目的端逻辑地址。

(5) 将构造完成的以太网数据帧头和ARP数据包头拷贝到缓冲区。

2. 所有主机打开协议分析器，开始捕获数据

3. 所有主机调试并运行程序

4. 各主机停止数据捕获，观察实验现象

5.参考代码如下：

/\*\*  
\*\brief编辑并发送一个ARP数据包  
\*  
\*由学生完成这个函数，发送一个ARP请求数据包。主线程将会调用这个函数。  
\*/  
void  
netp\_arp\_output\_student()  
{  
structnetp\_eth\_headereth\_header;  
structnetp\_arp\_headerarp\_header;  
u8\_tarp\_buffer[ETH\_HEADER\_LEN+ARP\_HEADER\_LEN];

//构造、填充以太网数据帧头  
u8\_tdest\_mac\_address[ETH\_ADDRESS\_LEN]={0xFF,0xFF,0xFF,0xFF,0xFF,0xFF};  
structnetp\_eth\_addrsour\_mac\_address;  
netp\_current\_hw\_addr(&sour\_mac\_address);

memcpy(eth\_header.dest\_address.addr,dest\_mac\_address,ETH\_ADDRESS\_LEN);  
memcpy(eth\_header.sour\_address.addr,sour\_mac\_address.addr,ETH\_ADDRESS\_LEN);  
eth\_header.type=htons(MAC\_PROTO\_ARP);

//构造、填充ARP数据包头  
arp\_header.hardware\_type=htons(ARP\_HWTYPE\_ETH);  
arp\_header.proto\_type=htons(ARP\_PROTOTYPE\_IP);  
arp\_header.hw\_addr\_len=ARP\_HWADDR\_LEN\_ETH;  
arp\_header.proto\_addr\_len=ARP\_PROTOADDR\_LEN\_IP;  
arp\_header.opcode=htons(ARP\_OPCODE\_REQUEST);  
memcpy(arp\_header.src\_hw\_addr.addr,sour\_mac\_address.addr,ETH\_ADDRESS\_LEN);  
arp\_header.src\_ip\_addr.addr=netp\_current\_ip\_addr();  
memset(arp\_header.dest\_hw\_addr.addr,0,ETH\_ADDRESS\_LEN);  
arp\_header.dest\_ip\_addr.addr=inet\_addr(ARP\_DEST\_IP);

memcpy(arp\_buffer,&eth\_header,ETH\_HEADER\_LEN);  
memcpy(arp\_buffer+ETH\_HEADER\_LEN,&arp\_header,ARP\_HEADER\_LEN);

//发送ARP数据帧  
if(!netp\_packet\_send(arp\_buffer,ETH\_HEADER\_LEN+ARP\_HEADER\_LEN)){  
printf("发送ARP请求到%s----成功！\n",ARP\_DEST\_IP);  
}else{  
printf("发送ARP请求到%s----失败！\n",ARP\_DEST\_IP);  
}  
}

### 练习5处理输入的arp数据包功能的实现

本练习将主机A、C和D作为一组，主机B、E和F作为一组。现仅以主机A、C、D所在组为例，其它组的操作参考主机A、C、D所在组的操作。实验开始前，先单击“初始环境”。

该练习需要在前一个练习的基础上进行。

在实验中，主机A将调试接口的IP地址设置为172.16.1.12、主机B使用物理接口2，将调试接口的IP设置为172.16.0.11、主机C将调试接口的IP地址设置为172.16.1.13、主机D使用处于连接状态的物理接口，将调试接口的IP地址设置为172.16.1.14、主机E使用处于连接状态的物理接口，将调试接口的IP地址设置为172.16.0.12、主机F将调试接口的IP地址设置为172.16.0.13。所有主机使用子网掩码255.255.255.0，默认网关设置为0.0.0.0。

1. 所有主机编码实现arp数据包的过滤

(1) 各主机使用VS2005软件打开实验平台安装目录下的JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_arp\_student\netproto\_arp\_student.sln文件，在源文件netproto\_arp\_student.c的函数netp\_arp\_input\_student内编写实现代码，参考实验原理处理arp输入数据包推荐流程图给出的流程，思考代码编写方案。

注：若实验平台安装在C盘，则目录为：C:\ProgramFiles\JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_arp\_student\netproto\_arp\_student.sln。

(2) 过滤arp数据包

通过判断以太网帧中的“协议类型或数据长度”字段值是否为0x0806（宏MAC\_PROTO\_ARP定义该数值）来过滤arp数据包。如果接收到的数据包不是arp数据包，则应该返回NETP\_PUSH\_TO\_LWIP交给协议栈处理。

2. 处理arp请求数据包

(1) 判断arp类型

通过arp包头中的“操作码”字段的值来判断arp数据包类型，如果该值为0x0001（宏ARP\_OPCODE\_REQUEST定义了该数值），则这个arp包为arp请求报文，应该返回arp应答数据包。

(2) 构造arp应答数据包的以太网帧头

使用已经定义的变量response\_eth\_header来构造以太网帧头。其中：目的MAC地址为arp请求数据包头中的发送端硬件地址；源MAC地址为本接口的MAC地址；“协议类型与数据长度”字段值为0x0806（宏MAC\_PROTO\_ARP定义该数值）。

(3) 构造arp应答数据包的arp包头

使用已经定义的变量response\_arp\_header来构造以太网帧头。其中：

硬件类型值为0x0001，可以使用ARP\_HWTYPE\_ETH宏。

协议类型值为0x0800，可以使用ARP\_PROTOTYPE\_IP宏。

硬件地址长度值为0x06，可以使用ARP\_HWADDR\_LEN\_ETH宏。

协议地址长度值为0x04，可以使用ARP\_PROTOADDR\_LEN\_IP宏。

操作码值为0x0002，可以使用ARP\_OPCODE\_RESPONSE宏。

发送端硬件地址值为本接口的MAC地址，可以使用netp\_current\_hw\_addr函数获取本接口的MAC地址。

发送端逻辑地址值为本接口的IP地址，可以使用netp\_current\_ip\_addr函数获取本接口的IP地址。

目的端硬件地址值为arp请求数据包头中的发送端硬件地址。

目的端逻辑地址为arp请求数据包头中的发送端逻辑地址。

3. 处理arp应答数据包

(1) 判断arp类型

通过arp包头中的“操作码”字段的值来判断arp数据包类型，如果该值为0x0001（宏ARP\_OPCODE\_REQUEST定义了该数值），则这个arp包为arp请求报文，应该返回arp应答数据包。

(2) 判断arp应答数据包是否发送给本接口

判断以太网帧“目的MAC地址”字段值是否为本接口的MAC地址，如果不是则返回NETP\_PUSH\_TO\_LWIP交给协议栈处理。

判断arp应答数据包头中“目的端逻辑地址”字段值是否为本接口的IP地址，如果不是则返回返回NETP\_PUSH\_TO\_LWIP交给协议栈处理。

(3) 更新arp缓存表

根据arp应答数据包中的发送端逻辑地址和发送端硬件地址字段的值来更新arp缓存表，arp缓存表是命名为netp\_arp\_table的一个数组，如果发送端逻辑地址已经在arp缓存表中，则只需要更新相应的MAC地址即可。如果发送端逻辑地址没有在arp缓存表中，则需要在新的位置创建arp缓存表条目，其中ip地址为发送端逻辑地址，MAC地址为发送端硬件地址。

4. 主机C和主机D修改练习四编写的程序，将目的端逻辑地址设置为172.16.1.12

5. 所有主机打开协议分析器，开始捕获数据

6. 所有主机调试并运行程序

7. 各主机停止数据捕获，观察实验现象

8.参考代码如下。

/\*\*  
\*\brief当有数据帧到达时，将调用这个函数  
\*  
\*\parampacket指向接收到的数据  
\*\parampacket\_len数据帧的长度  
\*  
\*\return一个put\_to\_lwip类型的返回值。返回NETP\_PUT\_LWIP表示处理数据后将数据帧交给上层协议栈继续处理  
\*返回NETP\_NO\_PUT\_LIWP表示不将数据帧交给上层协议栈处理。  
\*/  
enumpush\_to\_lwip  
netp\_arp\_input\_student(void\*packet,intpacket\_len)  
{  
structnetp\_eth\_addrmy\_mac\_address;//本机mac地址  
u32\_tmy\_ip\_address;//本机IP地址  
structnetp\_eth\_headereth\_header;  
structnetp\_arp\_headerarp\_header;  
inti;  
staticintarp\_table\_index;

memcpy(&eth\_header,packet,ETH\_HEADER\_LEN);  
//过滤ARP数据帧，如果不是ARP数据帧，直接返回不进行任何处理  
if(eth\_header.type!=htons(MAC\_PROTO\_ARP)){  
returnNETP\_PUSH\_TO\_LWIP;  
}

//填充arp\_header结构  
memcpy(&arp\_header,(u8\_t\*)packet+ETH\_HEADER\_LEN,ARP\_HEADER\_LEN);

//获取本机物理地址  
netp\_current\_hw\_addr(&my\_mac\_address);

//获取本机IP地址  
my\_ip\_address=netp\_current\_ip\_addr();

//判断arp类型  
switch(ntohs(arp\_header.opcode)){  
caseARP\_OPCODE\_REQUEST://arp请求  
if(arp\_header.dest\_ip\_addr.addr==my\_ip\_address){  
structnetp\_eth\_headerresponse\_eth\_header;  
structnetp\_arp\_headerresponse\_arp\_header;  
u8\_tarp\_buffer[ETH\_HEADER\_LEN+ARP\_HEADER\_LEN];

printf("接收到一个来自%s的ARP请求数据包。\n",inet\_ntoa(\*(structin\_addr\*)&arp\_header.dest\_ip\_addr));

//构造ARP应答数据帧  
//构造、填充以太网数据帧头  
memcpy(response\_eth\_header.dest\_address.addr,arp\_header.src\_hw\_addr.addr,ETH\_ADDRESS\_LEN);  
memcpy(response\_eth\_header.sour\_address.addr,my\_mac\_address.addr,ETH\_ADDRESS\_LEN);

response\_eth\_header.type=htons(MAC\_PROTO\_ARP);

//构造、填充ARP数据包头  
response\_arp\_header.hardware\_type=htons(ARP\_HWTYPE\_ETH);  
response\_arp\_header.proto\_type=htons(ARP\_PROTOTYPE\_IP);  
response\_arp\_header.hw\_addr\_len=ARP\_HWADDR\_LEN\_ETH;  
response\_arp\_header.proto\_addr\_len=ARP\_PROTOADDR\_LEN\_IP;  
response\_arp\_header.opcode=htons(ARP\_OPCODE\_RESPONSE);

memcpy(response\_arp\_header.src\_hw\_addr.addr,my\_mac\_address.addr,ETH\_ADDRESS\_LEN);  
response\_arp\_header.src\_ip\_addr.addr=my\_ip\_address;  
memcpy(response\_arp\_header.dest\_hw\_addr.addr,arp\_header.src\_hw\_addr.addr,ETH\_ADDRESS\_LEN);  
response\_arp\_header.dest\_ip\_addr.addr=arp\_header.src\_ip\_addr.addr;

//将以太网数据帧头和ARP数据包头拷贝到缓冲区中  
memcpy(arp\_buffer,&response\_eth\_header,ETH\_HEADER\_LEN);  
memcpy(arp\_buffer+ETH\_HEADER\_LEN,&response\_arp\_header,ARP\_HEADER\_LEN);

//发送数据帧  
if(!netp\_packet\_send(arp\_buffer,ETH\_HEADER\_LEN+ARP\_HEADER\_LEN)){  
printf("向%s发送ARP应答数据包成功！\n",inet\_ntoa(\*(structin\_addr\*)&arp\_header.dest\_ip\_addr));  
}else{  
printf("向%s发送ARP应答数据包失败！\n",inet\_ntoa(\*(structin\_addr\*)&arp\_header.dest\_ip\_addr));  
returnNETP\_PUSH\_TO\_LWIP;  
}  
returnNETP\_NO\_PUSH\_LIWP;  
}else{  
returnNETP\_PUSH\_TO\_LWIP;  
}  
caseARP\_OPCODE\_RESPONSE://arp应答  
//判断目的MAC地址是否为本接口的MAC地址  
if(0!=memcmp(my\_mac\_address.addr,packet,ETH\_ADDRESS\_LEN)){  
returnNETP\_PUSH\_TO\_LWIP;  
}

//判断目的逻辑地址是否为本接口的IP地址  
if(arp\_header.dest\_ip\_addr.addr!=my\_ip\_address){  
returnNETP\_PUSH\_TO\_LWIP;  
}

printf("接收到%s发送的ARP应答数据包\n",inet\_ntoa(\*(structin\_addr\*)&arp\_header.src\_ip\_addr));

//更新ARP缓存表  
i=0;  
while(i<NETP\_ARP\_TABLE\_SIZE){  
if(netp\_arp\_table[i].ip\_address.addr==arp\_header.src\_ip\_addr.addr){  
memcpy(netp\_arp\_table[i].hardware\_addr.addr,arp\_header.src\_hw\_addr.addr,ETH\_ADDRESS\_LEN);  
break;  
}  
i++;  
if(i==NETP\_ARP\_TABLE\_SIZE){  
netp\_arp\_table[arp\_table\_index].ip\_address=arp\_header.src\_ip\_addr;  
memcpy(netp\_arp\_table[arp\_table\_index].hardware\_addr.addr,arp\_header.src\_hw\_addr.addr,ETH\_ADDRESS\_LEN);  
arp\_table\_index=(arp\_table\_index++)%NETP\_ARP\_TABLE\_SIZE;  
}  
}

display\_arp\_table();

returnNETP\_NO\_PUSH\_LIWP;  
default:  
returnNETP\_PUSH\_TO\_LWIP;  
}  
returnNETP\_PUSH\_TO\_LWIP;  
}

## 实验3网际协议（IP）

【实验目的】

1.掌握IP数据报的报文格式

2.掌握IP校验和计算方法

3.掌握子网掩码和路由转发

4.理解特殊IP地址的含义

5.理解IP分片过程

6.理解协议栈对IP协议的处理方法

7.理解IP路由表作用以及IP路由表的管理

【学时分配】

4学时

【实验环境】

该实验采用网络结构二

【实验原理】

### 一.IP协议简介

IP（网际协议）是TCP/IP协议族中最核心的协议，它负责将数据包从源点交付到终点。所有的TCP、UDP、ICMP及IGMP数据都以IP数据报格式传输。IP协议提供不可靠、无连接的数据报传送服务，即它对数据进行“尽力传输”，只负责将数据包发送到目的主机，不管传输正确与否，不做验证、不发确认、也不保证IP数据包到达顺序，将纠错重传问题交由传输层来解决。

### 二.IP地址及其表示方法

IP地址是网际协议地址（InternetProtocoladdress）的简称。一个IP地址惟一地标识了Internet上的一台主机。通信时要使用IP地址来指定相应的目的主机。

#### 1.地址空间

地址空间就是协议所使用的地址总数。如果协议使用N位来定义地址，每一位可以有两种不同的值（1或0），那么地址空间就是2N。

IP使用32位地址，这表示地址空间是232，或4294967296（超过40亿个）。从理论上讲，可以有超过40亿个设备连接到Internet。但是，实际的数字要远小于这个数值。

#### 2.IP地址的表示方法

IP地址有三种常用的表示方法：二进制表示方法、点分十进制表示方法和十六进制表示方法。

●二进制表示方法：

在二进制表示方法中，用一个32位的比特序列表示IP地址，为了使这个地址有更好的可读性，通常在每个字节（8位）之间加上一个或多个空格做分隔。例如：

10000001000011100000011000011111

●点分十进制表示方法：

为了使32位地址更加简洁和更容易阅读，Internet的地址通常写成用小数点把各字节分隔开的形式。每个字节用一个十进制数表示，这个数小于256。例如：

129.14.6.31

●十六进制表示方法：

有时会见到十六进制表示方法的IP地址。每一个十六进制数字等效于4个位。例如：

0x810E061F

#### 3.IP地址的分类

IP地址分成5类：A类，B类，C类，D类和E类。其中A类、B类和C类地址是基本的Internet地址，是用户使用的地址，D类地址用于广播，E类地址为保留地址。

下图描述了IP地址的二进制表示方法的分类：

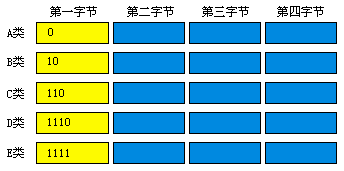


图3-1在二进制记法中找出IP地址的类别

下图描述了IP地址的十进制表示方法的分类：

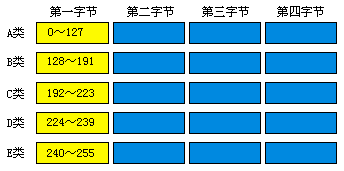


图3-2在点分十进制记法中找出IP地址的类别

#### 4.网络号和主机号

在分类编址的A类，B类和C类地址中，IP地址可划分为网络号（net-id）和主机号（host-id）。这两部分长度都是可变的，取决于地址的类型。下图给出了网络号和主机号所占的字节。应该注意的是，D类地址和E类地址不划分网络号和主机号。

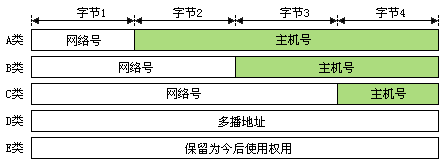


图3-3网络号和主机号

#### 5.地址类和地址块

分类编址将每一类地址都划分为固定数目的地址块，并且每一个地址块的大小都是固定的。

A类地址共分为128个地址块，每个地址块都包含有16777216个地址。这表明要使用这类地址的机构一定是一个非常庞大的机构。但是，每个地址块中的地址数比机构的地址需求大的多，所以，在这类地址中，许多地址都被浪费了。

B类地址共划分为16384个地址块，每个地址块都包含有65536个地址。这个地址数往往大于中等规模机构的地址需求，所以，在这类地址中，也有许多地址被浪费了。

C类地址共划分为2097152个地址块，每个地址块都包含有256个地址。这类地址中的地址数对大多数机构来说是不够用的，因此许多机构不太愿意要这类地址。

D类地址只有一个地址块。它用来进行多播。

E类地址只有一个地址块。它是保留地址。

### 三.特殊的IP地址

#### 1.特殊的IP地址

●网络地址：主机号为全“0”的IP地址不分配给任何主机，而是作为网络本身的标识。

例：主机202.198.151.136所在网络地址为202.198.151.0。

●直接广播地址：主机号为全“1”的IP地址不分配给任何主机，用作广播地址，目的地址为直接广播地址的数据包传递给该网络中的所有节点（能否执行广播，则依赖于支撑的物理网络是否具有广播功能）。

例：202.198.151.136所在网络的广播地址为202.198.151.255。

●有限广播地址：32位为全“1”的IP地址（255.255.255.255）称为有限广播地址，通常由无盘工作站启动时使用，希望从网络IP地址服务器处获得一个IP地址。

●主机本身地址：32位全“0”的IP地址（0.0.0.0）称为主机本身地址。

●回环地址：127.0.0.1称为回环地址，常用于本机上软件测试和本机上网络应用程序之间的通信地址。

#### 2.专用IP地址

随着Internet的飞速发展，IP地址资源已经开始告急，专用IP地址的使用是解决IP地址紧缺的一种方法。原理是定义两类IP地址：

●全局IP地址：用于Internet上的公共主机；

●专用IP地址：仅用于专用网内部的本地主机。

公共主机和本地主机可以共存于同一网络并进行互访，而大多数路由器不转发携带专用IP地址的数据包。本地主机必须经网络地址转换服务器（NAT或代理服务器）才能访问Internet。专用IP地址为：

10.0.0.0-10.255.255.2551个A类网络

172.16.0.0-172.31.255.25516个连续的B类网络

192.168.0.0-192.168.255.255256个连续的C类网络

### 四.子网划分

为了便于网络的管理、提高系统的可靠性、改进系统性能、克服简单局域网的技术条件限制、通过设置不同访问权限来增强系统的安全保障，人们采用了划分子网的办法将网络进一步划分成独立的组成部分。

现在的主机都要求支持子网编址。不是把IP地址看成由单纯的一个网络号和一个主机号组成，而是把主机号再分成一个子网号和一个主机号。例如，把一个B类网络地址的16位主机号分成8位子网号和8位主机号如下图所示：

|  |  |  |
| --- | --- | --- |
| 16位 | 8位 | 8位 |
| 网络号 | 子网号 | 主机号 |

图3-4子网划分

这样就允许有254个子网，每个子网可以有254台主机。对A类和B类网络，许多管理员采用自然的划分方法，即以8位为单位划分子网地址和主机号。这样用点分十进制方法表示的IP地址就可以比较容易确定子网号。但是，并不要求A类或B类地址的子网划分都要以字节为分界限。子网对外部路由器来说隐藏了内部网络组织的细节。

主机除了知道IP地址以外，还需要知道IP中有多少位用于子网号，多少位用于主机号。这是通过使用一个称为“子网掩码”的32位值来完成的。其中值为1的位留给网络号和子网号，为0的位留给主机号。

给定IP地址和子网掩码以后，主机就可以确定IP数据包的目的是本子网中的主机、本网络中其它子网中的主机还是其它网络上的主机。

如果知道本机的IP地址，那么就知道它是否为A类、B类或C类地址（从IP地址的高位可以得知），也就知道网络号和子网号之间的分界线。而根据子网掩码就可知道子网号与主机号之间的分界线。

子网掩码除了可以如IP地址一样用“点分十进制”方式表示外，还可以在IP地址后用一个斜线（/）后面写明子网掩码的位数的方法来表示。如：192.168.1.25/24表示IP地址192.168.1.25的掩码为255.255.255.0。

### 五.IP报文格式

IP数据报格式如下图所示，它是由IP首部与数据组成的。IP首部长度通常为20字节。如果含有选项字段，IP首部长度将会大于20字节，但不会超过60字节。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号  （4位） | 首部长度（4位） | 区分服务（8位） | 总长度（16位） | |
| 标识（16位） | | | 标志（3位） | 偏移量（13位） |
| 生存时间（8位） | | 高层协议类型（8位） | 首部校验和（16位） | |
| 源IP地址（32位） | | | | |
| 目的IP地址（32位） | | | | |
| IP选项（如果有） | | | | |
| 数据 | | | | |

图3-5IP报文格式

在IP首部中各个字段的意义如下：

●版本号：这个字段定义了IP的版本。目前主流的是版本4（IPv4），但它正逐渐地被版本6（IPv6）所替代。

●首部长度：由于IP选项字段的存在，所以IP首部长度是可变的。该字段用4位来定义首部长度。将该值乘4可得到用字节表示的长度，所以IP首部长度为20～60个字节。

●区分服务：该字段以前叫做服务类型（在RFC791中定义），是由3位优先域、4位服务类型域和1位未用位(该位必须置为0)组成，其中4位服务类型分别代表：最小时延、最大吞吐量、最高可靠性和最小费用。该字段在RFC2474中被重新定义,在新的定义中将该字段命名为区分服务，其中包含一个6位的区分服务码点(DSCP)字段和一个2位的未用位（CU）字段。如下图所示：

|  |  |
| --- | --- |
| 区分服务码点（6位） | 未用位（2位） |

图3-6RFC2474定义的区分服务字段

前6位的区分服务码点的值用来映射一个底层的服务，它决定了每一跳行为。在RFC3168中，将区分服务字段中的未用位字段定义为显示拥塞通告（ECN）字段，ECN字段包括2个子字段，分别为：ECT字段和CE字段，如下图所示：

|  |  |  |
| --- | --- | --- |
| 区分服务码点（6位） | ECT（1位） | CE（1位） |

图3-7RFC3168定义的区分服务字段

ECN字段用于指定发送数据的主机是否支持拥塞通告以及指示网络中是否有拥塞发生。ECN通过两个子字段的不同值组合来做到这一点：

00：发送主机不支持ECN。

01或10：发送主机支持ECN。

11：路由器正在经历拥塞。

●总长度：该字段以字节为单位定义IP数据报的总长度（首部加上数据）。要得到IP上层数据的长度，只需从总长度中减去首部长度即可。

●标识：每一个IP数据包在发送时被给定特有的标识值。如果数据包必须被分割成碎片以适应支持小型数据包的网络，那么每一个碎片中都设置相同的标识号码。

●标志：标志由3位组成，第1位保留；第2位为不分片标志，表示此数据包不可以被分片；第3位为更多分片标志，表示在分片包之后还有分片，即此包不是最后分片。

●偏移量：如果数据包是一个分片包，该域指明了当前分片包在与其它分片包被重新组装成一个单独数据包时，应该位于数据包的什么位置。该域的值以8字节为单位。

●生存时间：该字段表明数据包保存的生存时间，单位为秒，在实际的应用中，生存时间是按照数据包经过路由器的跳数计算的。通常生存时间的值是32、64、128。

●高层协议类型：该字段定义了使用IP层服务的较高层协议。一个IP数据报能封装来自诸如TCP、UDP、ICMP和IGMP等较高层协议的数据。

●首部校验和：IP首部校验和只对首部内容进行错误检测，并不包括数据包的其它内容。校验和采用16位反码求和的算法。

●源IP地址：该字段定义了源主机的IP地址。在IP数据包从源主机传送到目的主机期间，该字段保持不变。

●目的IP地址：该字段定义了目的主机的IP地址。在IP数据报从源主机传送到目的主机期间该字段保持不变。

●IP选项：这个字段是可选项。它们通常用于网络测试和调试。虽然可选项不是IP头部所必需的部分，但要求IP软件能够处理它们。目前，这些选项定义如下：

(1)安全和处理限制（用于军事领域）；

(2)记录路径（让每个路由器记下它的IP地址）；

(3)时间戳（让每个路由器都记下它的IP地址和时间）；

(4)宽松的源站选路（为数据报指定一系列必须经过的IP地址）；

(5)严格的源站选路（与宽松的源站选路类似，但是要求只能经过指定的这些地址，不能经过其它的地址）。

### 六.IP封装

IP数据报直接封装到数据链路层帧中，其封装方法如下图所示：

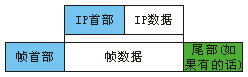


图3-8IP封装

### 七.IP数据报分片

数据包可能通过多个不同的网络。每一个路由器把收到的帧进行拆装，再进行处理，然后又封装成另一个帧。收到帧的格式与长度取决于这个帧刚刚经过的物理网络所使用的协议。发送出去的帧格式与长度则取决于这个帧将要经过的物理网络所使用的协议。例如，如果路由器把以太网连接到一个广域网，那么这个路由器收到的帧是以太网的格式，而发送的帧是广域网的格式。

#### 1.最大传送单元（MTU）

不同的网络所能传送的数据包的最大长度是不同的，这个最大长度叫做最大传送单元（MTU），这是由网络所使用的硬件与软件所决定的。每种网络的数据链路层都有自己的帧格式，其中有一个字段是“数据字段最大长度”。当数据包封装成帧时，数据包的总长度必须小于这个数据字段最大长度，如下图所示。

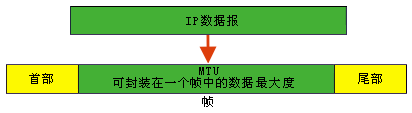


图3-9MTU

对于不同的物理网络协议，MTU的值是不同的。下表给出了不同协议的MTU值。

表3-1不同网络的MTU值

|  |  |  |  |
| --- | --- | --- | --- |
| 协议 | MTU | 协议 | MTU |
| 超级通道（Hyperchannel） | 65535 | 以太网 | 1500 |
| 令牌环（16Mbps） | 17914 | X.25 | 576 |
| 令牌环（4Mbps） | 4464 | PPP | 296 |
| FDDI | 4352 | —— | —— |

为了使IP协议与物理网络无关，IP协议不考虑底层网络的MTU，只是规定IP数据包的最大负载长度为65535字节。对于物理网络，如果数据包的长度超过了MTU，就要把数据包进行分割，使它们能够通过这些网络，这就叫做分片。源主机的传输层会自动对数据包进行分片工作，把数据包划分成IP协议和数据链路层都可接受的大小。

当数据包被分片时，每一个数据包片有它自己的首部，其中大部分的字段都是重复的，但有些是不同的。如果已经分片的数据包要经过更小MTU的网络，那么这些已经分片的数据包还可再进行分片，数据包在到达最后终点之前可以经过多次的分片。

数据包可以被源主机或在其路径上的路由器进行分片。但是数据包的重装却只能在目的主机上进行。由于被分片的数据包可能会通过不同的路由到达目的主机，所以应当在最后的目的主机上进行重装。

当数据包被分片时，首部中的一些字段会被复制到所有的分片中（选项字段可以被复制，也可不被复制）。有三个字段是与数据包分片相关的，这三个字段是：标识字段、标志字段和偏移量字段。当然，不管是否进行分片，校验和的值总是要重新计算的。

对于最大传输单元（MTU）还可以理解为某层协议报文可携带数据的最大长度，对于不同协议，对应的MTU可能会有不同的值和计算方法。如我们常说的“以太网MTU为1500”，说的是以太网MAC帧可封装的最大数据长度，其只包含IP报头及其上层协议数据的总长度。对于IP协议的MTU值应为65535-20，也只是说一个IP数据报（包含IP包头的长度和IP上层协议数据的长度），最多可以携带65535个字节的数据。

#### 2.与分片有关的字段

与数据包的分片和重装有关的三个字段是：标识字段、标志字段和偏移量字段。

●标识：IP数据包的标识字段值与源IP地址惟一地确定了一个数据包。IP协议使用一个计数器来保证每个数据包标识的惟一性。当IP协议发送数据包时，就把这个计数器的当前值复制到标识字段中，并把这个记数器的值加1。当数据包被分片时，标识字段的值就复制到所有的分片中。这样所有的分片具有相同的标识。这个标识号在终点重装数据包时很有用。终点会将所有具有相同标识号的分片组装成一个数据包。

●标志：这是一个长度为3位的字段。第一位保留。第二位是不分片位。若这个位为1，就表示不能把该数据包进行分片。若无法把这个数据包通过任何可用的物理网络进行转发，就丢弃这个数据包，并向源主机发送ICMP差错报文（关于ICMP差错报文，参见实验4）。若这个位为0，则在需要时可把这个数据包进行分片。第三位是还有分片位。若这个位是1，则表示这个数据包不是最后的分片，在这个分片后面还有其它分片。若这个位是0，则表示这是最后的分片。标志字段如下图所示：

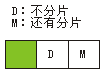


图3-10标志字段

●偏移量：偏移量字段表示一个分片在整个数据包中的相对位置。以8字节为度量单位。下图表示具有4000字节长度的数据包被划分为三个分片。在原始数据包中的数据编号是0～3999。第一个分片携带的数据是字节0～1399。对于这个数据包，偏移量是0／8＝0。第二个分片携带的数据是字节1400～2799；对于这个数据包，偏移量是1400／8＝175。最后，第三个分片携带的数据是字节2800～3999。对于这个数据包，偏移量是2800／8＝350。如下图。

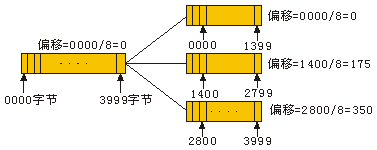


图3-11分片举例

偏移量字段的长度只有13位，它不能表示超过8191的字节数，所以偏移量是以8字节为单位的。因此，把数据包进行分片的主机或路由器必须选择一个能够被8整除的长度来为数据包分片。

### 八.IP数据报校验和

大多数TCP/IP协议族中的协议采用的差错检测方法是校验和。校验和能够识别数据包在传输过程中是否受到损伤。校验和是在数据包上附加的信息。

在发送端先计算校验和，并把得到的结果与数据包一起发送出去。接收端对整个数据包重复进行同样的计算。若得到的结果正确则接受这个数据包；否则就把它丢弃。

#### 1.在发送端计算校验和

在发送端，将数据包按16位长度分段。把这些段用反码算数运算相加，将相加后得到的和再取反码就得出了校验和。

#### 2.在接收端计算校验和

接收端把收到的数据包按16位长度分段，并把这些段相加。把得到的和取反码。若结果为0，则接受这个数据包；否则就拒绝这个数据包。

下图用图解的方法描述了发送端和接收端计算校验和的过程。

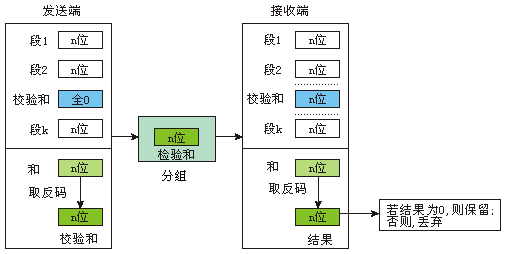


图3-12校验和的概念

#### 3.IP数据包中的校验和

IP数据包的校验和只校验IP首部部分而不校验IP数据部分。这是因为所有将数据封装在IP数据报中的高层协议，都有覆盖整个数据包的校验和。因此，IP数据报的校验和就不必再检验所封装的数据部分。其次，每经过一个路由器，IP数据包的首部就要改变一次，但数据部分不改变。因此若校验和包含数据部分，这就意味着路由器要花费很多时间计算没有改变的数据部分的校验和。

### 九.交付与转发

所谓交付，是指在网络层的控制下，一个数据包被底层网络处理的方式。例如，面向无连接的服务和面向连接的服务，以及直接交付和间接交付等。

所谓转发是指数据包被交付到下一站的方式。

#### 1.交付

网络层控制底层物理网络对数据包的处理过程，这种过程叫做交付。在交付过程中有几个重要概念，即连接类型，直接与间接交付。

#### 2.连接类型

数据包在网络层的交付可以用面向连接的服务或无连接服务来完成。

(1)面向连接服务

在面向连接服务的情况下，本地网络层协议在发送数据包之前先要和远地网络层协议建立一条连接。当连接建立后，一系列的数据包就从源点一个接一个地发送到终点。在这种情况下，各个数据包之间存在着一种关系。它们都沿着同一条路径按序发送。一个数据包与走在它前面的数据包以及与走在它后面的数据包在逻辑上是连接在一起的。若报文中的所有数据包都已被交付，连接就终止了。

在面向连接服务的情况下，对从同一个源点到同一个终点的一系列数据包来说，其路由的确定只需进行一次。路由器不需要对每一个单个的数据包重新计算路由。

(2)无连接服务

在无连接服务的情况下，网络协议独立地对待每一个数据包，而每一个数据包与任何其它数据包都没有关系。一个报文中的各数据包到它们的终点可以走相同的路径也可以走不同的路径。在无连接服务的情况下，对一个数据包的路由是由每一个路由器单独地确定。IP协议是无连接协议，它提供无连接服务。

#### 3.直接交付与间接交付

把一个数据包交付到它最后的终点有两种方式：直接交付和间接交付。

(1)直接交付

在直接交付的情况下，数据包的最后终点是与交付者连接在同一个网络上的主机。当数据包的源点和终点都在同一个物理网络上时，或者交付是在最后一个路由器与目的主机之间进行时，就为直接交付，如下图所示。

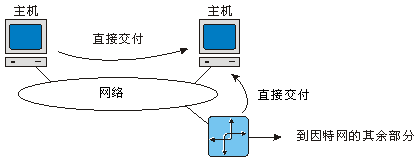


图3-13直接交付

发送端很容易确定交付是否为直接交付。发送端可以提取出终点的网络地址（使用掩码），并与它所连接的网络的网络地址相比较。若匹配，交付就是直接的。

在直接交付时，发送端使用目的IP地址找出目的物理地址。然后把目的IP地址和目的物理地址一起交付给数据链路层，以便进行实际的交付。这个过程叫做把IP地址映射到物理地址。地址解析协议（ARP）可以动态地把IP地址映射到相应的物理地址。

(2)间接交付

如果目的主机与交付者不在同一个网络上，数据包就要进行间接交付。在间接交付时，数据包从一个路由器传送到另一个路由器，直到这个数据包到达与最后的终点连接在同一个网络上的路由器为止。如下图所示。

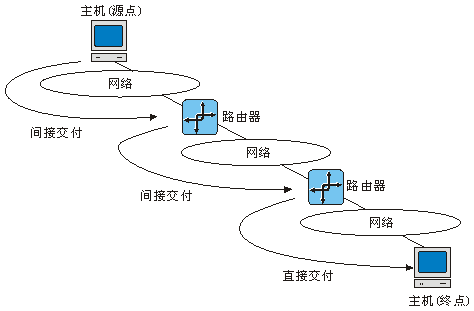


图3-14间接交付

交付永远包括一个直接交付，以及零个或多个间接交付。最后的交付就是直接交付。

在间接交付中，发送端使用目的IP地址和路由表来找出下一个路由器的IP地址。然后使用ARP协议找出下一个路由器的物理地址。在直接交付时，地址的映射是在终点的IP地址与终点的物理地址之间进行的。在间接交付时，地址的映射是在下一个路由器的IP地址与下一个路由器的物理地址之间进行的。

#### 4.转发

转发表示把数据包放到去终点的路由上。进行转发就要求主机或路由器装有路由表。当主机有数据包要发送时，或路由器收到数据包要进行转发时，就要查找路由表，以便找出到达最后终点的路由。但是，这种简单的转发方法，在今天的互联网中已经变得不可能了，因为路由表中的项目数已使得路由表的查找效率非常低。

#### 5.转发技术

使用一些转发技术可以减小路由表的大小，同时还能够处理一些安全问题。下面为三种简单的转发技术。

(1)下一跳方法

下一跳方法就是在路由表中只保留下一跳的地址，而不是保留完整路由信息。下图所描绘的例子说明了路由表如何被简化。

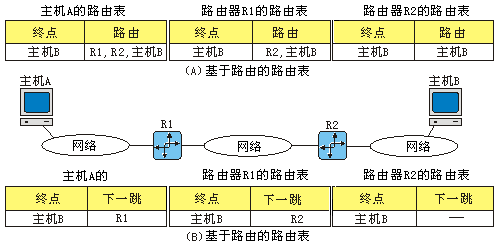


图3-15下一跳方法

(2)特定网络方法

特定网络方法不是对连接在同一个物理网络上的所有主机都设置一个路由表项，而是仅用一个路由表项来代表这个目的网络本身的地址。换言之，这种技术把连接在同一个网络上的所有主机看成是一个路由表项。如下图所示：

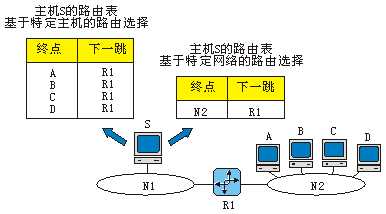


图3-16特定网络方法

(3)特定主机方法

特定主机方法将目的主机地址在路由表中全部给出。这种方法与特定网络方法的思想相反。这是用牺牲效率来换取其它一些优点的方法。虽然把主机地址放在路由表中会降低效率，但有时管理员还是想对路由选择作出更多的控制。如下图所示：

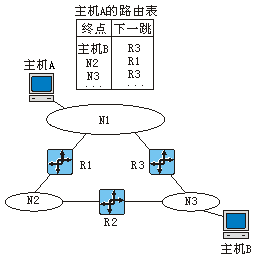


图3-17特定主机方法

### 十.路由选择

路由选择涉及的问题是创建和维护路由表。

#### 1.静态和动态路由选择

主机或路由器要转发IP数据包就要有一个路由表，并给每一个终点或一组终点设置一个路由表项。路由表可以是静态的，也可以是动态的。

(1)静态路由表

静态路由表中的路由信息是由管理员手动设置的。管理员把到达每一个终点的路由输入到路由表中。路由表的更新由管理员手动完成。

静态路由表用在不会经常变动的小型互联网中，或用在故障查找的试验互联网中。在大的互联网中很少使用静态路由表。

(2)动态路由表

动态路由表使用一种动态路由选择协议，如RIP、OSPF（关于这两个协议，请参考实验十六，实验十七）。当网络中发生变化时，动态路由选择协议就更新所有路由器中的路由表。

为了有效地交付IP数据包，规模较大的网络一般使用动态路由表。

#### 2.路由表

下图给出了路由表中常用的一些列：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 掩码 | 网络地址 | 下一跳地址 | 接口号 | 标志 | 引用计数 | 使用 |
| …… | …… | …… | …… | …… | …… | …… |

图3-18路由表中常用的列

●掩码：这列定义应用到路由表项的掩码。

●网络地址：这列定义数据包要交付到的网络地址。在特定主机路由选择中，这个字段定义目的主机的地址。

●下一跳地址：这列定义了数据包要交付到的下一跳路由器的地址。

●接口：这列指出接口的名字。

●标志：这列定义了五个标志：U（Up，工作），G（网关），H（特定主机），D（因改变路由而增加的）和M（因改变路由而修改的）。

路由选择过程如下图所示：

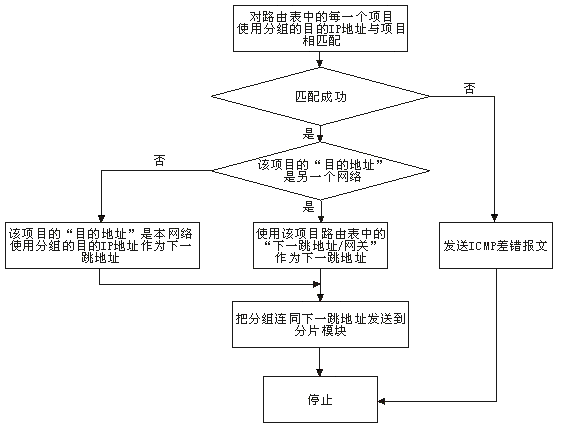


图3-19路由表使用过程

### 十一.协议栈实现代码解析

本实验将通过对安装目录JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_ip\_student\netproto\_ip\_student下的netproto\_ip\_student.h、netproto\_ip\_shudent.c和安装目录JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_ip\_student\netproto\_iproute\_student下的netproto\_iproute\_student.h、netproto\_iproute\_student.c四个文件进行编码，完成协议栈中IP协议的实现。

netproto\_ip\_student.h和netproto\_ip\_shudent.c文件用于实现IP数据包发送和接收。其中，netproto\_ip\_student.h文件中定义了IP协议实现相关数值以及IP的负载内容、负载长度，关键代码如下所示：

#defineIP\_VERSION4  
#defineIP\_HEADERLEN5  
#defineIP\_TTLNETP\_TTL  
#defineIP\_DSNETP\_DS  
#defineIP\_CUS\_PROTO93//IP自定义高层协议  
#defineIP\_DEST\_ADDR"0.0.0.0"

#definePAYLOAD\_DATA"Hello,World!"  
#definePAYLOAD\_LENsizeof(PAYLOAD\_DATA)

这段代码定义了8个宏，他们代表的含义如下表所示：

表3-2netproto\_ip\_student.h中定义的宏

|  |  |  |
| --- | --- | --- |
| 宏 | 值 | 描述 |
| IP\_VERSION | 4 | 定义IP包头中“版本号”字段的值，必须为4 |
| IP\_HEADERLEN | 5 | 定义IP包头中“首部长度”字段值，一般为5 |
| IP\_TTL | NETP\_TTL | 定义IP包头中“生存时间”字段值，默认为128 |
| IP\_DS | NETP\_DS | 定义IP包头中“区分服务”字段值，默认为0 |
| IP\_CUS\_PROTO | 93 | 定义IP包头中“高层协议类型”字段值，该值为自定义数值，学生可以更改 |
| IP\_DEST\_ADDR | "0.0.0.0" | 定义IP请求数据包头中“目的IP地址”字段值 |
| PAYLOAD\_DATA | "Hello,World!" | 自定义IP负载内容 |
| PAYLOAD\_LEN | sizeof(PAYLOAD\_DATA) | 自定义IP负载的长度 |

在实验的编码过程中，应该使用这些宏对相应的变量进行赋值。

netproto\_ip\_shudent.c文件是协议栈中IP数据包发送和接收的实现部分，其中定义了2个函数。下面介绍这些协议栈的实现部分。

函数netp\_ip\_output\_student的功能是构造并发送一个IP数据包，其高层协议为自定义协议类型，负载内容为自定义负载。这个函数的编码工作需要由学生完成。

当有数据到达本机网络接口时，函数netp\_ip\_input\_student将被调用，并传递给这个函数原始数据。在该函数中，需要判断一些条件值来确定接收到的数据包为自定义IP数据，如果是自定义IP数据包，则输出负载内容，如果不是，则返回NETP\_PUSH\_TO\_LWIP交给协议栈继续处理。

netproto\_iproute\_student.h和netproto\_iproute\_shudent.c文件用于实现IP路由表的管理。其中netproto\_iproute\_student.h文件的内容与netproto\_ip\_student.h相似，只是添加了netp\_create\_iproute\_table()和display\_iproute\_table()函数的声明。

netproto\_iproute\_shudent.c文件是协议栈中IP路由表的管理的实现部分，其中定义了1个全局数组和5个函数。

全局数组iproute\_table用来存储IP路由表，数组元素为netp\_iproute\_table\_item结构对象，netp\_iproute\_table\_item结构的定义如下：

structnetp\_iproute\_table\_item{  
structip\_addrdest\_network;  
structip\_addrnetmask;  
intinterface;  
intflag;  
};

函数display\_iproute\_table的功能是显示IP路由表的内容，该函数功能已经完成，学生不需要修改。

函数netp\_create\_iproute\_table完成IP路由表的创建，该函数需要学生完成，实现IP路由表的创建。

函数query\_iproute完成IP路由表的查询功能，该函数需要学生完成，实现IP路由表的查询。

函数netp\_iproute\_output\_student通过查询IP路由表完成IP数据包的发送功能。该函数功能已经完成，学生不需要修改。

函数netp\_iproute\_input\_student处理输入数据包，显示负载内容，该函数功能已经完成，学生不需要修改。

在编码过程中可能会设计到一些结构体、宏和函数，下表是对他们进行和介绍：

表3-3实验涉及的结构体和函数

| 结构体/函数 | 声明或定义 | 描述 |
| --- | --- | --- |
| structnetp\_ip\_header | structnetp\_ip\_header{  u8\_theaderlen:4;  u8\_tversion:4;  u8\_tdiff\_services;  u16\_ttotal\_length;  u16\_tidentification;  u16\_tflags\_offset;  u8\_ttime\_to\_live;  u8\_tprotocol;  u16\_theader\_checksum;  structip\_addrsource\_address;  structip\_addrdestination\_address;  }; | ipv4包头结构 |
| structin\_addr | structin\_addr{  u32\_ts\_addr;  }; | 32位地址 |
| structnetp\_eth\_header | structnetp\_eth\_header{  u8\_tdest\_address[ETH\_ADDRESS\_LEN];  u8\_tsour\_address[ETH\_ADDRESS\_LEN];  u16\_ttype;  }; | 以太网帧头结构 |
| IP\_HEADER\_LEN | #defineIP\_HEADER\_LEN20 | IP包头长度 |
| PAYLOAD\_DATA | #definePAYLOAD\_DATA"Hello,World!" | IP负载内容 |
| PAYLOAD\_LEN | #definePAYLOAD\_LENsizeof(PAYLOAD\_DATA) | IP负载长度 |
| ETH\_HEADER\_LEN | #defineETH\_HEADER\_LEN14 | 以太网帧头长度 |
| netp\_current\_ip\_addr | u32\_tnetp\_current\_ip\_addr(); | 获取当前正在使用的网络接口的IP地址 |
| htons | u16\_thtons(u16\_tn); | 将16位数值由主机字节序转换为网络字节序 |
| inet\_addr | u32\_tinet\_addr(constchar\*cp); | 将ASCII编码的Internet地址转换成为网络字节序地址 |
| inet\_chksum | u16\_tinet\_chksum(void\*dataptr,u16\_tlen) | 计算校验和 |
| netp\_ip\_output | err\_tnetp\_ip\_output(  void\*buffer,intbuff\_len,  intadapterID  ) | 发送IP数据包，需要构造IP头部 |

### 十二.各模块推荐流程

#### 1.IP数据包发送流程

编码实现IP数据包发送推荐使用如下流程：

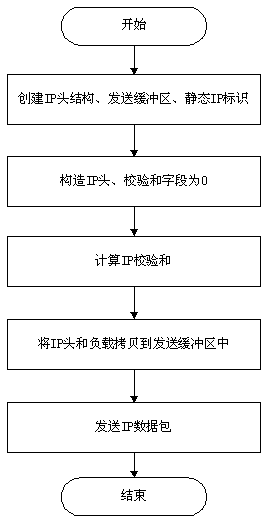


图3-20IP数据包发送推荐流程

#### 2.输入IP数据包处理流程

编码实现处理IP输入数据包推荐使用如下流程：

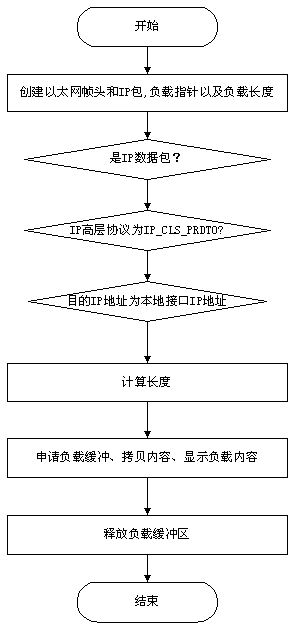


图3-21处理IP输入数据包推荐流程

#### 3.IP路由表创建流程

编码实现IP路由表创建推荐使用如下流程：

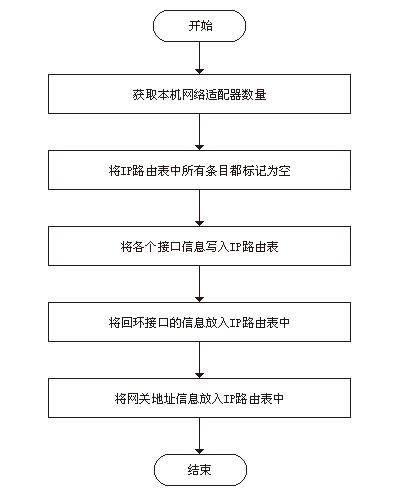


图3-22IP路由表创建推荐流程

#### 4.IP路由表查询流程

编码实现IP路由表查询推荐使用如下流程：

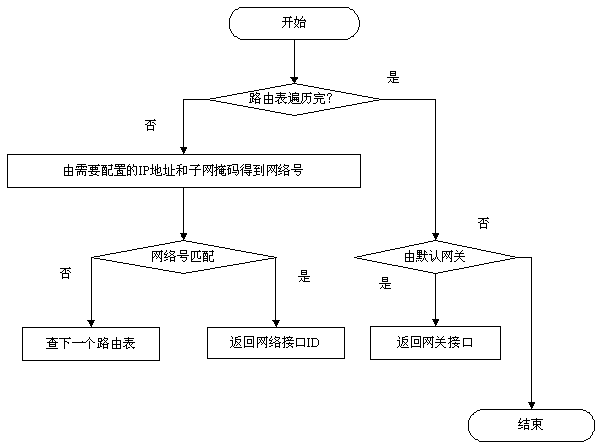


图3-23IP路由表查询推荐流程

【实验步骤】

### 练习1编辑并发送IP数据报

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.主机B在命令行方式下输入staticroute\_config命令，开启静态路由服务。

2.主机A启动协议编辑器，编辑一个IP数据报，其中：

MAC层：

目的MAC地址：主机B的MAC地址（对应于172.16.1.1接口的MAC）。

源MAC地址：主机A的MAC地址。

协议类型或数据长度：0800。

IP层：

总长度：IP层长度。

生存时间：128。

源IP地址：主机A的IP地址（172.16.1.2）。

目的IP地址：主机E的IP地址（172.16.0.2）。

校验和：在其它所有字段填充完毕后计算并填充。

自定义字段：

数据：填入大于1字节的用户数据。

【说明】先使用协议编辑器的“手动计算”校验和，再使用协议编辑器的“自动计算”校验和，将两次计算结果相比较，若结果不一致，则重新计算。

●IP在计算校验和时包括哪些内容？

3.在主机B（两块网卡分别打开两个捕获窗口）、E上启动协议分析器，设置过滤条件（提取IP协议），开始捕获数据。

4.主机A发送第1步中编辑好的报文。

5.主机B、E停止捕获数据，在捕获到的数据中查找主机A所发送的数据报，并回答以下问题：

●第1步中主机A所编辑的报文，经过主机B到达主机E后，报文数据是否发生变化？若发生变化，记录变化的字段，并简述发生变化的原因。

6.将第1步中主机A所编辑的报文的“生存时间”设置为1，重新计算校验和。

7.主机B、E重新开始捕获数据。

8.主机A发送第5步中编辑好的报文。

9.主机B、E停止捕获数据，在捕获到的数据中查找主机A所发送的数据报，并回答以下问题：

主机B、E是否能捕获到主机A所发送的报文？简述产生这种现象的原因。

思考问题：

1.说明IP地址与硬件地址的区别，为什么要使用这两种不同的地址？

### 练习2特殊的IP地址

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.受限广播地址

(1)主机A编辑一个IP数据报，其中：

目的MAC地址：FFFFFF-FFFFFF。

源MAC地址：A的MAC地址。

源IP地址：A的IP地址。

目的IP地址：255.255.255.255。

自定义字段数据：填入大于1字节的用户数据。

校验和：在其它字段填充完毕后，计算并填充。

(2)主机B、C、D、E、F重新启动协议分析器并设置过滤条件（提取IP协议，捕获172.16.1.2接收和发送的所有IP数据包，设置地址过滤条件如下：172.16.1.2<->Any）。

(3)主机B、C、D、E、F重新开始捕获数据。

(4)主机A发送这个数据报。

(5)主机B、C、D、E、F停止捕获数据。

●记录实验结果

表3-5实验结果

|  |  |
| --- | --- |
|  | 主机号 |
| 收到主机A发送的IP数据报 |  |
| 未收到主机A发送的IP数据报 |  |

●结合实验结果，简述受限广播地址的作用。

2.环回地址

(1)主机F重新启动协议分析器开始捕获数据并设置过滤条件（提取IP协议）。

(2)主机Eping127.0.0.1。

(3)主机F停止捕获数据。

●主机F是否收到主机E发送的目的地址为127.0.0.1的IP数据报？为什么？

思考问题：

1.受限广播地址的作用范围？

2.受限广播地址和直接广播地址的区别？

3.路由器转发受限广播吗？

### 练习3IP数据报分片

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.主机B设置以太网端口的MTU为800字节（两个端口都设置），执行命令如下（进入cmd控制台）：

# 查看以太网端口当前MTU值

netsh interface ipv4 show subinterfaces

# 修改指定以太网端口的MTU值，按实际情况指定网络连接名称

netsh interface ipv4 set subinterface "*网络连接名称*" mtu=800 store=persistent

2.主机A、B、E启动协议分析器，打开捕获窗口进行数据捕获并设置过滤条件(提取ICMP协议)。

3.在主机A上，执行命令ping-l1000172.16.0.2。

4.主机A、B、E停止捕获数据。在主机E上重新定义过滤条件（取一个ICMP数据包，按照其IP层的标识字段设置过滤），如图所示：

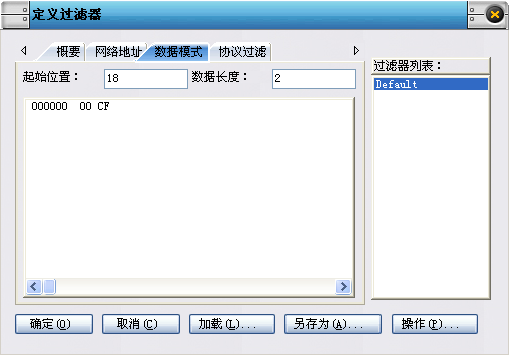


图3-24过滤条件设置

●将ICMP报文分片信息填入下表，分析表格内容，理解分片的过程。

表3-6实验结果

| 字段名称 | 分片序号1 | 分片序号2 | 分片序号3 |
| --- | --- | --- | --- |
| “标识”字段值 |  |  |  |
| “还有分片”字段值 |  |  |  |
| “分片偏移量”字段值 |  |  |  |
| 传输的数据量 |  |  |  |

5.主机E恢复默认过滤器。主机A、B、E重新开始捕获数据。

6.在主机A上，执行命令ping-l2000172.16.0.2。

7.主机A、B、E停止捕获数据。察看主机A、E捕获到的数据，比较两者的差异，体会两次分片过程。

8. 主机B恢复以太网端口的MTU为1500。

思考问题：

1.Ping的数据部分为3000字节，回显请求报文为何被分为3片而不是2片？

2.数据部分长度为多少时报文正好被分为2片？

3.不同协议的MTU的范围从296到65535，使用大的MTU有什么好处？使用小的MTU有什么好处？

### 练习4子网掩码的作用

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.所有主机取消网关。

2.主机A、C、E设置子网掩码为255.255.255.192，主机B（172.16.1.1）、D、F设置子网掩码为255.255.255.224。

3.主机Aping主机B（172.16.1.1），主机Cping主机D（172.16.1.4），主机Eping主机F（172.16.0.3）。

●记录实验结果

表3-7实验结果

|  |  |
| --- | --- |
|  | 是否ping通 |
| 主机A----主机B |  |
| 主机C----主机D |  |
| 主机E----主机F |  |

●请问什么情况下两主机的子网掩码不同，却可以相互通信？

4.主机B在命令行方式下输入recover\_config命令，停止静态路由服务。

5.所有主机恢复到网络结构二的配置。

思考问题：

1.IP数据报中的首部校验和并不检验数据报中的数据，这样做的最大好处是什么？缺点是什么？

### 练习5IP数据报的接收与发送

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。实验开始前，先单击“初始环境”。

在实验中，主机A将新接口的IP地址设置为172.16.1.12、主机B使用物理接口1，将新接口的IP设置为172.16.1.11、主机C将新接口的IP地址设置为172.16.1.13、主机D使用处于连接状态的物理接口，将新接口的IP地址设置为172.16.1.14、主机E使用处于连接状态的物理接口，将新接口的IP地址设置为172.16.0.12、主机F将新接口的IP地址设置为172.16.0.13。所有主机使用子网掩码255.255.255.0，默认网关设置为0.0.0.0。

1. 所有主机编码实现发送IP数据包

（1） 各主机使用VS2005软件打开实验平台安装目录下的JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_ip\_student\netproto\_ip\_student.sln文件，在源函数netproto\_ip\_student.c的函数netp\_ip\_output\_student内编写实现代码。

注：若实验平台安装在C盘，则目录为：C:\Program Files\JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_ip\_student\netproto\_ip\_student.sln。

(2) 参考实验原理IP数据包发送推荐流程图给出的流程，分析已经存在的代码。

已经存在的代码创建了IP头结构、发送缓冲区和静态IP标识，将IP头和负载拷贝到发送缓冲区中以及将缓冲区数据发送到网络中的实现。

(3) 所有主机设置目的IP地址

各主机使用VS2005软件打开平台安装目录下的JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_ip\_student\netproto\_ip\_student.sln文件，在头文件的netproto\_ip\_student.h内将IP\_DEST\_ADDR宏定义的IP地址设置为同组主机新接口的IP地址。负载内容自定义，也可以使用默认值。

(4) 构造IP头，校验和字段为0

构造并填充一个IP数据包头。可以使用netproto\_ip\_student.h文件中所定义的值。其中，总长度应该为IP\_HEADER\_LEN+PAYLOAD\_LEN，标识为ip\_id并将ip\_id加1，IP高层协议为IP\_CUS\_PROTO宏所定义的，校验和为0，源IP地址为本接口的IP地址，目的IP地址为IP\_DEST\_ADDR所定义的IP地址。

(5) 计算IP校验和

使用inet\_chksum函数计算IP校验和，IP校验和只校验IP头部分。

2.所有主机编码实现IP数据包输入处理功能

(1) 各主机使用VS2005软件打开实验平台安装目录下的JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_ip\_student\netproto\_ip\_student.sln文件，在源函数netproto\_ip\_student.c的函数netp\_ip\_input\_student内编写实现代码。

(2) 参考实验原理处理IP输入数据包推荐流程图给出的流程，分析已经存在的代码。

已经存在的代码定义了负载缓冲区指针payload\_buff、负载长度payload\_len、以太网头结构eth\_header和IP头结构ip\_header以及负载内容显示到标准输出的实现。

(3) 提取IP数据包

各主机通过以太网头结构eth\_header中的type即“协议类型或数据长度”字段值判断该数据帧的高层协议是否为IP协议，可以使用宏MAC\_PROTO\_IP。如果不是IP数据包则返回NETP\_PUSH\_TO\_LWIP交给协议栈继续执行。

(4) 提取IP高层协议

各主机通过IP头结构ip\_header中的protocol即“高层协议类型”字段值判断该IP数据包高层是否为自定义协议。如果不是则返回NETP\_PUSH\_TO\_LWIP交给协议栈继续执行。

(5) 提取目的IP地址为本接口IP的数据包

各主机通过IP头结构ip\_header中的destination\_address即“目的IP地址”字段值判断该IP数据包是否是发送给本接口的。如果不是则返回NETP\_PUSH\_TO\_LWIP交给协议栈继续执行。

(6) 计算负载长度

各主机通过“总长度字段”值和“首部长度”字段值来计算IP数据包负载的长度，将长度结果保存在payload\_len变量中。

(7) 将负载拷贝到负载缓冲区

各主机编码将负载拷贝到负载缓冲区中，用于打印到标准输出设备。

3. 所有主机打开协议分析器，开始捕获数据

4. 所有主机调试并运行程序

5. 各主机停止数据捕获，观察实验现象

● 你收到的负载内容是什么？

6.参考代码如下：

/\*\*  
\*\brief编辑并发送一个IP数据包  
\*  
\*由学生完成这个函数，发送一个IP数据包。主线程将会调用这个函数。  
\*/  
void  
netp\_ip\_output\_student()  
{  
//创建IP头结构、发送缓冲区、静态IP标识  
structnetp\_ip\_headerip\_header;  
staticu16\_tip\_id=0;  
u8\_tsend\_buff[IP\_HEADER\_LEN+PAYLOAD\_LEN];  
  
//构造IP头，校验和字段为0  
ip\_header.version=IP\_VERSION;  
ip\_header.headerlen=IP\_HEADERLEN;  
ip\_header.diff\_services=IP\_DS;  
ip\_header.total\_length=htons(IP\_HEADER\_LEN+PAYLOAD\_LEN);  
ip\_header.identification=htons(ip\_id++);  
ip\_header.flags\_offset=0;  
ip\_header.time\_to\_live=IP\_TTL;  
ip\_header.protocol=IP\_CUS\_PROTO;  
ip\_header.header\_checksum=0;  
ip\_header.source\_address.addr=netp\_current\_ip\_addr();  
ip\_header.destination\_address.addr=inet\_addr(IP\_DEST\_ADDR);  
  
//计算IP校验和  
ip\_header.header\_checksum=inet\_chksum(&ip\_header,IP\_HEADER\_LEN);  
  
//将IP头和负载拷贝到发送缓冲区中  
memcpy(send\_buff,&ip\_header,IP\_HEADER\_LEN);  
memcpy(send\_buff+IP\_HEADER\_LEN,PAYLOAD\_DATA,PAYLOAD\_LEN);  
  
//发送IP数据包  
if(ERR\_OK==netp\_ip\_output(send\_buff,IP\_HEADER\_LEN+PAYLOAD\_LEN,0)){  
printf("\n发送IP数据包----成功！\n");  
}else{  
printf("\n发送IP数据包----失败！\n");  
}  
}  
/\*\*  
\*\brief当有数据帧到达时，将调用这个函数  
\*  
\*\parampacket指向接收到的数据  
\*\parampacket\_len数据帧的长度  
\*  
\*\return一个put\_to\_lwip类型的返回值。  
\*返回NETP\_PUT\_LWIP表示处理数据后将数据帧交给上层协议栈继续处理  
\*返回NETP\_NO\_PUT\_LIWP表示不将数据帧交给上层协议栈处理。  
\*/  
enumpush\_to\_lwip  
netp\_ip\_input\_student(void\*packet,intpacket\_len)  
{  
//创建以太网帧头和IP包头，负载指针以及负载长度  
char\*payload\_buff;//负载缓冲区指针  
intpayload\_len;//负载长度  
structnetp\_eth\_headereth\_header={0};  
structnetp\_ip\_headerip\_header;  
  
memcpy(&eth\_header,packet,sizeof(eth\_header));  
  
//过滤非IP数据包  
if(eth\_header.type!=htons(MAC\_PROTO\_IP)){  
returnNETP\_PUSH\_TO\_LWIP;  
}  
  
memcpy(&ip\_header,(u8\_t\*)packet+ETH\_HEADER\_LEN,IP\_HEADER\_LEN);  
  
//过滤IP高层协议非IP\_CUS\_PROTO数据包  
if(ip\_header.protocol!=IP\_CUS\_PROTO){  
returnNETP\_PUSH\_TO\_LWIP;  
}  
  
//过滤目的IP地址非本接口IP的数据包  
if(ip\_header.destination\_address.addr!=netp\_current\_ip\_addr()){  
returnNETP\_PUSH\_TO\_LWIP;  
}  
  
//计算负载长度  
payload\_len=(ip\_header.total\_length-ip\_header.headerlen\*4)\*sizeof(u8\_t);  
  
//申请负载缓冲区  
payload\_buff=malloc(payload\_len);  
memset(payload\_buff,0,payload\_len);  
  
//拷贝内容  
memcpy(payload\_buff,(u8\_t\*)packet+ETH\_HEADER\_LEN+IP\_HEADER\_LEN,  
payload\_len-1);  
  
//显示负载内容  
printf("接收到由%s发送的自定义IP数据包，负载为：%s\n",  
inet\_ntoa(\*(structin\_addr\*)(&ip\_header.source\_address)),  
payload\_buff);  
  
//释放负载缓冲区  
free(payload\_buff);  
  
returnNETP\_NO\_PUSH\_LIWP;  
}

### 练习6IP路由表管理的设计与实现

本练习将主机A、B、C、D、E、F作为一组进行实验。实验开始前，先单击“初始环境”。

在实验中，主机A将新接口的IP地址设置为172.16.1.12、主机B将与物理接口1相关联的新接口的IP设置为172.16.1.11、主机B将与物理接口2相关联的新接口的IP设置为172.16.0.11、主机C将新接口的IP地址设置为172.16.1.13、主机D使用处于连接状态的物理接口，将新接口的IP地址设置为172.16.1.14、主机E使用处于连接状态的物理接口，将新接口的IP地址设置为172.16.0.12、主机F将新接口的IP地址设置为172.16.0.13。所有主机使用子网掩码255.255.255.0。主机A、B、C、D的默认网关设置为172.16.1.1，主机E、F的默认网关设置为172.16.0.1。

1. 所有主机编码构造IP路由表

(1) 各主机使用VS2005软件打开实验平台安装目录下的JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_ip\_student\netproto\_ip\_student.sln文件，在源函数netproto\_iproute\_student.c的函数netp\_create\_iproute\_table内编写实现代码。在netp\_create\_iproute\_table函数中，已经存在的代码将IP路由表中所有表项的值都设置为空。

注：若实验平台安装在C盘，则目录为：C:\Program Files\JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_ip\_student\netproto\_ip\_student.sln。

(2) 获取本机网络适配器数量

各主机使用netp\_get\_adapter\_num获取本机网络适配器数量，并保存在adapter\_num变量中。

(3) 将各个接口信息写入IP路由表

利用在步骤（2）中获得的网络适配器数量，将所有接口的信息写入IP路由表中，这些信息包括：目的网络号、子网掩码号、接口号以及路由表标志。

(4) 将回环接口的信息放入IP路由表中

利用网络接口ID：NETP\_LOOP\_IF\_ID将回环接口的信息写入路由表中。

(5) 将网关地址信息放入IP路由表中

利用netp\_get\_gw函数将默认网关信息写入路由表。

2. 所有主机编码查询ip路由表

(1) 各主机使用VS2005软件打开实验平台安装目录下的JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_ip\_student\netproto\_ip\_student.sln文件，在源函数netproto\_iprout\_student.c的函数query\_iproute内编写实现代码。在query\_iproute函数中，已经定义了记录默认网关接口的变量gw\_interface。

(2) 遍历IP路由表，查找相匹配的接口并且记录默认网关所在接口

各主机遍历IP路由表，IP路由表的长度为NETP\_IPROUTE\_TABLE\_SIZE。利用待查询的IP地址，路由表中的目的网络号，路由表中的子网掩码号确定网络接口号。

3. 所有主机打开协议分析器，开始捕获数据

4. 所有主机调试并运行程序

(1) 主机B依次向其它主机发送信息，注意使用的网络接口号。

5. 各主机停止数据捕获，观察实验现象

6.参考代码如下：

/\*\*  
\*\brief构造IP路由表  
\*/  
void  
netp\_create\_iproute\_table()  
{  
inti;  
  
intadapter\_num;  
  
//获取本机网络适配器数量  
adapter\_num=netp\_get\_adapter\_num();  
  
//将IP路由表中所有条目都标记为空  
for(i=0;i<NETP\_IPROUTE\_TABLE\_SIZE;++i){  
iproute\_table[i].flag=NETP\_IPROUTE\_FLAG\_EMPTY;  
}  
  
//将各个接口信息写入IP路由表  
for(i=0;i<adapter\_num;++i){  
intadapterID=i+1;  
iproute\_table[i].dest\_network=netp\_get\_ip\_addr(adapterID);  
iproute\_table[i].netmask=netp\_get\_mask(adapterID);  
iproute\_table[i].dest\_network.addr&=iproute\_table[i].netmask.addr;  
iproute\_table[i].interface=adapterID;  
iproute\_table[i].flag=NETP\_IPROUTE\_FLAG\_DESTNET;  
}  
  
//将回环接口的信息放入IP路由表中  
iproute\_table[i].dest\_network=netp\_get\_ip\_addr(NETP\_LOOP\_IF\_ID);  
iproute\_table[i].netmask=netp\_get\_mask(NETP\_LOOP\_IF\_ID);  
iproute\_table[i].dest\_network.addr&=iproute\_table[i].netmask.addr;  
iproute\_table[i].interface=NETP\_LOOP\_IF\_ID;  
iproute\_table[i].flag=NETP\_IPROUTE\_FLAG\_LOOP;  
  
//将网关地址信息放入IP路由表中  
++i;  
iproute\_table[i].dest\_network=netp\_get\_gw(netp\_default\_adapter\_ID());  
iproute\_table[i].interface=netp\_default\_adapter\_ID();  
iproute\_table[i].flag=NETP\_IPROUTE\_FLAG\_GW;  
}  
  
/\*\*  
\*\brief查询ip路由表  
\*  
\*\paramipaddr需要匹配的IP地址  
\*  
\*\return返回相匹配的网络接口  
\*如果没有接口可以发送目的ip地址为ipaddr的数据包，返回默认网关所在的接口。  
\*如果没有默认网关可以使用，则返回-1代表查询失败  
\*/  
intquery\_iproute(structip\_addr\*ipaddr)  
{  
inti;  
//记录默认网关接口的变量  
intgw\_interface=0;  
  
//遍历IP路由表，查找相匹配的接口并且记录默认网关所在接口  
for(i=0;i<NETP\_IPROUTE\_TABLE\_SIZE;++i){  
if(iproute\_table[i].flag!=NETP\_IPROUTE\_FLAG\_EMPTY){  
if(iproute\_table[i].dest\_network.addr==  
(ipaddr->addr&iproute\_table[i].netmask.addr)){  
returniproute\_table[i].interface;  
}  
if(iproute\_table[i].flag==NETP\_IPROUTE\_FLAG\_GW){  
gw\_interface=iproute\_table[i].interface;  
}  
}  
}  
  
//没有相匹配的接口，如果有默认网关则返回网关所在接口，否则返回-1  
if(gw\_interface){  
returngw\_interface;  
}else{  
return-1;  
}  
}

## 实验4Internet组控报文协议（ICMP）

【实验目的】

1.掌握ICMP协议的报文格式

2.理解不同类型ICMP报文的具体意义

3.了解常见的网络故障

【学时分配】

2学时

【实验环境】

该实验采用网络结构二

【实验原理】

### 一.ICMP简介

IP协议是一种不可靠无连接的协议，当数据包经过多个网络传输后，可能出现错误、目的主机不响应、包拥塞和包丢失等问题。为了处理这些问题，在IP层引入了另一个协议ICMP（Internet控制信息协议）。ICMP报文有两种类型：差错报文和查询报文。ICMP报文封装在IP报文里传输。ICMP报文可以被IP协议、传输层协议（TCP或UDP）和用户进程使用。ICMP与IP一样，都是不可靠传输，ICMP的信息也可能丢失。为了防止ICMP报文无限制的连续发送，对于ICMP报文在传输中发生的问题，将不再发送ICMP差错报文。

### 二.ICMP报文格式

ICMP数据包由8字节的首部和可变长度的数据部分组成。如下图所示，第一个字段是ICMP的类型，它定义了报文类型。第二个字段是代码字段，它指明了发送这个特定报文类型的原因。校验和字段为ICMP数据包提供差错校验。对于不同类型的ICMP数据包，首部的最后4个字节的格式是不同的，具体的格式将在下面讨论。

差错报文的数据部分携带引起差错的原始数据。查询报文的数据部分携带了基于查询类型的额外信息。

|  |  |  |
| --- | --- | --- |
| 类型（8位） | 代码（8位） | 校验和（16位） |
| 首部的其余部分 | | |
| 数据部分 | | |

图4-1ICMP报文的一般格式

●类型：8位字段，用于描述特定类型的ICMP报文。

●代码：8位字段，进一步描述某些ICMP报文的具体说明。

●校验和：16位字段，覆盖这个ICMP报文的校验和。

### 三.ICMP封装

ICMP报文封装在IP数据报中，具体的封装方法如下图所示：

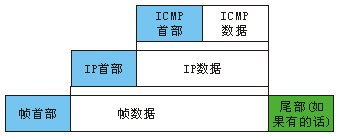


图4-2ICMP封装

### 四.ICMP报文类型

ICMP报文可分为两大类：差错报文和查询报文，如下图所示：

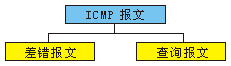


图4-3ICMP报文类型

差错报文报告路由器或主机在处理IP数据报时遇到的问题。

查询报文是成对出现的，它帮助主机或网络管理员从一个路由器或另一个主机得到特定的信息。例如，主机使用ICMP回显请求和回显应答报文发现它们的邻站。下表列出了每一类ICMP报文。

表4-1ICMP报文

|  |  |  |
| --- | --- | --- |
| 种类 | 类型 | 报文 |
| 差错报文 | 3 | 目的端不可达 |
| 4 | 源点抑制 |
| 11 | 超时 |
| 12 | 参数问题 |
| 5 | 改变路由 |
| 查询报文 | 8或0 | 回显请求或应答 |
| 13或14 | 时间戳请求或应答 |
| 17或18 | 地址掩码请求或应答 |
| 10或9 | 路由器询问和通告 |

### 五.ICMP查询报文

ICMP查询报文能够获得特定主机或路由器的信息，能够对某些网络问题进行诊断。ICMP查询报文包括4对不同类型的报文，分别为回显请求和应答报文、时间戳请求和应答报文、地址掩码请求和应答报文以及路由器询问和通告报文，如下图所示。

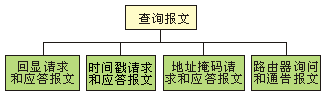


图4-4查询报文

#### 1.回显请求和应答

回显请求报文和回显应答报文用来确定了两个节点（主机或路由器）之间是否能够正常通信。用户可以使用这对报文来发现网络问题。

主机或路由器可以发送回显请求报文给另一个主机或路由器。收到回显请求报文的主机或路由器产生回显应答报文，并将其返回给原来的发送端。

回显请求和回显应答报文可用来确定是否在IP这级能够通信。因为ICMP报文被封装在IP数据报中，发送回显请求的主机在收到回显应答报文时，就证明了在发送端和接收端之间能够使用IP数据报进行通信。另外，这还证明了链路中的路由器能够接收、处理和转发数据报。

回显请求和回显应答报文还可以用于检查另一个主机是否可达。用户可以调用数据包因特网搜寻器（ping）命令做到这一点。现在，大多数系统都提供ping命令，它可以产生一连串的回显请求或回显应答报文。

回显请求和回显应答报文也可以用来验证某个节点是否正常工作。可以向被测试的节点发送回显请求报文，该报文的数据字段包含一段信息，如果这段信息被测试的节点在回显应答报文中完全一样地重复，则说明该节点工作正常；否则说明该节点出了问题。下图给出了回显请求和回显应答报文的格式。类型值为8表示回显请求报文，类型值为0表示回显应答报文。标识符和序号字段在协议中没有正式定义，可以由发送端任意使用。

|  |  |  |
| --- | --- | --- |
| 类型：8或0 | 代码：0 | 校验和 |
| 标识符 | | 序号 |
| 回请求报文发送  由应答报文重复 | | |

图4-5ICMP回显请求和应答报文

#### 2.时间戳请求和应答

两个机器（主机或路由器）可使用时间戳请求和时间戳应答报文来确定IP数据报在这两个机器之间传输所需要的时间，也可以用作两个机器时钟的同步。这两个报文的格式如下图所示。其中，类型值为13表示时间戳请求报文，类型值为14表示时间戳应答报文。

|  |  |  |
| --- | --- | --- |
| 类型：13或14 | 代码：0 | 校验和 |
| 标识符 | | 序号 |
| 原始时间戳 | | |
| 接收时间戳 | | |
| 发送时间戳 | | |

图4-6时间戳请求和应答报文

在报文格式中3个时间戳字段的长度都是32位。每一个字段都保存一个整数，代表从通用时间（格林尼治标准时间）的午夜起测量出的时间，以毫秒为单位。

源节点在时间戳请求报文的原始时间戳字段填入它的时钟所显示的通用时间。其它两个时间戳字段都填入零。

收到时间戳请求报文后，终点将生成时间戳应答报文。终点把请求报文中的原始时间戳字段值复制到应答报文的同一个字段中。然后在接收时间戳字段中填入收到这个请求报文时其时钟所显示的通用时间。最后，终点在应答报文将要发送时在发送时间戳字段中填入其时钟所显示的通用时间。

时间戳请求和时间戳应答报文可以用来计算数据报从源点到终点所需的时间，还可以用于计算数据报再返回到源点所需的时间。

#### 3.地址掩码请求和应答

要得到掩码，主机应该向局域网上的路由器发送地址掩码请求报文。若主机知道路由器的地址，它就把请求直接发送给该路由器。若主机不知道路由器的地址，则它就广播地址掩码请求报文。路由器收到地址掩码请求报文后，用地址掩码应答报文进行响应，向主机提供所需的掩码。

地址掩码请求和地址掩码应答报文的格式如下图所示。其中，类型值为17表示地址掩码请求报文，类型值为18表示地址掩码应答报文。在请求报文中，地址掩码字段填入全0。当路由器把地址掩码应答发回给主机时，这个字段就包含真正的掩码。

|  |  |  |
| --- | --- | --- |
| 类型：17或18 | 代码：0 | 校验和 |
| 标识符 | | 序号 |
| 地址掩码 | | |

图4-7地址掩码请求和应答报文

无盘工作站在启动时是需要地址掩码的，它会使用RARP协议查找完整的IP地址，在收到IP地址以后，无盘工作站就可使用地址掩码请求报文找出地址掩码，从而确定IP地址的哪一部分定义了子网号，哪一部分定义了主机号。

#### 4.路由器询问和通告

主机若想把数据发送给另一个网络上的主机，就需要知道连接到该网络上的路由器的地址。此外，这个主机还需要知道这些路由器是否正常工作。路由器询问报文和路由器通告报文可以完成这项工作。主机可把路由器询问报文进行广播（或多播）。收到询问报文的一个或几个路由器就使用路由器通告报文广播其路由选择信息。甚至在没有主机询问时，路由器也可周期性地发送路由器通告报文。路由器发送出通告报文时，它不仅通告了自己的存在，而且也通告了它所知道的所有在这个网络上的路由器。下图给出了路由器询问报文的格式。

|  |  |  |
| --- | --- | --- |
| 类型：10 | 代码：0 | 校验和 |
| 标识符 | | 序号 |

图4-8路由器询问报文

下图给出了路由器通告报文的格式。生存期字段表示这个报文在多长时间内是有效的。在通告报文中每一个路由器的项目有两个字段：路由器地址和地址优先级。地址优先级定义了路由器的等级。优先级用来选择某个路由器作为默认路由器。若地址优先级为零，则这个路由器就被认为是默认路由器。若地址优先级是0x80000000，则这个路由器永远不能被选为默认路由器。

|  |  |  |
| --- | --- | --- |
| 类型：9 | 代码：0 | 校验和 |
| 地址数 | 地址项目长度 | 生存期 |
| 路由器地址1 | | |
| 地址优先级1 | | |
| 路由器地址2 | | |
| 地址优先级2 | | |
| …… | | |

图4-9路由器通告报文

### 六.ICMP差错报文

ICMP差错报文用来报告差错。虽然现代的技术已经制造出很可靠的传输媒体，但差错仍然存在，因而必须进行处理。正如在实验三中所讨论的，IP是个不可靠的协议。这就表示IP不考虑差错校验和差错控制。ICMP就是为了补偿这个缺点而设计的。然而ICMP不能纠正差错，它只是报告差错，差错纠正留给高层协议去做。ICMP使用源IP地址把差错报文发送给数据报的源点（发出者）。

一共有5种差错报文：目的端不可达、源点抑制、超时、参数问题以及改变路由，如下图所示。

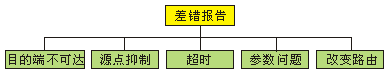


图4-10差错报文

差错报文的数据字段包括原始数据报（引起差错的报文）的首部和原始数据报数据部分的前8个字节。包括原始数据报首部的目的是为了向差错报文的原始信源给出关于数据报本身的信息。包括数据的前8个字节是因为这前8个字节提供了关于端口号（UDP和TCP）和序号（TCP）的信息。根据这些信息，源点可以把差错情况通知给上层协议。

#### 1.目的端不可达

当路由器不能够为数据报找到路由或主机，就丢弃这个数据报，然后向发出这个数据报的源主机发送目的端不可达报文。下图给出了目的端不可达报文的格式。这种类型的代码字段指明了丢弃该数据报的原因。

|  |  |  |
| --- | --- | --- |
| 类型：3 | 代码：0至15 | 校验和 |
| 未使用（全0） | | |
| 收到的IP数据报的一部分，包括IP首部以及数据报数据的前8个字节 | | |

图4-11目的端不可达报文

#### 2.源点抑制

IP协议是无连接协议，因此通信缺乏流量控制。ICMP源点抑制报文就是为了给IP增加一种流量控制而设计的。当路由器或主机因拥塞而丢弃数据报时，它就向数据报的发送端发送源点抑制报文。第一，它通知发送端，数据报已被丢弃。第二，它警告发送端，在路径中的某处出现了拥塞，因而源端必须放慢发送过程。源点抑制报文的格式如下图所示：

|  |  |  |
| --- | --- | --- |
| 类型：4 | 代码：0 | 校验和 |
| 未使用（全0） | | |
| 收到的IP数据报的一部分，包括IP首部以及数据报数据的前8个字节 | | |

图4-12源点抑制报文

#### 3.超时

超时报文是在以下两种情况下产生的：

●数据报的生存时间字段值被减为0时，路由器丢弃这个数据报，并向发送端发送超时报文。

●当组成报文的所有分段未能在某一时限内到达目的主机时，也要产生超时报文。当第一个分段到达时，目的主机就启动计时器。当计时器的时限到了，目的主机就将所有分段丢弃，并向发送端发送超时报文。超时报文格式如下图所示：

|  |  |  |
| --- | --- | --- |
| 类型：11 | 代码：0或1 | 校验和 |
| 未使用（全0） | | |
| 收到的IP数据报的一部分，包括IP首部以及数据报数据的前8个字节 | | |

图4-13超时报文

#### 4.参数问题

当数据报在Internet上传送时，如果路由器或目的主机发现数据报首部中出现了二义性问题，或在数据报的某个字段中缺少某个值，它就丢弃这个数据报，并向发送端发送参数问题报文。下图给出了参数问题报文格式。代码字段指明了丢弃数据报的原因。

|  |  |  |
| --- | --- | --- |
| 类型：12 | 代码：0或1 | 校验和 |
| 指针 | 未使用（全0） | |
| 收到的IP数据报的一部分，包括IP首部以及数据报数据的前8个字节 | | |

图4-14参数问题报文

●代码为0时表示在首部的某个字段中有差错或二义性。指针字段值指向有问题的字节。

●代码为1时表示缺少所需的选项部分。这种情况下不使用指针。

#### 5.重定向

为了提高效率，主机不参与路由选择更新过程，因此，主机可能会把某数据报发送给一个错误的路由器。这时，收到这个数据报的路由器会把数据转发给正确的路由器，同时向主机发送重定向报文，告诉主机正确路由器的地址。下图给出了重定向报文的格式。

|  |  |  |
| --- | --- | --- |
| 类型：5 | 代码：0到3 | 校验和 |
| 目标路由器IP地址 | | |
| 收到的IP数据报的一部分，包括IP首部以及数据报数据的前8个字节 | | |

图4-15改变路由报文

### 七.ICMP校验和

ICMP的校验和的计算覆盖了整个ICMP报文（首部和数据）。

#### 1.校验和的计算

发送端按以下步骤使用反码算术运算计算校验和：

(1)把校验和字段置为零。

(2)把报文按照16位长度分段，使用反码算术运算计算所有分段之和。

(3)把得到的和求反码，得到校验和。

(4)把校验和存储在校验和字段中。

#### 2.校验和的测试

接收端按以下步骤使用反码算术运算来测试校验和的正确性：

(1)把报文按照16位长度分段，使用反码算术运算计算所有分段之和。

(2)把得到的和求反码。

(3)若结果是全0，则接受这个报文；否则就拒绝这个报文。

【实验步骤】

### 练习1运行Ping命令

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A、B、C、D、E、F作为一组进行实验。

实验开始前主机B首先执行命令“staticroute\_config”启动静态路由。

1.主机B、E、F启动协议分析器，打开捕获窗口进行数据捕获并设置过滤条件（提取ICMP协议）。

2.主机Aping主机E（172.16.0.2）。

主机Cping主机F（172.16.0.3）。

3.主机B、E、F停止捕获数据，察看捕获到的数据，并回答以下问题：

●捕获的报文对应的“类型”和“代码”字段分别是什么？

●分析报文中的哪些字段保证了回显请求报文和回显应答报文的一一对应？

### 练习2ICMP查询报文

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.主机A启动协议编辑器，编辑一个ICMP时间戳请求数据帧发送给主机C（172.16.1.3）。

MAC层：

目的MAC地址：C的MAC地址。

源MAC地址：A的MAC地址。

协议类型或数据长度：0800。

IP层：

总长度：包含IP层和ICMP层长度。

高层协议类型：1。

校验和：在其它字段填充完毕后计算并填充。

源IP地址：A的IP地址。

目的IP地址：C的IP地址。

ICMP层：

类型：13。

代码字段：0。

校验和：在ICMP层其它字段填充完毕后，计算并填充。

其它字段使用默认值。

2.主机C启动协议分析器进行数据捕获，并设置过滤条件（提取ICMP协议）。

3.主机A发送已编辑好的数据帧。

4.主机C停止捕获数据。察看主机C捕获到的数据，并填写下表：

表4-2实验结果

|  |  |  |  |
| --- | --- | --- | --- |
| 时间戳请求报文 | | 时间戳应答报文 | |
| ICMP字段名 | 字段值 | ICMP字段名 | 字段值 |
| 类型 |  | 类型 |  |
| 标识号 |  | 标识号 |  |
| 序列号 |  | 序列号 |  |
| 发起时间戳 |  | 发起时间戳 |  |
| 接收时间戳 |  | 接收时间戳 |  |
| 传送时间戳 |  | 传送时间戳 |  |

思考问题：

1.能否根据时间戳计算出当前的时间？

2.使用时间戳得到的时间比从系统得到的时间有什么好处？

### 练习3ICMP差错报文

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.目的端不可达

(1)主机A、B、C、D、E、F启动协议分析器捕获数据，并设置过滤条件（提取ICMP）。

(2)主机A、C、D、E、Fping172.16.2.10（不存在的IP地址）。

(3)主机A、B、C、D、E、F停止捕获数据。察看捕获到的数据，并回答以下问题：

●捕获到的是哪一种目的端不可达报文？

2.超时

(1)主机A、C、D分别启动协议编辑器，编写一个发送给主机F（172.16.0.3）的ICMP数据帧。其中：

MAC层：

目的MAC地址：主机B的MAC地址（172.16.1.1接口的MAC）。

源MAC地址：本机的MAC地址。

协议类型或数据长度：0800。

IP层：

总长度：包含IP层和ICMP层长度。

生存时间（TTL）：0。

高层协议类型：1。

校验和：在其它字段填充完毕后，计算并填充。

源IP地址：本机的IP地址。

目的IP地址：F的IP地址。

ICMP层：

类型：8。

代码字段：0。

校验和：在ICMP其它字段填充完毕后，计算并填充。

其它字段使用默认值。

(2)主机E、F分别启动协议编辑器，编写一个发送给主机D（172.16.1.4）的ICMP数据帧。其中：

MAC层：

目的MAC地址：主机B的MAC地址（172.16.0.1接口的MAC）。

源MAC地址：本机的MAC地址。

协议类型或数据长度：0800。

IP层：

总长度：包含IP层和ICMP层长度。

TTL：0。

高层协议类型：1。

校验和：在其它字段填充完毕后，计算并填充。

源IP地址：本机的IP地址。

目的IP地址：D的IP地址。

ICMP层：

类型：8。

代码字段：0。

校验和：在ICMP其它字段填充完毕后，计算并填充。

其它字段使用默认值。

(3)主机B启动协议分析器，网卡b1(172.16.1.1)、网卡b2（172.16.0.1）分别捕获数据，并设置过滤条件（提取ICMP协议）。

(4)主机A、C、D、E、F各自发送已编辑好的数据帧。

(5)主机B停止捕获数据，察看并分析捕获到的数据。

(6)主机B在命令行方式下输入recover\_config命令，停止静态路由服务。

思考问题：

1.为什么要设置TTL字段？

2.为什么要限制由失效的ICMP差错报文再产生一个ICMP报文？

3.什么样的ICMP报文是由路由器发送出的？什么样的ICMP报文是由目的主机发送出的？

4.主机A向主机B发送数据报，主机B从未收到该数据报，而主机A也从未收到出问题的通知。试给出可能发生情况的两种不同解释。

## 实验5Internet组管理协议（IGMP）

【实验目的】

1.掌握IGMP协议的报文格式

2.掌握IGMP协议的工作原理

3.理解多播组地址到以太网地址的映射

【学时分配】

4学时

【实验环境】

该实验采用网络结构二

【实验原理】

### 一.单播、多播与广播

#### 1.单播

在单播通信中，源点和终点是一对一的。IP数据报中的源IP地址和目的IP地址分别代表了发送端主机和目的端主机。如下图所示，单播数据包从源点S1出发，经过一些路由器后，到达终点D1。



图5-1单播

路由器收到数据包后，会根据路由表的路由信息，选择一个最佳接口转发这个数据包。如果路由器根据路由表找不到合适的路由，那么就可以丢弃这个数据包。

#### 2.多播

在多播通信中，源点和终点是一对多的关系。在这种类型的通信中，源地址是单播地址，而目的地址是组地址（D类）。组地址定义这个组的成员。下图给出了多播的概念。多播数据包从源点S1出发，然后到达属于组G1的所有终点。

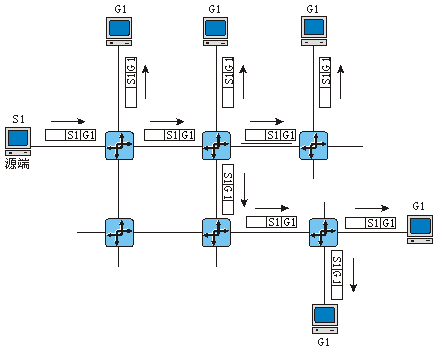


图5-2多播

一个多播数据包从源点出发到达属于组G1的所有终点，路由器收到数据包后可以从它的多个接口转发这个数据包。

目前，多播路由机制还没有被广泛支持，主要的多播路由机制有距离向量多播路由协议（DVMRP）和协议无关多播（PIM），有关这两种机制，请参考相关资料。

#### 3.广播

在广播通信中，源点和终点是一对多的关系。源点只有一个，但所有其它的主机都是终点。Internet明确地不支持广播，因为这会产生非常巨大的通信量，同时它所需的带宽也是非常巨大的。如果有一个节点要向连接在Internet上的每一个节点发送消息，那么可以想象这将产生多大的通信量。

### 二.组播的地址

多播使用组地址，IANA（国际因特网地址分配委员会）把D类地址空间用于IP组播地址。所以，IP组播地址的范围是：224.0.0.0—239.255.255.255。其中：

表5-1组播地址

|  |  |  |  |
| --- | --- | --- | --- |
| 224.0.0.0 | 基地址（保留） | 224.0.1.15 | IETF-2-VIDEO |
| 224.0.0.1 | 在本子网上的所有参加多播的主机和路由器 | 224.0.1.16 | MUSIC-SERVICE |
| 224.0.0.2 | 在本子网上的所有参加多播的路由器 | 224.0.1.17 | SEANET-TELEMETRY |
| 224.0.0.3 | 未指派 | 224.0.1.18 | SEANET-IMAGE |
| 224.0.0.4 | DVMRP路由器 | 224.0.1.19 | MLOADD |
| 224.0.0.5 | 用来给一个网段中所有的OSPF路由器发送OSPF路由信息 | 224.0.1.20 | 私有实验使用 |
| 224.0.0.6 | 用来给一个网段中指定的OSPF路由器发送OSPF路由信息 | 224.0.1.21 | DVMRPonMOSPF |
| 224.0.0.7 | ST路由器 | 224.0.1.22 | SVRLOC |
| 224.0.0.8 | ST主机 | 224.0.1.23 | XINGTV |
| 224.0.0.9 | RIP2组地址 | 224.0.1.24 | WINS服务器组地址； |
| 224.0.0.10 | 供IGRP路由器使用 | 224.0.1.25 | nbc-pro |
| 224.0.0.11 | 供DHCP服务器和中继代理使用 | 224.0.1.26 | nbc-pfn |
| 224.0.0.22 | IGMPv3成员关系报告 | 224.0.1.27-224.0.1.255 | 未指派 |
| 224.0.1.0 | VMTP管理组 | 224.0.2.1 | "rwho"Group(BSD)(unofficial) |
| 224.0.1.1 | NTP网络时间协议 | 224.0.2.2 | SUNRPCPMAPPROC\_CALLIT |
| 224.0.1.2 | SGI-Dogfight | 224.0.3.0-224.0.3.255 | RFE常用服务 |
| 224.0.1.3 | Rwhod | 224.0.4.0-224.0.4.255 | RFE个人会议 |
| 224.0.1.4 | VNP | 224.0.5.0-224.0.5.127 | CDPD组 |
| 224.0.1.5 | ArtificialHorizons | 224.0.5.128-224.0.5.255 | 未指派 |
| 224.0.1.6 | NSS名字服务服务器 | 224.0.6.0-224.0.6.127 | CornellISISProject |
| 224.0.1.7 | AUDIONEWS音频新闻广播 | 224.0.6.128-224.0.6.255 | 未指派 |
| 224.0.1.8 | SUNNIS+InformationService | 224.1.0.0-224.1.255.255 | ST多播组 |
| 224.0.1.9 | MTP多播传输协议 | 224.2.0.0-224.2.255.255 | 多媒体会议调用 |
| 224.0.1.10 | IETF-1-LOW-AUDIO | 224.252.0.0-224.255.255.255 | DIS临时组 |
| 224.0.1.11 | IETF-1-音频 | 232.0.0.0-232.255.255.255 | VMTP临时组 |
| 224.0.1.12 | IETF-1-视频 | 239.192.0.0-239.251.255.255 | 限制在一个组织的范围 |
| 224.0.1.13 | IETF-2-LOW-AUDIO | 239.252.0.0-239.255.255.255 | 限制在一个地点的范围 |
| 224.0.1.14 | IETF-2-AUDIO |  |  |

动态的组成员：多播组中的成员是动态的。一个进程可请求其主机参加某个特定的组，或在任意时间退出该组。

使用硬件进行多播：当数据报传送到以太网时，以太网就利用硬件进行多播，交付给属于该组成员的主机。

#### 1.多播组地址到以太网地址的转换

下图是D类IP地址与以太网多播地址的映射关系。

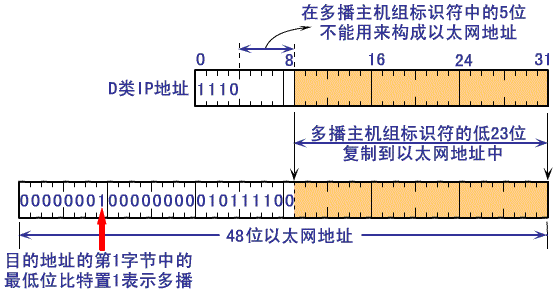


图5-3多播组地址到以太网地址的转换

在多播中，任何一个以太网地址的首字节必须是01（如图5-3），以太网地址的前25位是0000000100000000010111100这意味着与IP多播相对应的以太网地址范围从01005E000000到01005E7FFFFF。然后将多播IP地址的低23位映射到以太网地址中的低23位。

由于多播IP地址中的最高5位在映射过程中被忽略，因此每个以太网多播地址对应的多播组是不惟一的。32个不同的多播IP地址被映射为一个以太网地址。既然地址映射不是惟一的，那么设备驱动程序或IP层就必须对数据报进行过滤。因为网卡可能接收到主机不想接收的多播数据帧。另外，如果网卡不提供足够的多播数据帧过滤功能，设备驱动程序就必须接收所有多播数据帧，然后对它们进行过滤。

### 三.IGMP协议简介

随着Internet的飞速发展，利用Internet进行多媒体传播的应用研究越来越多。对某些应用而言，点对点交换信息不管对网络还是对信息发送者，都是一种负担。因此，需要一种办法让本身规模较大而相对互连网又较小的工作组能相互方便、快捷地传递信息。这就引起了组播（Multicase）技术的发展。

IGMP协议（Internet组管理协议）是“InternetGroupManagementProtocol”的缩写，它位于网络层。IGMP协议运行于主机和与主机直接相连的路由器之间，是IP主机用来报告多播组成员身份的协议。通过IGMP协议，一方面可以使主机通知本地路由器希望加入并接收某个特定组播组的信息；另一方面，路由器可以周期性地查询局域网内某个已知组的成员是否处于活动状态。

IGMP使用预留的组播组地址224.0.0.1与本地路由器通信。

### 四.IGMP报文格式

IGMP协议目前有三个版本IGMP、IGMPv2和IGMPv3。其中IGMPv2有3种报文类型：查询报文，成员关系报告报文和退出报告报文。查询报文共有两种：一般的和特殊的。如下图所示：

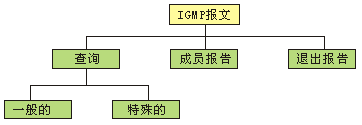


图5-4IGMPv2的报文类型

IGMPv3有2种报文类型：查询报文和成员关系报告报文。

本书中将讨论IGMP版本2和IGMP版本3的相关知识，由于IGMP版本1现在已经很少使用了，所以本书不讨论IGMP版本1。

#### 1.IGMPv2报文的格式

下图给出了IGMP（版本2）报文的格式。

|  |  |  |
| --- | --- | --- |
| 类型（8位） | 最大响应时间（8位） | 校验和（16位） |
| 在成员报告和退出报告中的组地址，在一般的查询报文中是全0 | | |

图5-5IGMP的报文格式

●类型：该字段定义了IGMP报文类型。表5-2给出了类型和对应的值。

表5-2IGMP报文类型

|  |  |
| --- | --- |
| 类型 | 值 |
| 查询（一般或特殊） | 0x11 |
| 成员关系报告 | 0x16 |
| 退出报告 | 0x17 |

●最大响应时间：该字段定义了查询必须在多长时间内回答。

●校验和：该字段包含覆盖整个IGMP报文的校验和。

●组地址：该字段在一般的查询报文中值是0。在特殊的查询报文、成员关系报告报文以及退出报告报文中值为多播组地址。

#### 2.IGMPv3查询报文的格式

IGMPv3查询分为三类：

普通查询：普通查询由多播路由器发出，用于获知邻接接口(即查询所传输的网络中所相连的接口)的完整的多播接收状态。在一个普通查询中，组地址字段和源数量(N)字段都为0。

指定组查询：由一台多播路由器发出，用于获知邻接接口中跟某一个IP地址相关的多播接收状态。在指定组查询中，组地址字段含有需要查询的那个组地址，源数量(N)字段为0。

指定组和源查询：由一台多播路由器发出，用于获知邻接接口是否需要接收来自指定的这些源的，发往指定组的多播数据报。在一个指定组和源的查询中，组地址字段含有要查询的多播地址,源地址[i]字段含有相关的源地址。

下图给出了IGMP（版本3）查询报文的格式。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 类型（8位） | | 最大响应代码（8位） | 校验和（16位） | |
| 组地址（32位） | | | | |
| 保留（4位） | S（1位） | 查询者健壮变量（8位） | 查询者查询间隔代码（8位） | 源数量(N) |
| 源地址[1] | | | | |
| …… | | | | |
| 源地址[N] | | | | |

图5-6IGMP查询报文的报文格式

IGMPv3查询报文各字段的含义如下：

●类型：该字段定义了IGMP报文类型，查询报文该字段的值为0x11。

●最大响应代码：该指定在发送一个响应报告之前所允许的最大时间。

●校验和：该字段包含覆盖整个IGMP报文的校验和。

●组地址：当发送一个普通查询的时候，组地址字段必须被置0。当发送一个指定组查询或者发送一个指定组和源的查询时，必须被设置成要被查询的IP组地址。

●保留：该字段为全0，在接收时被忽略。

●S标志：S标志用来指定禁止路由器处理（SuppressRouter-SideProcessing）。当被设置为1时，表示任何路由器都不可以更新它们在收到查询时要更新的那些定时器。

●查询者健壮变量：如果不为0，查询者健壮变量中包含一个被查询者使用的健壮变量的值，如果查询者的健壮变量的值超过7，即该字段的最大值，那么查询者健壮变量被设成0。路由器取最近收到的查询中的查询者健壮变量值作为它们自己的健壮性变量的值。如果收到的查询者健壮变量全是0，那么接收者就使用缺省的健壮性变量值，或者是一个静态配置的值。

●查询者查询间隔代码：该字段指定查询者使用的查询间隔，以秒为单位。

●源数量：该字段表明该查询中存在多少个源地址。

●源地址：源地址[i]是n个IP单播地址的数组，n就是源数量字段的值。

#### 3.IGMPv3成员关系报告报文的格式

下图给出了IGMP（版本3）成员关系报告报文的格式。

|  |  |  |
| --- | --- | --- |
| 类型（8位） | 保留（8位） | 校验和（16位） |
| 保留（13位） | | 组记录数(N) |
| 组记录[1] | | |
| …… | | |
| 组记录[N] | | |

图5-7IGMP成员关系报告报文的报文格式

IGMP成员关系报告报文各字段的含义如下：

●类型：成员关系报告报文中该字段的值为0x22。

●保留：该字段为全0，在接收时被忽略。

●校验和：该字段包含覆盖整个IGMP报文的校验和。

●组记录数：该字段指定在报告存在多少个组记录。

●组记录：每一个组记录字段都是一整块数据，其中含有发送者在报告发送接口上的某一个多播组的成员关系。

#### 4.IGMPv3成员关系报告报文中的组记录

在IGMPv3成员关系报告报文中，组记录是一整块数据，它的格式如下图所示：

|  |  |  |
| --- | --- | --- |
| 记录类型（8位） | 辅助数据长度（8位） | 源数量(N) |
| 组地址（32位） | | |
| 源地址[1] | | |
| … | | |
| 源地址[N] | | |
| 辅助数据 | | |

图5-8组记录格式

组记录格式中，各个字段的含义如下：

●记录类型：当前定义了6种记录类型，如下表：

表5-3记录类型

| 值 | 含义 |
| --- | --- |
| 1 | MODE\_IS\_INCLUDE-标明接口相关于某一指定多播地址的过滤模式为INCLUDE。该组记录中的源地址[i]字段含有该接口的相关于该多播地址的源列表(如果非空的话)。 |
| 2 | MODE\_IS\_EXCLUDE-标明接口相关于某一指定多播地址的过滤模式为EXCLUDE。该组记录中的源地址[i]字段含有该接口的相关于该多播地址的源列表(如果非空的话)。 |
| 3 | CHANGE\_TO\_INCLUDE\_MODE，标明接口相关于某一指定的多播地址的过滤模式改变到INCLUDE。该组记录中的源地址[i]字段含有该指定多播地址相关的新的源列表(如果非空的话)。 |
| 4 | CHANGE\_TO\_EXCLUDE\_MODE，标明接口相关于某一指定的多播地址的过滤模式改变到EXCLUDE。该组记录中的源地址[i]字段含有该指定多播地址相关的新的源列表(如果非空的话)。 |
| 5 | ALLOW\_NEW\_SOURCE，标明组记录中的源地址[i]字段含有系统希望接收的发往某一多播地址的，新的源的列表。如果这是对一个INCLUDE列表的改变，那么这些地址会被添加到列表中，如果这是对一个EXCLUDE列表的改变，那么这些地址会被从列表中删除。 |
| 6 | BLOCK\_OLD\_SOURCE，标明组记录中的源地址[i]字段含有系统不希望再接收的发往某一多播地址的源的列表。如果这是对一个INCLUDE列表的改变，那么这些地址会被从列表中删除，如果这是对一个EXCLUDE列表的改变，那么这些地址会被添加到列表中。 |

●辅助数据长度：该字段含有在组记录中的辅助数据的实际长度，其单位是32位。它有可能是0，这就表示辅助数据不存在。

●源数量：该字段标明在组记录中存在多少源地址。

●组地址：该字段标明该组记录从属的多播IP地址。

●源地址：该字段是一个数组，含有n个单播地址。n就是该记录的源数量(N)字段的值。

●辅助数据：该字段如果存在，它含有关于该组记录的一些附加信息。在IGMPv3中，没有定义任何辅助数据。所以，IGMPv3的实现在任何传输的组记录中都不应该含有任何辅助数据（即必须把辅助数据长度字段置0）。

### 五.IGMP封装

IGMP报文被封装在IP数据报中，如下图所示：

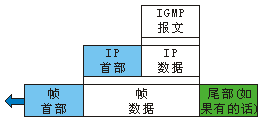


图5-9IGMP数据包的封装

在IGMP数据包封装过程中，IP层有3个字段需要注意，它们是高层协议类型字段、生存时间字段和目的IP地址字段。

#### 1.高层协议类型字段

对于IGMP协议，IP的高层协议类型字段值是2。高层协议类型字段值为2的所有IP数据包，其数据部分都交付给IGMP协议处理。

#### 2.生存时间字段

当IGMP报文封装成IP数据报时，生存时间的值必须是1。因为IGMP的作用范围是局域网，IGMP报文不能够发送到局域网以外的地方。生存时间值为1保证了这个报文不离开这个局域网，因为到了下一个路由器这个数值就减小到零，因而这个数据包要被丢弃。

#### 3.目的IP地址

下表给出了每一种类型报文的目的IP地址。

表5-4各种类型的目的IP地址

|  |  |
| --- | --- |
| 类型 | 目的IP地址 |
| 查询 | 224.0.0.1在这个子网上的所有节点 |
| 成员关系报告 | 224.0.0.22 |

### 六.多播组成员的加入与退出

#### 1.加入一个组

一个主机或路由器能够加入一个组。每一个主机要维护一个有组内成员关系的进程表。当一个进程要加入到一个新组时，它就向主机发送请求。这个主机就在它的表中增加这个进程的名字和所请求的组的名字。如果所请求的组在表中不存在，这个主机就发送成员关系报告报文。如果所请求的组在表中已经存在，那就没有必要发送成员关系报告报文，因为这个主机已经是这个组的成员了，它已经开始接收发送给这个组的所有数据。

路由器也维护一个多播地址表，这个表给出了连接到每一个接口的成员关系。如果对于这些接口中的一个组有兴趣时，路由器就发送成员关系报告报文。这里的路由器就和主机一样，但它的组表的范围更大，因为它记录了连接在它的接口上的所有忠实成员。

下图给出了主机或路由器发送的成员关系报告。

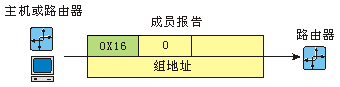


图5-10成员关系报告

IGMP协议要求把成员关系报告报文在很短的时间内发送两次。这样，如果第一个报文丢失或受到损伤，那么第二个报文可以代替它。

#### 2.退出一个组

在IGMPv2中，当主机发现在一个特定组中已经没有感兴趣的进程，它就发送退出报告报文。同样地，当路由器发现在一个特定组中已经没有连接在它的接口上感兴趣的网络，它就发送退出这个组的报告。

但是，当多播路由器收到退出报告时，它并不能立即从它的表中清除这个组，因为这个退出报告仅仅是从一个主机或路由器发送来的，可能还有其它的主机或路由器仍然对这个组感兴趣。这时，这个路由器要发送一个特殊的查询报文，在这个报文中使用这个组的多播地址，并允许任何主机或路由器在指定的响应时间内回答。如果在这段时间内没有收到这个组的成员关系报告报文，这个路由器就认为在这个网络上没有忠实成员，因而就从表中清除这个组。

在IGMPv1和IGMPv3中，并没有退出报告报文，当使用IGMPv1和IGMPv3的主机要退出多播组时，它不发送任何报文。路由器会定期的发送查询报文，如果一段时间内路由器没有收到成员关系报告报文，那么路由器就清除这个组。

下图给出了IGMPv2的退出一个组的机制。

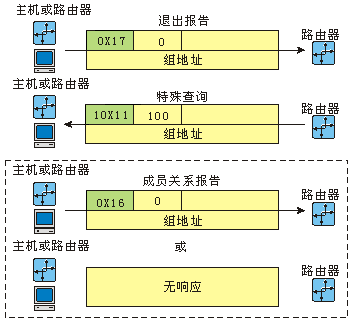


图5-11退出报告

### 七.监视成员关系

当仅有一台主机留在一个组中时，如果这台主机离开了这个组而没有发送退出报告报文，那么多播路由器将永远收不到退出报告报文，它认为这个组一直存在这个忠诚成员。为了解决这个问题，多播路由器应该负责监视局域网上的所有主机或路由器，以便知道它们是否愿意继续留在多播组中。

路由器周期性地（默认的间隔是每隔125s）发送一般查询报文。在这个报文中，组地址被置为0.0.0.0。

路由器期望着在它的组表中的每一个组的回答，甚至新的组也可以回答。查询报文的最大响应时间是10秒。当主机路由器收到一般查询报文时，如果它对一个组感兴趣，就发送成员关系报告报文。

#### 1.延迟响应

为了避免不必要的通信量，IGMP使用延迟响应策略。当主机或路由器收到查询报文时，它不是立即响应，而是延迟响应。主机或路由器使用一个随机数产生计时器，它的截止时间是1～10秒。截止时间可以使用1秒或更短的步长。对表中的每一个组都设置计时器。例如，对第一组的计时器可以设截止时间为2秒，但对第三组的计时器则可设截止时间为5秒。每一个主机或路由器在发送成员关系报告报文之前都要等待它的计时器的时间。在这段等待时间中，如果在同一组中的其它主机或路由器的计时器先到了截止时间，那么这个主机或路由器就发送成员关系报告报文。因为这个报文是广播发送的，等待的主机或路由器都能收到这个报文，并知道没有必要对这个组发送重复的报文；因此，等待的主机或路由器就取消发送成员关系报告。

【实验步骤】

### 练习1观察IGMP报文

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.主机B启动协议分析器，网卡b1（172.16.1.1）和网卡b2（172.16.0.1）分别捕获数据，并设置过滤条件（提取IGMP）。

2.在主机B上启动IGMP协议：

(1)在主机B的命令行下使用“igmp\_config”命令启动IGMP协议。

(2)在主机B的命令行下使用“igmp\_config"172.16.1.1的接口名"route”命令将172.16.1.1接口设置为“IGMP路由器”。

(3)在主机B的命令行下使用“igmp\_config"172.16.0.1的接口名"route”命令将172.16.0.1接口设置为“IGMP路由器”。

3.观察主机B的协议分析器所采集到的数据。

●找到“成员关系查询”报文，并填写下表：

表5-5实验结果

|  | 数据内容 | 含义 |
| --- | --- | --- |
| 目的MAC地址 |  |  |
| 目的IP地址 |  |  |
| 生存时间 |  |  |
| 组地址 |  |  |

●通过目的MAC地址和目的IP地址，简述组播IP地址到MAC地址的映射方式。

思考问题：

1.一个组的多播地址是231.24.60.9。当局域网在使用TCP/IP时，其48位的以太网地址是什么？

### 练习2利用IGMP加入一个多播组

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.在主机A、B、C、D、E、F上启动协议分析器捕获数据，并设置过滤条件（提取IGMP）。

2.在主机A、C、D、E上启动“组播工具”（方法：实验平台工具栏中的组播工具），并加入多播组（使用224.0.1.88作为多播地址）。

3.在主机A、B、C、D、E、F上观察协议分析器上采集到的数据。

4.理解“组播工具”使用IGMP协议加入一个多播组的过程。

5.在主机A、C、D、E上点击“离开组播”退出多播组。

### 练习3多播通信

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.在主机B、F上启动协议分析器捕获数据，并设置过滤条件（提取IGMP和UDP）。

2.在主机A、C、E上启动“组播工具”，并加入到同一个多播组（如：224.0.1.88）。

●察看主机B上的组表信息（在命令行方式下，输入“igmp\_configshowgrouptable”），记录其中条目。

3.主机A发送数据，同时观察主机C、E上“组播工具”接收到的数据。

4.在主机E上点击“离开组播”退出多播组。

(1)等待一段时间（约5分钟）后察看主机B上的组表信息。

●在命令行方式下，输入“igmp\_configshowgrouptable”。

●察看主机B上的“路由和远程访问/IGMP/显示组表格”。

(2)其中是否含有主机E所加入的组的相关条目？为什么？

5.在主机A、C上点击“离开组播”退出多播组。

(1)等待一段时间（约5分钟）后察看主机B上的组表信息。

●在命令行方式下，输入“igmp\_configshowgrouptable”。

●察看主机B上的“路由和远程访问/IGMP/显示组表格”。

(2)其中是否含有主机A、C所加入的组的相关条目？为什么？

6.主机B、F停止捕获数据，观察协议分析器所捕获的数据。

●简述IGMP在多播通信过程中所起到的作用，绘制多播组成员和IGMP路由器的报文交互过程（包括IGMP和UDP）。

7.主机B在命令行下输入recover\_config命令，停止IGMP协议。

思考问题：

1.为什么没有必要让IGMP报文在它自己的网络以外传送？

2.若一主机愿意在5个组中继续其成员关系，它应当发送5个不同的成员关系报告报文还是只发送一个？

## 实验6用户数据报协议（UDP）

【实验目的】

1.掌握UDP协议的报文格式

2.掌握UDP协议校验和的计算方法

3.理解UDP协议的优缺点

4.理解协议栈对UDP协议的处理方法

5.理解UDP上层接口应满足的条件

【学时分配】

4学时

【实验环境】

该实验采用网络结构一

【实验原理】

### 一.进程到进程的通信

在学习UDP协议之前，首先应该了解主机到主机的通信和进程到进程的通信，以及这两种通信之间的区别。

IP协议负责主机到主机的通信。作为一个网络层协议，IP协议只能把报文交付给目的主机。这是一种不完整的交付，因为这个报文还没有送交到正确的进程。像UDP这样的传输层协议负责进程到进程的通信。UDP协议负责把报文交付到正确的进程。下图描绘了IP协议和UDP协议的作用范围。

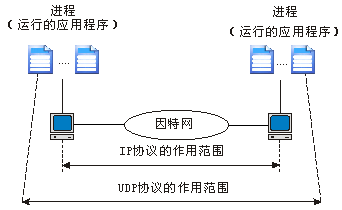


图6-1UDP与IP的区别

#### 1.端口号

在网络中，主机是用IP地址来标识的。而要标识主机中的进程，就需要第二个标识符，这就是端口号。在TCP/IP协议族中，端口号是在0～65535之间的整数。

在客户/服务器模型中，客户程序使用端口号标识自己，这种端口号叫做短暂端口号，短暂的意思是生存时间比较短。一般把短暂端口取为大于1023的数，这样可以保证客户程序工作得比较正常。

服务器进程也必须用一个端口号标识自己。但是这个端口号不能随机选取。如果服务器随机选取端口号，那么客户端在想连接到这个服务器并使用其服务的时候就会因为不知道这个端口号而无法连接。TCP/IP协议族采用熟知端口号的办法解决这个问题。每一个客户进程都必须知道相应的服务器进程熟知端口号。

UDP的熟知端口号如下表所示：

表6-1UDP的熟知端口号

|  |  |  |
| --- | --- | --- |
| 端口 | 协议 | 说明 |
| 7 | Echo | 把收到的数据报回送到发送端 |
| 9 | Discard | 丢弃收到的任何数据报 |
| 11 | Users | 活跃的用户 |
| 13 | Daytime | 返回日期和时间 |
| 17 | Quote | 返回日期和引用（译者注：可参阅RFC856） |
| 19 | Chargen | 返回字符串 |
| 53 | Nameserver | 域名服务 |
| 67 | Bootps | 下载引导程序信息的服务器端口 |
| 68 | Bootpc | 下载引导程序信息的客户端口 |
| 69 | TFTP | 简单文件传送协议 |
| 111 | RPC | 远程过程调用 |
| 123 | NTP | 网络时间协议 |
| 161 | SNMP | 简单网络管理协议 |
| 162 | SNMP | 简单网络管理协议（陷阱） |
| 520 | RIP | 路由信息协议 |

在一个IP数据包中，目的IP地址和端口号起着不同的寻址作用。目的IP地址定义了在世界范围内惟一的一台主机。当主机被选定后，端口号定义了在这台主机上运行的多个进程中的一个。

#### 2.套接字地址

一个IP地址与一个端口号结合起来就叫做一个套接字地址。客户套接字地址惟一地定义了客户进程，而服务器套接字地址惟一地定义了服务器进程。

要使用UDP的服务，就需要一对套接字地址：客户套接字地址和服务器套接字地址。客户套接字地址指定了客户端的IP地址和客户进程，服务器套接字地址指定了服务器的IP地址和服务器进程。

### 二.面向连接的服务与面向无连接的服务

从通信的角度来看，在OSI参考模型中，下层能向上层提供两种不同形式的服务：面向连接的服务和面向无连接的服务。

#### 1.面向连接的服务

所谓连接，就是两个对等实体为进行数据通信而进行的一种结合。面向连接服务在进行数据交换前，先建立连接。当数据传输结束后，应释放这个连接。因此，采用面向连接的服务进行数据传送要经历三个阶段：

(1)建立连接阶段：在有关的服务原语以及协议数据单元中，必须给出源用户和目的用户的完整地址。同时可以协商服务质量和其它一些选项。

(2)数据交换阶段：在这个阶段，每个报文中不必包含完整的源用户和目的用户的完整地址，而是使用一个连接标识符来代替。由于连接标识符相对于地址信息要短得多，因此使控制信息在报文中所占的比重相对减小，从而可减小系统的额外开销，提高信道的有效利用率。另外，报文的发送和接收都是按固定顺序的，即发送方先发送的报文，在接受方先收到。

(3)释放连接阶段：通过相应的服务原语完成释放操作。

从面向连接服务的三个阶段来看，连接就像一个管道，发送端在其一端依次发送报文，接收者依次在其另一端按同样的顺序接收报文。这种连接又称虚拟电路。它可以避免报文的丢失、重复和乱序。

若两个用户经常需要通信，则可以建立永久虚电路。这样可以免除每次通信时建立连接和释放连接这两个阶段。这点与电话网中的专线很相似。

#### 2.面向无连接的服务

在面向无连接服务的情况下，两个实体之间的通信不必事先建立一个连接。相对于面向连接的服务，面向无连接服务灵活方便且快速。但它不能防止报文的丢失、重复和乱序。由于它的每个报文必须包括完整的源地址的目的地址，因此开销较大。

面向无连接服务主要有三种类型：

(1)数据报：它的特点是发完报文就结束，而对方不做任何响应。数据报的服务简单，额外开销少，但可靠性差，它比较适合于数据具有很大的冗余度以及要求有较高的实时性的通信场合。

(2)证实交付：又称可靠的数据报。这种服务对每一个报文产生一个证实给发送方，不过这种证实不是来自对应方用户，而是来自提供服务的层。这种证实只能保证报文已经发给目的站了，而不能保证对应方用户正确地接收到报文。

(3)请求回答：这种类型服务是接收端用户每收到一个报文，即向发送端用户发送一个应答报文。但是双方发送的报文都有可能丢失。如果接收端发现报文有错误，则回送一个表示有错误的报文。

### 三.UDP协议简介

UDP（用户数据报协议），主要用来支持那些需要在计算机之间传输数据的网络应用。包括网络视频会议系统在内的众多的客户/服务器模式的网络应用都需要使用UDP协议。UDP协议从问世至今已经被使用了很多年，虽然其最初的光彩已经被一些类似的协议所掩盖，但是即使是在今天，UDP仍然不失为一项非常实用和可行的网络传输层协议。

UDP协议直接位于IP协议的上层。根据OSI参考模型，UDP和TCP都属于传输层协议。UDP协议不提供端到端的确认和重传功能，它不保证数据包一定能到达目的地，因此是不可靠协议。

UDP协议有以下特点：

●UDP是面向事务的协议，它用最少的传输量为应用程序向其它程序发送报文提供了一个途径。

●UDP是无连接的、不可靠的传输机制。在发送数据报前，UDP在发送和接收两者之间不建立连接。

●UDP让应用程序能直接访问网络层的数据报服务，例如分段和重组等网络层所提供的数据报服务。

●UDP使用IP协议作为数据传输机制的底层协议。

●UDP报头和数据都以与最初传输时相同的形式被传送到最终目的地。

●UDP不提供确认，也不对数据的到达顺序加以控制。因此UDP报文可能会丢失。

●不实现数据包的传送和重复检测。

●当数据包在传送过程中发生错误时，UDP不能报告错误。

●吞吐量不受拥塞控制算法的调节，只受应用程序生成数据的速率、传输带宽、发送端和接收端主机性能的限制。

### 四.UDP报文格式

下图显示了UDP报文格式。每个UDP报文称为一个用户数据报（UserDatagram），用户数据报分为两个部分：UDP首部和UDP数据。首部被分为四个16位的字段，分别代表源端口号﹑目的端口号﹑报文的长度以及UDP校验和。

|  |  |
| --- | --- |
| 源端口（16位） | 目的端口（16位） |
| 有效负载长度（16位） | 校验和（16位） |
| 数据 | |
| …… | |

图6-2UDP报文格式

●源端口：该字段表示发送端的端口号。如果源端口没有使用，那么此字段的值就被指定为0。这是一个可选的字段。不同的应用程序使用不同的端口号，UDP协议使用端口号为不同的应用程序保留其各自的数据传输通道，从而实现了同一时间段内多个应用程序可以一起使用网络进行数据的发送和接收。

●目的端口：该字段表示数据包被发往的目的端的端口号。

●有效负载长度：该字段表示包括UDP首部和UDP数据在内的整个用户数据报的长度。该字段的最小值是8。数据报的最大尺寸随操作系统的不同而不同。在两字节字段中，理论上数据报最多可达65535字节。然而，一些UDP实现将数据报的大小限制到了8192字节。

●校验和：UDP的校验的校验范围包括伪首部（IP首部一部分字段）、UDP首部和UDP数据，该字段是可选的。如果该字段值为零就说明不进行校验。

### 五.UDP封装

当进程有报文要通过UDP发送时，它就把这个报文连同一对套接字地址以及数据的长度传递给UDP。UDP收到数据后就加上UDP首部。然后UDP就把这用户数据报连同套接字地址一起传递给IP。IP加上自己的首部，在高层协议类型字段使用值17，指出该数据是从UDP协议来的。这个IP数据报再传递给数据链路层。数据链路层接收到IP数据报后，加上自己的首部（可能还有尾部），再传给物理层。物理层把这些位编码为电信号或光信号，把它发送到远程的主机。如下图所示：

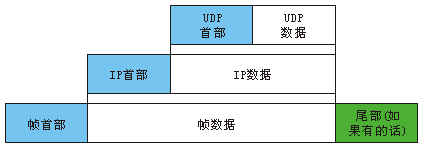


图6-3UDP封装

### 六.UDP校验和

UDP校验和的计算与IP和ICMP校验和的计算不同。UDP校验和校验的范围包括三部分：伪首部、UDP首部以及从应用层来的数据。

伪首部是IP首部的一部分，其中有些字段要填入0。用户数据报封装在IP数据包中。如下图所示：

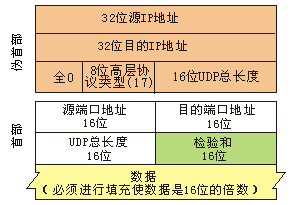


图6-4伪首部添加在UDP数据报上

若校验和不包括伪首部，用户数据报也可能是安全的和正确的。但是，若IP首部受到损伤，则它可能被交付到错误的主机。

伪首部中包含高层协议类型字段是为了确保这个数据包是属于UDP而不是属于TCP（参见实验七）的。使用UDP的进程和使用TCP的进程可以使用同一个端口号。UDP的高层协议类型字段是17。若在传输过程中这个值改变了，在接收端计算校验和时就可检测出来，UDP就可丢弃这个数据包。这样就不会交付给错误的协议。

#### 1.在发送端的校验和计算

在发送端按以下步骤计算校验和：

(1)把伪首部填加到UDP用户数据报上。

(2)把校验和字段填入零。

(3)按16位长度将数据报分段。

(4)若分段总数不是偶数，则增加一个分段的填充（全0）。填充只是为了计算校验和，计算完毕后就把它丢弃。

(5)把所有16位的分段使用反码算术运算相加。

(6)把得到的结果取反码，它是一个16位的数，把这个数插入到校验和字段。

(7)把伪首部和填充丢掉。

(8)把UDP用户数据报交付给IP进行封装。

在伪首部中的各行的顺序对校验和的计算没有任何影响。此外，增加0也不影响计算的结果。

下图给出了一个计算UDP校验和的例子。这里假定用户数据报的长度是15字节，因此要添加一个全0的字节。

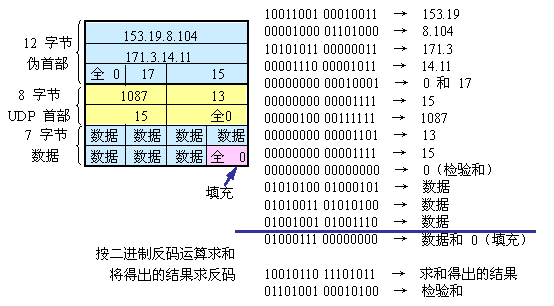


图6-5UDP校验和的计算过程

#### 2.在接收端的校验和计算

接收端按以下6个步骤计算校验和是否正确：

(1)把伪首部加到UDP用户数据报上。

(2)若需要，就增加填充。

(3)把数据报按16位长度分段。

(4)把所有16位的分段使用反码算术运算相加。

(5)把得到的结果取反码。

(6)若得到的结果是全零，则丢弃首部和填充，并接受这个用户数据报。若结果是非零，就丢弃这个用户数据报。

校验和是可选使用的，若不计算校验和，则校验和字段就填入0。

### 七.UDP应用

下面列出了UDP协议的一些用途：

●UDP适用于这样的进程，它需要简单的请求——响应通信，而较少考虑流量控制和差错控制。对于需要传送成块数据的进程，如FTP，则通常不使用UDP；

●UDP适用于具有内部流量控制和差错控制机制的进程；

●对多播和广播来说，UDP是个比较合适的传输层协议；

●UDP可用于管理进程，如SNMP协议；

●UDP可用于某些路由选择更新协议，如路由信息协议（RIP协议，参考实验17）。

### 八.协议栈实现代码解析

本实验将通过对安装目录JLCSS\ExpCNC\Work\NPL\ExpNPL\_studentnet\netproto\_udp\_student\netproto\_udp\_student下的proto\_udp\_student.h、netproto\_udp\_shudent.c和安装目录JLCSS\ExpCNC\Work\NPL\ExpNPL\_studentnet\netproto\_udp\_student\netproto\_udpif\_student下的netproto\_udpif\_student.h、netproto\_udpif\_student.c四个文件进行编码，完成协议栈中UDP协议的实现。

netproto\_udp\_student.h和netproto\_udp\_shudent.c文件用于实现UDP数据包发送和接收。其中，netproto\_udp\_student.h文件中定义了UDP协议实现相关数值以及UDP的负载内容、负载长度，关键代码如下所示：

#defineUDP\_SOUR\_PORT5893/\*\*<源端口号\*/  
#defineUDP\_DEST\_PORT5893/\*\*<目的端口号\*/  
#defineIP\_DEST\_ADDR"0.0.0.0"/\*\*<目的IP地址\*/  
#definePAYLOAD\_DATA"Hello,World!"/\*\*<有效负载\*/  
#definePAYLOAD\_LENsizeof(PAYLOAD\_DATA)/\*\*<有效负载长度\*/

这段代码定义了5个宏，他们代表的含义如下表所示：

表6-2netproto\_udp\_student.h中定义的宏

|  |  |  |
| --- | --- | --- |
| 宏 | 值 | 描述 |
| UDP\_SOUR\_PORT | 5893 | 定义UDP包头中“源端口”字段的值 |
| UDP\_DEST\_PORT | 5893 | 定义UDP包头中“目的端口”字段值 |
| IP\_DEST\_ADDR | "0.0.0.0" | 指定目的IP地址 |
| PAYLOAD\_DATA | "Hello,World!" | 自定义UDP负载内容 |
| PAYLOAD\_LEN | sizeof(PAYLOAD\_DATA) | 自定义UDP负载的长度 |

在实验的编码过程中，应该使用这些宏对相应的变量进行赋值。

netproto\_udp\_shudent.c文件是协议栈中UDP数据包发送和接收的实现部分，其中定义了2个函数。下面介绍这些协议栈的实现部分。

函数netp\_udp\_output\_student的功能是构造并发送一个UDP数据包，其高层协议为自定义协议类型，负载内容为自定义负载。这个函数的编码工作需要由学生完成。

当有数据到达本机网络接口时，函数netp\_udp\_input\_student将被调用，并传递给这个函数原始数据。在该函数中，需要判断一些条件值来确定接收到的数据包为自定义UDP数据，如果是自定义UDP数据包，则输出负载内容，如果不是，则返回NETP\_PUSH\_TO\_LWIP交给协议栈继续处理。

netproto\_udpif\_student.h和netproto\_udpif\_shudent.c文件用于实现UDP上层投递的功能，即为高层使用UDP协议提供了接口。其中，netproto\_udpif\_student.h文件中并没有定义太多内容。netproto\_udpif\_shudent.c文件是协议栈中UDP上层投递的功能的实现部分，其中定义了一个全局变量recv\_port和2个函数。

全局变量recv\_port的作用很简单，它记录了发送UDP数据报时的源端口号作为接收UDP数据报的过滤条件。

函数netp\_send\_udp通过IP层接口发送UDP数据报，该函数功能需要学生完成。

函数netp\_udp\_input\_student处理输入数据包，如果输入的数据报满足过滤条件，则投递给上层协议使用。该函数功能需要学生完成。

在编码过程中可能会设计到一些结构体、宏和函数，下表是对他们进行介绍：

表6-3实验涉及的结构体、宏和函数

| 结构体/宏/函数 | 声明或定义 | 描述 |
| --- | --- | --- |
| struct  netp\_eth\_header | structnetp\_eth\_header{  u8\_tdest\_address[ETH\_ADDRESS\_LEN];  u8\_tsour\_address[ETH\_ADDRESS\_LEN];  u16\_ttype;  }; | 以太网帧头结构 |
| structnetp\_ip\_header | structnetp\_ip\_header{  u8\_theaderlen:4;  u8\_tversion:4;  u8\_tdiff\_services;  u16\_ttotal\_length;  u16\_tidentification;  u16\_tflags\_offset;  u8\_ttime\_to\_live;  u8\_tprotocol;  u16\_theader\_checksum;  structip\_addrsource\_address;  structip\_addrdestination\_address;  }; | ipv4包头结构 |
| structnetp\_udp\_header | structnetp\_udp\_header{  u16\_tsour\_port;  u16\_tdest\_port;  u16\_tlength;  u16\_tchecksum;  }; | UDP报头结构 |
| structnetp\_udp\_pseudo | structnetp\_udp\_pseudo{  u32\_tsour\_addr;  u32\_tdest\_addr;  u8\_tzero;  u8\_tprotocol;  u16\_tlength;  }; | UDP伪首部结构 |
| structin\_addr | structin\_addr{  u32\_ts\_addr;  }; | 32位地址 |
| UDP\_HEADER\_LEN | #defineUDP\_HEADER\_LEN8 | UDP包头长度 |
| PAYLOAD\_DATA | #definePAYLOAD\_DATA"Hello,EXPcns!" | UDP负载内容 |
| PAYLOAD\_LEN | #definePAYLOAD\_LENsizeof(PAYLOAD\_DATA) | UDP负载长度 |
| NETP\_UDP\_PSEUDO\_LEN | #defineNETP\_UDP\_PSEUDO\_LEN  sizeof(structnetp\_udp\_pseudo) | UDP伪首部长度 |
| ETH\_HEADER\_LEN | #defineETH\_HEADER\_LEN14 | 以太网帧头长度 |
| IP\_HEADER\_LEN | #defineIP\_HEADER\_LEN20 | IP包头长度 |
| netp\_current\_udp\_addr | u32\_tnetp\_current\_udp\_addr(); | 获取当前正在使用的网络接口的UDP地址 |
| htons | u16\_thtons(u16\_tn); | 将16位数值由主机字节序转换为网络字节序 |
| inet\_addr | u32\_tinet\_addr(constchar\*cp); | 将ASCII编码的Internet地址转换成为网络字节序地址 |
| inet\_chksum | u16\_tinet\_chksum(void\*dataptr,u16\_tlen) | 计算校验和 |

### 九.各模块推荐流程

#### 1.UDP数据包发送流程

编码实现UDP数据包发送推荐使用如下流程：

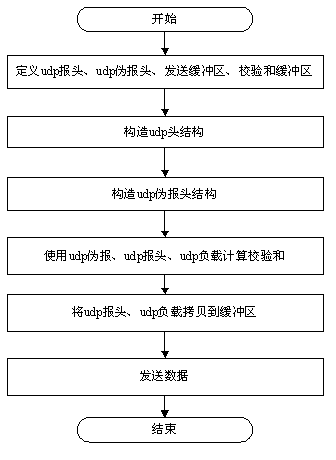


图6-6UDP数据包发送推荐流程

#### 2.输入UDP数据包处理流程

编码实现处理UDP输入数据包推荐使用如下流程：

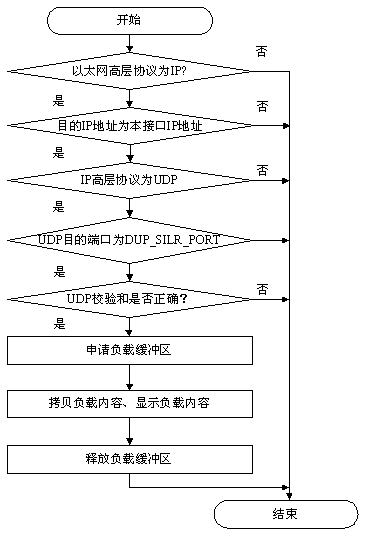


图6-7处理UDP输入数据包推荐流程

#### 3.UDP发送接口实现流程

编码实现UDP发送接口推荐使用如下流程：

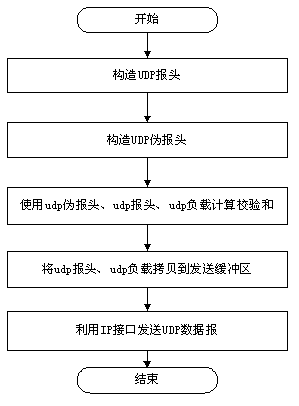


图6-8UDP发送接口实现推荐流程

#### 4.UDP接收接口实现流程

编码实现UDP接收接口推荐使用如下流程：

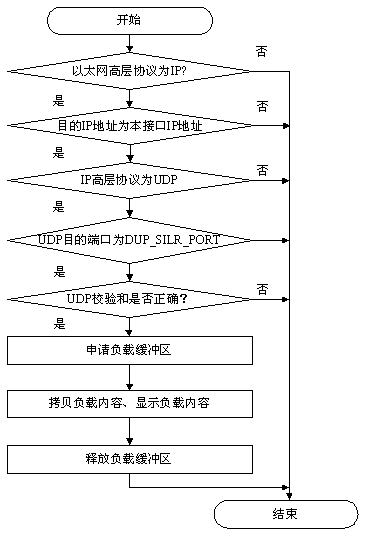


图6-9UDP接收接口实现推荐流程

【实验步骤】

### 练习1编辑并发送UDP数据报

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。

1.主机A打开协议编辑器，编辑发送给主机B的UDP数据报。

MAC层：

目的MAC地址：接收方MAC地址

源MAC地址：发送方MAC地址

协议类型或数据长度：0800，即IP协议

IP层：

总长度：包括IP层、UDP层和数据长度

高层协议类型：17，即UDP协议

首部校验和：其它所有字段填充完毕后填充此字段

源IP地址：发送方IP地址

目的IP地址：接收方IP地址

UDP层：

源端口：1030

目的端口：大于1024的端口号

有效负载长度：UDP层及其上层协议长度

其它字段默认，计算校验和。

●UDP在计算校验和时包括哪些内容？

2.在主机B上启动协议分析器捕获数据，并设置过滤条件（提取UDP协议）。

3.主机A发送已编辑好的数据报。

4.主机B停止捕获数据，在捕获到的数据中查找主机A所发送的数据报。

思考问题：

1.为什么UDP协议的“校验和”要包含伪首部？

2.比较UDP和IP的不可靠程度？

### 练习2UDP单播通信

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.主机B、C、D、E、F上启动实验平台工具栏中的UDP工具”，作为服务器端，监听端口设置为2483，“创建”成功。

2.主机C、E上启动协议分析器开始捕获数据，并设置过滤条件（提取UDP协议）。

3.主机A上启动实验平台工具栏中的“UDP工具”，作为客户端，以主机C的IP为目的IP地址，以2483为端口，填写数据并发送。

4.察看主机B、C、D、E、F上的“UDP工具”接收的信息。

●哪台主机上的“UDP工具”能够接收到主机A发送的UDP报文？

5.察看主机C协议分析器上的UDP报文，并回答以下问题：

●UDP是基于连接的协议吗？阐述此特性的优缺点。

●UDP报文交互中含有确认报文吗？阐述此特性的优缺点。

6.主机A上使用协议编辑器向主机E发送UDP报文，其中：

目的MAC地址：E的MAC地址

目的IP地址：主机E的IP地址

目的端口：2483

校验和：0

有效负载长度：UDP层及其上层协议长度

首部校验和：其它所有字段填充完毕后填充此字段

总长度：包括IP层、UDP层和数据长度

发送此报文，并回答以下问题：

●主机E上的UDP通信程序是否接收到此数据包？UDP是否可以使用0作为校验和进行通信？

7.主机B、C、D、E、F关闭服务端，主机A关闭客户端。

思考问题：

1.思考UDP的差错处理能力。

### 练习3UDP广播通信

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.主机B、C、D、E、F上启动实验平台工具栏中的“UDP工具”，作为服务器端，监听端口设为2483。

2.主机B、C、D、E、F启动协议分析器捕获数据，并设置过滤条件（提取UDP协议）。

3.主机A上启动UDP工具，作为客户端，以255.255.255.255为目的地址，以2483为端口，填写数据并发送。

4.察看主机B、C、D、E、F上的“UDP工具”接收的信息。

●哪台主机能够接收到主机A发送的UDP报文？

5.察看协议分析器上捕获的UDP报文，并回答以下问题：

●主机A发送的报文的目的MAC地址和目的IP地址的含义是什么？

思考问题：

1.如果将目的MAC地址换成某一个主机的MAC地址，是否所有主机还会收到这种报文？

2.如果将目的MAC地址设成广播地址，目的IP设成某一主机的IP地址，结果怎样？

3.在可靠性不是最重要的情况下，UDP可能是一个好的传输协议。试给出这种特定情况的一些示例。

4.UDP协议本身是否能确保数据报的发送和接收顺序？

### 练习4UDP数据报发送与接收

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。实验开始前，先单击“初始环境”。

在实验中主机的调试接口IP地址分别设为172.16.1.1n1、172.16.1.1n2、172.16.1.1n3、172.16.1.1n4、172.16.1.1n5、172.16.1.1n6(n为组别号，目的是为了避免IP地址冲突)，所有主机使用子网掩码255.255.255.0，默认网关设置为0.0.0.0。

以第一组为例，主机A将调试接口的IP地址设置为172.16.1.111、主机B使用处于连接状态的物理接口，将调试接口的IP设置为172.16.1.112、主机C将调试接口的IP地址设置为172.16.1.113、主机D使用处于连接状态的物理接口，将调试接口的IP地址设置为172.16.1.114、主机E使用处于连接状态的物理接口，将调试接口的IP地址设置为172.16.1.115、主机F将调试接口的IP地址设置为172.16.1.116。所有主机使用子网掩码255.255.255.0，默认网关设置为0.0.0.0。

1. 设置目的IP地址

各主机使用VS2005软件打开实验平台安装目录下的JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_udp\_student\netproto\_udp\_student.sln文件，在头文件netproto\_udp\_student.h内将IP\_DEST\_ADDR宏所定义的IP地址修改为同组同学的新接口IP地址。

注：若实验平台安装在C盘，则目录为：C:\Program Files\JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_udp\_student\netproto\_udp\_student.sln。

2. 编码实现发送UDP数据包

(1) 所有主机将VS2005软件打开后，依次选择【文件】->【打开】->【项目/解决方案】，打开“打开项目”对话框，选择平台安装目录（本实验平台安装在C盘）C:\Program Files\JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_udp\_student\netproto\_udp\_student.sln文件，在源文件netproto\_udp\_stuent.c的netp\_udp\_output\_student函数内编写实现代码。

(2) 参考实验原理UDP数据包发送推荐流程图给出的流程，分析已经存在的代码。

已经存在的代码创建了UDP头结构、UDP伪首部结构、源IP地址、目的IP地址、发送缓冲区和校验和缓冲区等变量，UDP校验和的计算过程，将UDP头和负载拷贝到发送缓冲区中以及将缓冲区数据发送到网络中的实现。

(3) 构造UDP头，校验和字段为0

构造并填充一个UDP数据包头。可以使用netproto\_udp\_student.h文件中所定义的端口值。总长度为UDP\_HEADER\_LEN+PAYLOAD\_LEN。

(4) 构造UDP伪首部

各主机构造并填充UDP伪首部。其中，源IP地址为本接口IP地址，目的IP地址为IP\_DEST\_ADDR所指定的IP地址，长度与UDP头中的总长度数值相同。

3. 所有主机编码实现UDP数据包输入处理功能

(1)各主机使用VS2005软件打开实验平台安装目录下的JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_udp\_student\netproto\_udp\_student.sln文件，在源函数netproto\_udp\_studnet.c的函数netp\_udp\_input\_student内编写实现代码。

注：若实验平台安装在C盘，则目录为：C:\Program Files\JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_udp\_student\netproto\_udp\_student.sln。

(2) 参考实验原理处理UDP输入数据包推荐流程图给出的流程，分析已经存在的代码。

已经存在的代码定义了以太网帧头、IP包头、UDP报头、UDP伪报头、校验和、负载缓冲区指针和校验和缓冲区指针，以太网高层协议的校验，目的IP地址的校验，申请负载缓冲区，打印负载内容等实现。

(3) 提取UDP数据包

各主机通过以IP头结构ip\_header中的protocol即“高层协议类型”字段值判断该数据帧的高层协议是否为UDP协议，可以使用宏IP\_PROTO\_UDP。如果不是UDP数据包则返回NETP\_PUSH\_TO\_LWIP交给协议栈继续执行。

(4) 提取UDP目的端口号为UDP\_SOUR\_PORT的数据报

各主机通过UDP头结构udp\_header中的dest\_port即“目的端口号”字段值判断该UDP数据报是否是我们感兴趣的UDP数据报。如果不是则返回NETP\_PUSH\_TO\_LWIP交给协议栈继续执行。

(5) 判断UDP校验和是否正确

如果UDP头的校验和字段值不为0，则需要验证校验和是否正确。各主机通过IP包头和UDP报头中的数值验证UDP校验和是否正确，如果不正确则丢弃该数据报。

4. 所有主机打开协议分析器，开始捕获数据

5. 所有主机调试并运行程序

6. 各主机停止数据捕获，观察实验现象

● 你收到的负载内容是什么？

7.参考代码如下：

/\*\*  
\*\brief编辑并发送一个UDP数据包  
\*  
\*由学生完成这个函数，发送一个UDP数据包。主线程将会调用这个函数。  
\*/  
void  
netp\_udp\_output\_student()  
{  
//定义udp报头、udp伪报头、发送缓冲区、校验和缓冲区  
structnetp\_udp\_headerudp\_header;  
structnetp\_udp\_pseudoudp\_pseudo;  
structip\_addrsour\_ipaddr;  
structip\_addrdest\_ipaddr;  
u8\_tsend\_buff[UDP\_HEADER\_LEN+PAYLOAD\_LEN];  
u8\_tchecksum\_buff[NETP\_UDP\_PSEUDO\_LEN+UDP\_HEADER\_LEN+PAYLOAD\_LEN];  
  
//构造udp头结构  
udp\_header.sour\_port=htons(UDP\_SOUR\_PORT);  
udp\_header.dest\_port=htons(UDP\_DEST\_PORT);  
udp\_header.length=htons(UDP\_HEADER\_LEN+PAYLOAD\_LEN);  
udp\_header.checksum=0;  
  
//构造udp伪报头结构  
sour\_ipaddr.addr=netp\_current\_ip\_addr();  
dest\_ipaddr.addr=inet\_addr(IP\_DEST\_ADDR);  
udp\_pseudo.sour\_addr=sour\_ipaddr.addr;  
udp\_pseudo.dest\_addr=dest\_ipaddr.addr;  
udp\_pseudo.zero=0;  
udp\_pseudo.protocol=IP\_PROTO\_UDP;  
udp\_pseudo.length=udp\_header.length;  
  
//将udp伪报头、udp报头、dup负载拷贝到校验和缓冲区  
memcpy(checksum\_buff,&udp\_pseudo,NETP\_UDP\_PSEUDO\_LEN);  
memcpy(checksum\_buff+NETP\_UDP\_PSEUDO\_LEN,&udp\_header,UDP\_HEADER\_LEN);  
memcpy(checksum\_buff+NETP\_UDP\_PSEUDO\_LEN+UDP\_HEADER\_LEN,  
PAYLOAD\_DATA,PAYLOAD\_LEN);  
  
//计算校验和  
udp\_header.checksum=inet\_chksum(checksum\_buff,  
NETP\_UDP\_PSEUDO\_LEN+UDP\_HEADER\_LEN+PAYLOAD\_LEN);

//将udp报头、udp负载拷贝到发送缓冲区  
memcpy(send\_buff,&udp\_header,UDP\_HEADER\_LEN);  
memcpy(send\_buff+UDP\_HEADER\_LEN,PAYLOAD\_DATA,PAYLOAD\_LEN);  
  
//发送数据  
if(ERR\_OK==netp\_udp\_output(&sour\_ipaddr,&dest\_ipaddr,  
send\_buff,IP\_HEADER\_LEN+PAYLOAD\_LEN,0)){  
printf("发送UDP数据包----成功！\n");  
}else{  
printf("发送UDP数据包----失败！\n");  
}  
}  
  
/\*\*  
\*\brief当有数据帧到达时，将调用这个函数  
\*  
\*\parampacket指向接收到的数据  
\*\parampacket\_len数据帧的长度  
\*  
\*\return一个put\_to\_lwip类型的返回值。  
\*返回NETP\_PUT\_LWIP表示处理数据后将数据帧交给上层协议栈继续处理  
\*返回NETP\_NO\_PUT\_LIWP表示不将数据帧交给上层协议栈处理。  
\*/  
enumpush\_to\_lwip  
netp\_udp\_input\_student(void\*packet,intpacket\_len)  
{  
//定义以太网帧头、IP包头、UDP报头、UDP伪报头、校验和  
//负载缓冲区指针和校验和缓冲区指针  
structnetp\_eth\_headereth\_header;  
structnetp\_ip\_headerip\_header;  
structnetp\_udp\_headerudp\_header;  
structnetp\_udp\_pseudoudp\_pseudo;  
u16\_tudp\_checksum;  
char\*payload\_buff;  
u8\_t\*checksum\_buff;  
  
memcpy(&eth\_header,packet,ETH\_HEADER\_LEN);  
  
//只提取ip数据包  
if(eth\_header.type!=htons(MAC\_PROTO\_IP)){  
returnNETP\_PUSH\_TO\_LWIP;  
}  
  
memcpy(&ip\_header,(u8\_t\*)packet+ETH\_HEADER\_LEN,IP\_HEADER\_LEN);  
//过滤掉目的ip地址不是本接口的数据包  
if(ip\_header.destination\_address.addr!=netp\_current\_ip\_addr()){  
returnNETP\_PUSH\_TO\_LWIP;  
}  
  
//只提取UDP数据报  
if(ip\_header.protocol!=IP\_PROTO\_UDP){  
returnNETP\_PUSH\_TO\_LWIP;  
}  
  
memcpy(&udp\_header,  
(u8\_t\*)packet+ETH\_HEADER\_LEN+IP\_HEADER\_LEN,UDP\_HEADER\_LEN);  
  
//过滤掉目的端口号不是UDP\_SOUR\_PORT的数据报  
if(udp\_header.dest\_port!=htons(UDP\_DEST\_PORT)){  
returnNETP\_PUSH\_TO\_LWIP;  
}

//判断UDP校验和是否正确  
if(0!=udp\_header.checksum){  
checksum\_buff=malloc(NETP\_UDP\_PSEUDO\_LEN+udp\_header.length);  
udp\_checksum=udp\_header.checksum;  
udp\_header.checksum=0;  
udp\_pseudo.dest\_addr=ip\_header.destination\_address.addr;  
udp\_pseudo.sour\_addr=ip\_header.source\_address.addr;  
udp\_pseudo.zero=0;  
udp\_pseudo.protocol=IP\_PROTO\_UDP;  
udp\_pseudo.length=udp\_header.length;  
  
memcpy(checksum\_buff,&udp\_pseudo,NETP\_UDP\_PSEUDO\_LEN);  
memcpy(checksum\_buff+NETP\_UDP\_PSEUDO\_LEN,&udp\_header,UDP\_HEADER\_LEN);  
memcpy(checksum\_buff+NETP\_UDP\_PSEUDO\_LEN+UDP\_HEADER\_LEN,  
(u8\_t\*)packet+ETH\_HEADER\_LEN+IP\_HEADER\_LEN+UDP\_HEADER\_LEN,  
ntohs(udp\_header.length)-UDP\_HEADER\_LEN);  
  
if(udp\_checksum!=inet\_chksum(checksum\_buff,  
NETP\_UDP\_PSEUDO\_LEN+ntohs(udp\_header.length))){  
returnNETP\_PUSH\_TO\_LWIP;  
}  
  
free(checksum\_buff);  
}  
  
//申请负载缓冲区  
payload\_buff=malloc(udp\_header.length-UDP\_HEADER\_LEN);  
  
//拷贝负载内容到负载缓冲区  
memcpy(payload\_buff,  
(u8\_t\*)packet+ETH\_HEADER\_LEN+IP\_HEADER\_LEN+UDP\_HEADER\_LEN,  
udp\_header.length-UDP\_HEADER\_LEN);  
  
//打印负载内容  
printf("接收到目的端口为：%x的UDP数据报，负载为：%s\n",  
ntohs(udp\_header.dest\_port),(char\*)payload\_buff);  
  
//释放负载缓冲区  
free(payload\_buff);  
  
returnNETP\_NO\_PUSH\_LIWP;  
}

### 练习5UDP报文的上层投递的设计与实现

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。实验开始前，先单击“初始环境”。

在实验中主机的调试接口IP地址分别设为172.16.1.1n1、172.16.1.1n2、172.16.1.1n3、172.16.1.1n4、172.16.1.1n5、172.16.1.1n6(n为组别号，目的是为了避免IP地址冲突)，所有主机使用子网掩码255.255.255.0，默认网关设置为0.0.0.0。

以第一组为例，主机A将调试接口的IP地址设置为172.16.1.111、主机B使用处于连接状态的物理接口，将调试接口的IP设置为172.16.1.112、主机C将调试接口的IP地址设置为172.16.1.113、主机D使用处于连接状态的物理接口，将调试接口的IP地址设置为172.16.1.114、主机E使用处于连接状态的物理接口，将调试接口的IP地址设置为172.16.1.115、主机F将调试接口的IP地址设置为172.16.1.116。所有主机使用子网掩码255.255.255.0，默认网关设置为0.0.0.0。

1. 编码实现发送UDP数据报接口

(1) 各主机使用VS2005软件打开实验平台安装目录下的JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_udp\_student\netproto\_udp\_student.sln文件，在源函数netproto\_udpif\_student.c的函数netp\_send\_udp内编写实现代码。

注：若实验平台安装在C盘，则目录为：C:\Program Files\JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_udp\_student\netproto\_udp\_student.sln。

(2) 参考实验原理UDP发送接口实现推荐流程图给出的流程，分析已经存在的代码。

已经存在的代码创建了UDP头结构、UDP伪首部结构、源IP地址、负载缓冲区指针和校验和缓冲区指针等变量，申请负载缓冲区、校验和缓冲区，UDP校验和的计算过程，将UDP头和负载拷贝到发送缓冲区中以及利用IP接口将缓冲区数据发送到网络中的实现。

(3) 构造UDP头，校验和字段为0

构造并填充一个UDP数据包头。可以使用VS2005软件打开实验平台安装目录下的JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_udp\_student\netproto\_udp\_student.sln文件，在头文件netproto\_udp\_student.h中所定义的端口值。总长度为UDP\_HEADER\_LEN+ buff\_len。

(4) 构造UDP伪首部

各主机构造并填充UDP伪首部。其中，源IP地址为本接口IP地址，目的IP地址为用户所指定的目的IP地址，长度与UDP头中的总长度数值相同。

2. 编码实现接收UDP数据报接口

(1) 各主机打开netproto\_ udpif \_student.c文件，在函数netp\_udp\_input\_student内编写实现代码。

(2) 参考实验原理UDP接收接口实现推荐流程图给出的流程，分析已经存在的代码。

已经存在的代码定义了以太网帧头、IP包头、UDP报头、UDP伪报头、校验和、负载缓冲区指针和校验和缓冲区指针，以太网高层协议的校验，目的IP地址的校验，申请负载缓冲区，将负载内容投递到上层协议、释放负载缓冲区等实现。

(3) 提取UDP数据包

各主机通过以IP头结构ip\_header中的protocol即“高层协议类型”字段值判断该数据帧的高层协议是否为UDP协议，可以使用宏IP\_PROTO\_UDP。如果不是UDP数据包则返回NETP\_PUSH\_TO\_LWIP交给协议栈继续执行。

(4) 提取UDP目的端口号为recv\_port的数据报

各主机通过UDP头结构udp\_header中的dest\_port即“目的端口号”字段值判断该UDP数据报是否是我们感兴趣的UDP数据报。如果不是则返回NETP\_PUSH\_TO\_LWIP交给协议栈继续执行。

(5) 判断UDP校验和是否正确

如果UDP头的校验和字段值不为0，则需要验证校验和是否正确。各主机通过IP包头和UDP报头中的数值验证UDP校验和是否正确，如果不正确则丢弃该数据报。

3. 所有主机打开协议分析器，开始捕获数据

4. 所有主机调试并运行程序

5.各主机停止数据捕获，观察实验现象

● 你收到的负载内容是什么？

6.参考代码如下：

/\*\*  
\*\brief对上层的接口，负责利用IP接口，发送一个UDP数据报  
\*  
\*由学生完成这个函数，发送一个UDP数据包。udpif将会调用这个函数。  
\*  
\*\paramdest\_ipaddr目的IP地址  
\*\paramdest\_port目的端口  
\*\paramsourport源端口  
\*\parambuff负载缓冲区  
\*\buff\_len负载长度  
\*  
\*\return0发送成功，-1发送失败  
\*/  
int  
netp\_send\_udp(structip\_addrdest\_ipaddr,u16\_tdest\_port,u16\_tsour\_port,  
void\*buff,intbuff\_len)  
{  
structnetp\_udp\_headerudp\_header;//UDP报头  
structnetp\_udp\_pseudoudp\_pseudo;//UDP伪首部  
structip\_addrsour\_ipaddr;//源IP地址  
u8\_t\*send\_buff;//负载缓冲区指针  
u8\_t\*checksum\_buff;//校验和缓冲区指针  
  
//申请负载缓冲区、校验和缓冲区  
send\_buff=malloc(UDP\_HEADER\_LEN+buff\_len);  
checksum\_buff=malloc(NETP\_UDP\_PSEUDO\_LEN+UDP\_HEADER\_LEN+buff\_len);  
  
//源IP地址设置为本接口IP地址  
sour\_ipaddr.addr=netp\_current\_ip\_addr();  
  
//构造UDP报头  
udp\_header.sour\_port=htons(sour\_port);  
udp\_header.dest\_port=htons(dest\_port);  
udp\_header.length=htons(UDP\_HEADER\_LEN+buff\_len);  
udp\_header.checksum=0;  
  
//构造UDP伪报头  
udp\_pseudo.sour\_addr=sour\_ipaddr.addr;  
udp\_pseudo.dest\_addr=dest\_ipaddr.addr;  
udp\_pseudo.zero=0;  
udp\_pseudo.protocol=IP\_PROTO\_UDP;  
udp\_pseudo.length=udp\_header.length;  
  
//将udp伪报头、udp报头、dup负载拷贝到校验和缓冲区  
memcpy(checksum\_buff,&udp\_pseudo,NETP\_UDP\_PSEUDO\_LEN);  
memcpy(checksum\_buff+NETP\_UDP\_PSEUDO\_LEN,&udp\_header,UDP\_HEADER\_LEN);  
memcpy(checksum\_buff+NETP\_UDP\_PSEUDO\_LEN+UDP\_HEADER\_LEN,buff,buff\_len);  
  
//计算校验和  
udp\_header.checksum=inet\_chksum(checksum\_buff,NETP\_UDP\_PSEUDO\_LEN+UDP\_HEADER\_LEN+(u16\_t)buff\_len);  
  
free(checksum\_buff);  
  
//将udp报头、udp负载拷贝到发送缓冲区  
memcpy(send\_buff,&udp\_header,UDP\_HEADER\_LEN);  
memcpy(send\_buff+UDP\_HEADER\_LEN,buff,buff\_len);  
  
//利用IP接口发送UDP数据报  
if(ERR\_OK==netp\_ip\_payload\_output(&sour\_ipaddr,&dest\_ipaddr,  
NETP\_TTL,NETP\_DS,IP\_PROTO\_UDP,send\_buff,UDP\_HEADER\_LEN+buff\_len,0)){  
recv\_port=sour\_port;  
free(send\_buff);  
return0;  
}else{  
free(send\_buff);  
return-1;  
}  
}  
  
/\*\*  
\*\brief当有数据帧到达时，将调用这个函数  
\*  
\*\parampacket指向接收到的数据  
\*\parampacket\_len数据帧的长度  
\*  
\*\return一个put\_to\_lwip类型的返回值。  
\*返回NETP\_PUT\_LWIP表示处理数据后将数据帧交给上层协议栈继续处理  
\*返回NETP\_NO\_PUT\_LIWP表示不将数据帧交给上层协议栈处理。  
\*/  
enumpush\_to\_lwip  
netp\_udp\_input\_student(void\*packet,intpacket\_len)  
{  
structnetp\_eth\_headereth\_header;  
structnetp\_ip\_headerip\_header;  
structnetp\_udp\_headerudp\_header;  
structnetp\_udp\_pseudoudp\_pseudo;  
u16\_tudp\_checksum;  
char\*payload\_buff;  
u8\_t\*checksum\_buff;  
memcpy(&eth\_header,packet,ETH\_HEADER\_LEN);  
  
//只提取ip数据包  
if(eth\_header.type!=htons(MAC\_PROTO\_IP)){  
returnNETP\_PUSH\_TO\_LWIP;  
}  
  
memcpy(&ip\_header,(u8\_t\*)packet+ETH\_HEADER\_LEN,IP\_HEADER\_LEN);  
//过滤掉目的ip地址不是本接口的数据包  
if(ip\_header.destination\_address.addr!=netp\_current\_ip\_addr()){  
returnNETP\_PUSH\_TO\_LWIP;  
}  
  
//只提取UDP数据报  
if(ip\_header.protocol!=IP\_PROTO\_UDP){  
returnNETP\_PUSH\_TO\_LWIP;  
}  
  
memcpy(&udp\_header,  
 (u8\_t\*)packet+ETH\_HEADER\_LEN+IP\_HEADER\_LEN,UDP\_HEADER\_LEN);  
  
//过滤掉目的端口号不是recv\_port的数据报  
if(udp\_header.dest\_port!=htons(recv\_port)){  
returnNETP\_PUSH\_TO\_LWIP;  
}  
  
//判断UDP校验和是否正确  
if(0!=udp\_header.checksum){  
checksum\_buff=malloc(NETP\_UDP\_PSEUDO\_LEN+udp\_header.length);  
udp\_checksum=udp\_header.checksum;  
udp\_header.checksum=0;  
udp\_pseudo.dest\_addr=ip\_header.destination\_address.addr;  
udp\_pseudo.sour\_addr=ip\_header.source\_address.addr;  
udp\_pseudo.zero=0;  
udp\_pseudo.protocol=IP\_PROTO\_UDP;  
udp\_pseudo.length=udp\_header.length;  
  
memcpy(checksum\_buff,&udp\_pseudo,NETP\_UDP\_PSEUDO\_LEN);  
memcpy(checksum\_buff+NETP\_UDP\_PSEUDO\_LEN,&udp\_header,UDP\_HEADER\_LEN);  
memcpy(checksum\_buff+NETP\_UDP\_PSEUDO\_LEN+UDP\_HEADER\_LEN,  
(u8\_t\*)packet+ETH\_HEADER\_LEN+IP\_HEADER\_LEN+UDP\_HEADER\_LEN,  
ntohs(udp\_header.length)-UDP\_HEADER\_LEN);

if(udp\_checksum!=inet\_chksum(checksum\_buff,  
NETP\_UDP\_PSEUDO\_LEN+ntohs(udp\_header.length))){  
returnNETP\_PUSH\_TO\_LWIP;  
}  
free(checksum\_buff);  
}  
  
//申请负载缓冲区  
payload\_buff=malloc(udp\_header.length-UDP\_HEADER\_LEN);  
  
//拷贝负载内容到负载缓冲区  
memcpy(payload\_buff,(u8\_t\*)packet+ETH\_HEADER\_LEN+IP\_HEADER\_LEN+UDP\_HEADER\_LEN,  
udp\_header.length-UDP\_HEADER\_LEN);  
  
//提交给高层处理  
netp\_user\_input(&ip\_header.source\_address,payload\_buff,udp\_header.length-UDP\_HEADER\_LEN);  
  
//释放负载缓冲区  
free(payload\_buff);  
  
returnNETP\_NO\_PUSH\_LIWP;  
}

## 实验7传输控制协议（TCP）

【实验目的】

1.掌握TCP协议的报文格式

2.掌握TCP连接的建立和释放过程

3.掌握TCP数据传输中编号与确认的过程

4.掌握TCP协议校验和的计算方法

5.理解TCP重传机制

【学时分配】

4学时

【实验环境】

该实验采用网络结构一

【实验原理】

### 一.TCP协议简介

TCP（传输控制协议）协议是TCP/IP协议族中的面向连接的、可靠的传输层协议。TCP与UDP不同，它允许发送和接收字节流形式的数据。为了使服务器和客户端以不同的速度发送和接收数据，TCP提供了发送和接收两个缓冲区。TCP提供全双工服务，数据同时能双向流动。通信的每一方都有发送和接收两个缓冲区，可以双向发送数据。TCP在报文中加上一个递增的确认序列号来告诉发送端，接收端期望收到的下一个报文，如果在规定时间内，没有收到关于这个包的确认响应，则重新发送此包，这保证了TCP是一种可靠的传输层协议。

TCP的常用熟知端口如下表所示：

表7-1TCP常用熟知端口

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 协议 | 端口 | 协议 |
| 20 | FTPData | 80 | HTTP |
| 21 | FTPControl | 110 | POP3 |
| 23 | TELNET | 143 | IMAP |
| 25 | SMTP |  |  |

### 二.TCP报文格式

TCP报文的格式如下图所示：

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 源端口（16位） | | | | | | | | | | 目的端口（16位） | |
| 序列号（32位） | | | | | | | | | | | |
| 确认号（32位） | | | | | | | | | | | |
| 首部长度（4位） | 保留  （4位） | CWR | ECE | URG | ACK | PSH | RST | SYN | FIN | | 窗口大小（16位） |
| 校验和（16位） | | | | | | | | | | | 紧急指针（16位） |
| 选项和填充 | | | | | | | | | | | |

图7-1TCP报文格式

TCP报文包括20～60字节的首部，接着是应用程序的数据部分。首部在没有选项时是20字节，而当有选项时长度会增加，但是最大不会超过60字节。

●源端口：该字段定义了在主机中发送这个报文的应用程序端口号。

●目的端口：该字段定义了数据报发往的主机中接收这个报文的应用程序的端口号。

●序列号：该字段定义了指派给本报文第一个数据字节的一个序号。TCP是流式传输协议，为了保证连通性，要在发送的每一个字节上编号。序号指定了这个序列中的哪一个字节是报文的第一个字节。在连接建立时，双方使用随机数产生器产生初始序号，通常每一方的初始序号都是不同的。

●确认号：该字段定义了报文的接收端期望从对方接收的序号。如果报文的接收端成功地接收了对方发来的序号为x的报文，它就把确认号定义为x+1。确认可以和数据一起发送。

●首部长度：该字段指定TCP首部的长度，以4字节为单位。首部长度可以在20～60字节之间。因此，这个字段的值可以在5至15之间。

●保留：这是6位字段，保留为今后使用。

●控制：这个字段定义了8种不同的标志。如下图所示。在同一时间可设置一位或多位标志。

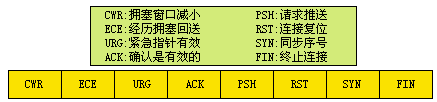


图7-2控制字段

这些标志用在TCP的流量控制、连接建立和终止以及数据传送的方式等方面。下表给出了每一位的简要说明。

表7-2TCP标志位

| 标志 | 说明 |
| --- | --- |
| CWR | 拥塞窗口减小（用来表明发送主机接收到了设置ECE标志的TCP包。拥塞窗口是被TCP维护的一个内部变量，用来管理发送窗口大小） |
| ECE | 经历拥塞回送（用来在TCP3次握手时表明一个TCP端是具备ECN功能的，并且表明接收到的TCP包的IP头部的ECN被设置为11） |
| URG | 紧急指针字段值有效 |
| ACK | 确认字段值有效 |
| PSH | 推送数据 |
| RST | 连接必须复位 |
| SYN | 在连接建立时对序号进行同步 |
| FIN | 终止连接 |

●窗口大小：该字段定义对方必须维持的窗口值（以字节为单位）。这个字段的长度是16位，因此窗口值的最大长度是65535字节。这个值通常是作为接收窗口，并由接收端来确定。这时，发送端必须服从接收端的决定。

●校验和：该字段的校验范围包括伪首部、TCP首部和TCP数据部分。

●紧急指针：只有当紧急标志置位时，这个16位字段才有效，这时的报文中包括紧急数据。

●选项：在TCP首部中可以有多达40字节的可选信息。

### 三.TCP封装

TCP报文封装在IP数据报中，然后再封装成数据链路层中的帧，如下图所示：

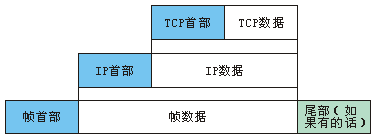


图7-3TCP封装

### 四.TCP校验和

TCP的校验和与UDP的校验和计算过程是一样的。但是，UDP是否使用校验和是可选的，而TCP是否使用校验和则是强制性的。在计算TCP校验和时也要在报文上添加伪首部。对于TCP的伪首部，高层协议类型字段的值是6。如下图所示：

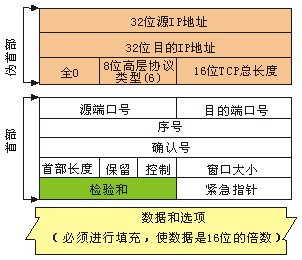


图7-4伪首部加到TCP报文上

### 五.TCP连接建立与释放

#### 1.连接建立

TCP以全双工方式传送数据。当两个进程建立了TCP连接后，它们能够同时向对方发送数据。在传送数据之前，双方都要对通信进行初始化，得到对方的认可。

#### 2.三次握手

TCP的连接建立过程叫做三次握手。服务器程序首先准备好接受TCP连接，这个过程叫做被动打开请求。这时，服务器的TCP就已准备好接受任何一台主机的TCP连接了。

客户程序发出TCP连接请求的过程叫做主动打开。然后服务器与客户端就开始三次握手过程，如下图所示（在图中客户端与服务器端各使用一条时间线，并给出每个阶段的几个重要字段，包括序号、确认号、控制标志以及非零的窗口值）。这个过程有以下3个步骤。

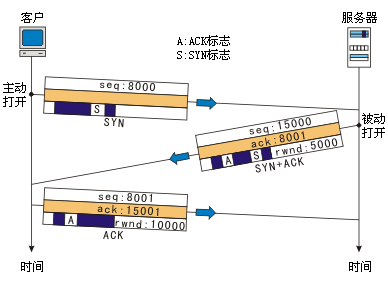


图7-5使用三次握手的连接建立

(1)客户发送第一个报文，这是一个SYN报文，在这个报文中只有SYN标志置为1。这个报文的作用是使序号同步。

(2)服务器发送第二个报文，即SYN+ACK报文，其中SYN和ACK标志被置为1。这个报文有两个目的。首先，它是一个用来和对方进行通信的SYN报文。服务器使用这个报文同步初始序号，以便从服务器向客户发送字节。服务器还使用ACK标志确认已从客户端收到了SYN报文，同时给出期望从客户端收到的下一个序号。另外，服务器还定义了客户端要使用的接收窗口的大小。

(3)客户发送第三个报文。这仅仅是一个ACK报文。它使用ACK标志和确认号字段来确认收到了第二个报文。

#### 3.连接终止

通信双方中的任何一方都可以关闭连接。当一方的连接被终止时，另一方还可继续向对方发送数据。TCP的连接终止有两种方式：三次握手和具有半关闭的四次握手。

#### 4.三次握手方式终止连接

使用三次握手的TCP终止过程如下图所示：



图7-6使用三次握手的连接终止

(1)当客户端想关闭TCP连接时，它发送一个TCP报文，把FIN标志位设置为1。

(2)服务器端在收到这个TCP报文后，把TCP连接即将关闭的消息发送给相应的进程，并发送第二个报文——FIN+ACK报文，以证实从客户端收到了FIN报文，同时也说明，另一个方向的连接也关闭了。

(3)客户端发送最后一个报文以证实从TCP服务器收到了FIN报文。这个报文包括确认号，它等于从服务器收到的FIN报文的序号加1。

#### 5.半关闭的四次握手方式终止连接

在TCP连接中，一方可以终止发送数据，但仍然保持接收数据，这就叫做半关闭。半关闭通常是由客户端发起的。图7-7描绘了半关闭的过程。客户发送FIN报文，半关闭了这个连接。服务器发送ACK报文接受这个半关闭。但是，服务器仍然可以发送数据。当服务器已经把所有处理的数据都发送完毕时，就发送FIN报文，客户端发送ACK报文给予确认。

在半关闭一条连接后，客户端仍然可以接收服务器发送的数据，而服务器也可以接收客户端发送的确认。但是，客户端不能传送数据给服务器。

### 六.流量控制

在发送端收到接收端的确认报文之前，流量控制可以对发送端发送的数据量进行管理。

在不考虑流量控制的情况下，传输层协议可以每次只发送一个字节的数据，然后在发送下一个字节数据之前等待接收端的确认报文。这是一个非常缓慢的过程，如果数据要走很长的距离，发送端就要在等待确认报文时一直处在空闲状态。还有一种情况是传输层协议一次就将全部数据发送出去，而不理会确认报文。这样虽然加速了发送过程，但可能会使接收端来不及接收而瘫痪。此外，若有一部分数据丢失、重复、失序或损坏，发送端就要一直等到接收端将全部数据都检查完毕后才能知道。

TCP的流量控制采用一种折中的方法。它在缓存上定义一个窗口。缓存是用来暂时存放将要发送的数据的。TCP发送数据的多少由这个窗口决定。

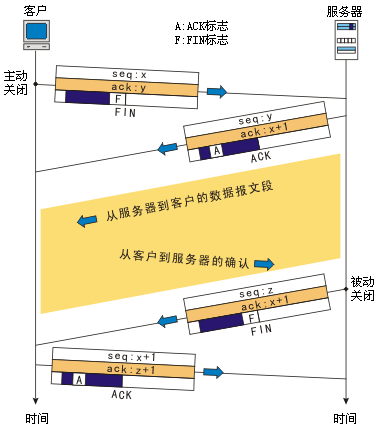


图7-7半关闭

#### 1.滑动窗口协议

为了完成流量控制，TCP使用滑动窗口协议。窗口覆盖了缓存的一部分，在这个窗口中的数据是可以发送而不必考虑确认的。窗口有两个沿：一个在左边，另一个在右边。因为左沿和右沿都是可以滑动的，所以这个窗口叫做滑动窗口。如下图所示：

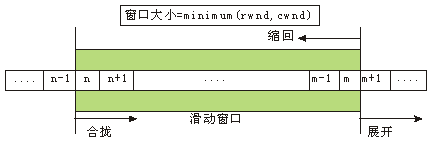


图7-8滑动窗口

窗口有三种动作：展开、合拢或缩回。这三种动作受接收端的控制而不是发送端的控制。

展开窗口表示窗口的右沿向右移动，这样就可以从缓存中发送更多的数据。合拢窗口表示窗口的左沿向右移动，这表示某些数据已经被确认了，发送端可以不再担心它们。缩回窗口表示窗口的右沿向左移动，这在某些实现中是不允许的，因为这会使某些可以发送的数据变成不能发送的。如果发送端已经发送了这些字节，就会产生错误。窗口的左沿不能向左移动，因为这表示已经发送出去的并且经过确认的数据现在又要收回了。

窗口大小由接收窗口和拥塞窗口两者中的较小者决定。接收窗口大小由接收方发送的确认报文中的窗口大小字段值所确定。这是接收端在缓存溢出导致数据被丢弃之前所能接受的最大字节数。拥塞窗口大小是由网络根据拥塞情况而确定的。

### 七.差错控制

TCP是可靠的传输层协议。应用程序把数据流交付给TCP后，就依靠TCP把整个数据流交付给接收端的应用程序，并且保证数据流是按序的、没有差错的、也没有任何一部分是丢失的或重复的。

TCP使用差错控制提供可靠性。差错控制包括以下的一些机制：检测受到损伤的报文、丢失的报文、失序的报文和重复的报文。差错控制还包括检测出差错后纠正差错的机制。TCP的差错检测和差错纠正是通过校验和、确认以及超时重传三种机制实现的。

#### 1.校验和

每一个TCP报文都包括校验和字段，用来检查报文是否损坏。若报文损坏，接收端就将报文丢弃，并认为这个报文丢失了。

#### 2.确认

TCP采用确认报文的方法来证实收到了数据报文。确认报文不携带数据，但消耗一个序号。除了ACK报文之外，确认报文也需要被确认。

#### 3.重传

差错控制的核心是报文的重传机制。当一个报文损坏、丢失或延迟时，就需要重传这个报文。有两种情况需要对报文进行重传：当重传超时计时器时间到期时，或当发送端收到了3个重复的确认报文时。

(1)重传超时计时器到期之后的重传

发送端为每一个TCP报文都设置一个重传超时计时器。若计时器时间到期时还没有收到对这个报文的确认报文，就认为这个报文丢失了，于是重传这个报文，即使可能由于报文延迟到达，或确认报文延迟到达，或确认报文丢失等原因。重传超时计时器的值是动态的，它根据报文的往返时间而更新。报文的往返时间是报文离开发送端到发送端收到此报文的确认报文所需的时间。

(2)三个重复的确认报文之后的重传

一个报文的丢失会导致接收端收到的报文失序，这时接收端会发送对丢失报文的确认报文，当发送端收到3个重复的确认报文之后，发送端立即重传这个报文，这叫做快重传。

对不消耗序号的报文不进行重传。对所有ACK报文都不进行重传。

【实验步骤】

### 练习1察看TCP连接的建立和释放

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B为例，其它组的操作参考主机A、B的操作。

1.主机B启动协议分析器捕获数据，并设置过滤条件（提取TCP协议）。  
主机B在命令行下输入：netstat -a -n命令来查看主机B的TCP端口号。

2.主机A启动TCP工具连接主机B。

主机A启动实验平台工具栏中的“TCP工具”。选中“客户端”单选框，在“地址”文本框中填入主机B的IP地址，在“端口”文本框中填入主机B的一个TCP端口，点击[连接]按钮进行连接。

3.察看主机B捕获的数据，填写下表。

表7-3实验结果

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 报文1 | 报文2 | 报文3 |
| 序列号 |  |  |  |
| 确认号 |  |  |  |
| ACK |  |  |  |
| SYN |  |  |  |

●TCP连接建立时，前两个报文的首部都有一个“最大字段长度”字段，它的值是多少？作用是什么？结合IEEE802.3协议规定的以太网最大帧长度分析此数据是怎样得出的。

4.主机A断开与主机B的TCP连接。

5.察看主机B捕获的数据，填写下表。

表7-4实验结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 报文4 | 报文5 | 报文6 | 报文7 |
| 序列号 |  |  |  |  |
| 确认号 |  |  |  |  |
| ACK |  |  |  |  |
| FIN |  |  |  |  |

●结合步骤3、5所填的表，理解TCP的三次握手建立连接和四次握手的释放连接过程，理解序号、确认号等字段在TCP可靠连接中所起的作用。

思考问题：

1.为什么在TCP连接过程中要使用三次握手？如不这样做可能会出现什么情况。

2.解释TCP协议的释放过程？

## 实验8ICP协议的高级特性

【实验目的】

1.掌握协议栈对TCP协议的处理方法

2.掌握TCP连接建立和释放的实现过程

3.掌握TCP滑动窗口机制

4.理解TCP传输控制块的作用

5.理解TCP的状态转换

【学时分配】

4学时

【实验环境】

该实验采用网络结构一

【实验原理】

### 一.TCP状态转换

要清楚的掌握在连接建立、连接终止以及数据传送时所发生的不同事件，TCP协议以有限状态机的形式来实现。有限状态机是能够经历有限数目状态的一种机器。这个机器总是处于某一个状态中。若没有事件发生，这个机器就一直处于那个状态。事件可使机器转入到一个新的状态，换言之，事件是加到状态上的输入。它可以改变这个状态并产生输出。下表给出了TCP的各种状态。

表8-1TCP的各种状态

| 状态 | 说明 |
| --- | --- |
| CLOSED | 没有连接 |
| LISTEN | 收到了被动打开；等待SYN |
| SYN-SEND | 已发送SYN；等待ACK |
| SYN-RCVD | 已发送SYN+ACK；等待ACK |
| ESTABLISHED | 连接已建立；数据传送在进行 |
| FIN-WAIT-1 | 第一个FIN已发送；等待ACK |
| FIN-WAIT-2 | 对第一个FIN的ACK已收到；等待第二个FIN |
| CLOSE-WAIT | 收到第一个FIN，已发送ACK；等待应用程序关闭 |
| TIME-WAIT | 收到第二个FIN，已发送ACK；等待2MSL超时 |
| LAST-ACK | 已发送第二个FIN；等待ACK |
| CLOSING | 双方都已决定同时关闭 |

这些状态都用椭圆表示。从一个状态通过有向线可转换到另一个状态。每一条连线都注上用斜线隔开的两个字符串。第一个字符串是TCP所收到的输入。第二个字符串是TCP所发送的输出。下图给出了TCP协议的状态转换图，这对客户端和服务器端都适用。图中的虚线表示服务器通常要经过转换，实线表示客户通常要经过的转换。但是在某些情况下，服务器的转换可以经过虚线，而客户的转换可以经过实线。

图8-1TCP状态转换图



### 二.传输控制块

TCP是面向连接的传输协议。一个连接可以打开很长一段时间。为了控制连接，TCP使用一种结构来保持每一条连接的有关信息。这就叫做传输控制块（TCB）。因为在同一时间内可能存在几条TCP连接，TCP就以表的形式保存TCB。这个表的名称也被称作TCB。

一个TCB中包括许多字段，一些常用的字段如下表所示：

表8-2TCB常用字段

| 字段 | 说明 |
| --- | --- |
| 状态 | 这个字段记录TCP状态转换图中定义的连接状态 |
| 进程 | 在这个机器上使用这个连接的进程（作为客户端或服务器） |
| 本地IP地址 | 连接使用的本地机器的IP地址 |
| 本地端口号 | 连接使用的本地端口号 |
| 远程IP地址 | 连接使用的远程机器的IP地址 |
| 远程端口号 | 连接使用的远程端口号 |
| 本地窗口 | 保存本地TCP的窗口信息 |
| 发送序号 | 保存发送序号 |
| 接收序号 | 保存接收序号 |
| 发送ACK号 | 保存发送出的ACK号 |

### 三.TCP连接创建与释放实现代码解析

本实验中所编写的代码只是TCP协议的一个简单处理过程，在编写时同学们不需要考虑差错控制和流量控制，只要能实现TCP协议的基本功能即可。

安装目录JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_tcp\_student\netproto\_tcpconn\_student下的netproto\_tcpconn\_student.h和netproto\_tcpconn\_student.c这两个文件完成了协议栈中TCP连接的创建与释放功能。

netproto\_tcpconn\_student.h文件中的内容并不多，主要是定义了一个宏\_\_TCP\_HALF\_CLOSE，这个宏用于控制TCP连接释放的方式，如果该宏定义为0，则采用三次握手方式关闭TCP连接，反之，如果该宏所定义的值不为0，则采用四次握手方式释放TCP连接。

netproto\_tcpconn\_shudent.c文件是协议栈中TCP连接的创建与释放功能的实现部分，在连接的创建与释放过程中没有数据传输，即当TCP连接建立以后立即释放。其中定义了3个函数。下面介绍这些协议栈的实现部分。

函数netp\_tcpconn\_init\_student将会被主线程调用，用于初始化一个TCP连接。在该函数中主要进行一些初始化工作，例如选择是作为TCP服务器角色进行连接还是作为TCP客户端角色进行连接、源端口号和目的端口号等。值得注意的是，作为TCP服务器的时候，程序会在指定端口进行监听。而作为TCP客户端的时候，该函数会负责向TCP服务器发送一个TCPSYN数据报，从而开始TCP连接过程。该函数中的发送TCPSYN数据报功能需要学生完成。

当有数据到达本机网络接口时，函数netp\_tcpconn\_input\_student将被调用，并传递给这个函数原始数据。在该函数中，需要判断一些条件值来确定接收到的数据包是否为TCP连接所感兴趣的，如果是TCP连接感兴趣的数据报，则提交给netp\_tcp\_input函数处理，如果不是，则返回NETP\_PUSH\_TO\_LWIP交给协议栈继续处理。

netp\_tcp\_input函数实现了TCP连接创建和释放的核心功能。该函数会根据当前的连接状态和收到的事件做相应的操作并切换到后续状态。其状态切换过程如TCP状态转换图所示。

### 四.TCP报文的上层投递实现代码解析

安装目录JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_tcp\_student\netproto\_tcpif\_student下的文件netproto\_tcpif\_student.h和文件netproto\_tcpif\_student.c实现了TCP对高层投递的接口。在netproto\_tcpif\_student.c中定义了8个函数，其中函数netp\_send\_tcp、netp\_tcp\_connect、netp\_tcp\_listen、netp\_tcp\_close和is\_conn\_established为高层协议调用接口。

函数netp\_tcp\_listen用于创建一个TCP连接，并保持在监听状态，这时高层协议充当TCP服务器的角色，当在监听状态时，就可以接收TCP客户端的连接请求。该函数有一个参数为指向用户回调函数的指针，当连接建立并有数据到来时，用户回调函数将被调用处理输入的数据。

函数netp\_tcp\_connect用于连接到一个TCP服务器，即向TCP服务器发送一个TCPSYN报文用于与服务器三次握手。这时高层协议充当TCP客户端的角色。该函数也接受一个用户回调函数的指针参数，其作用与函数netp\_tcp\_listen的用户回调函数参数作用相同。

在TCP连接建立以后，高层协议可以调用netp\_send\_tcp函数发送TCP数据。该函数使用已经建立好的TCP连接将数据发送给目的主机。

函数netp\_tcp\_close用于终止TCP连接。在调用该函数以后，TCP连接将被释放，TCB的状态将变为CLOSED。

netp\_con\_send\_tcp、netp\_tcpif\_input\_student和netp\_tcp\_input这三个函数不能被高层协议调用，他们的作用是维护TCP连接。

函数netp\_tcpif\_input\_student和netp\_tcp\_input的作用与netproto\_tcp\_student.c文件中的netp\_tcpif\_input\_student、netp\_tcp\_input函数作用类似，用于接受被更新TCP连接状态。

netp\_con\_send\_tcp函数发送指定类型的、不带有负载的TCP数据报，用于建立和释放TCP连接，或者发送确认报文。

### 五.滑动窗口实现代码解析

安装目录JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_tcp\_student\netproto\_tcpwindow\_student下的文件netproto\_tcpwindow\_student.h和文件netproto\_tcpwindow\_student.c实现了TCP滑动窗口。在实现中，只对输出缓冲区使用了滑动窗口机制，对输入缓冲区并没有使用滑动窗口机制。

netproto\_tcpwindow\_student.c中的内容与TCP报文的上层投递实现中netproto\_tcpif\_student.c中的内容相近，不同之处在于netproto\_tcpwindow\_student.c的实现中增加了一个全局的发送缓冲区以及一些滑动窗口控制变量。

netp\_send\_buff是一个全局数组，类型为u8\_t，长度为宏SEND\_BUFF\_SIZE所定义的数值，SEND\_BUFF\_SIZE是在一次TCP连接中能传送的最大字节数。

全局变量wnd\_left、wnd\_right、wnd\_next、recv\_wnd\_size和send\_wnd\_size共同维护TCP滑动窗口，wnd\_left记录滑动窗口左沿，wnd\_right记录滑动窗口右沿，wnd\_next记录下一个可发送的字节，recv\_wnd\_size记录接收滑动窗口大小，send\_wnd\_size记录发送滑动窗口大小。send\_wnd\_size的默认值为64，值得注意的是，在实际应用中，发送滑动窗口的大小会远远大于这个值，在这里使用一个较小的值是为了更方便的观察滑动窗口的变化。

函数netp\_send\_tcp、netp\_con\_send\_tcp、netp\_tcp\_connect、netp\_tcp\_listen、netp\_tcp\_close、netp\_tcpwindow\_input\_student、netp\_tcp\_input、is\_conn\_established与netproto\_tcpif\_student.c文件即TCP报文的上层投递实现中相应函数实现了类似的功能，这里就不再讲述。

另外一个函数netp\_set\_wndsize用于设置接收窗口大小。

### 六.实验中涉及的结构体、宏和函数

在编码过程中可能会涉及到一些结构体、宏和函数，下表是对他们进行和介绍：

表8-3实验涉及的结构体、宏和函数

| 结构体/宏/函数 | 声明或定义 | 描述 |
| --- | --- | --- |
| structnetp\_eth\_header | structnetp\_eth\_header{  u8\_tdest\_address[ETH\_ADDRESS\_LEN];  u8\_tsour\_address[ETH\_ADDRESS\_LEN];  u16\_ttype;  }; | 以太网帧头结构 |
| structnetp\_ip\_header | structnetp\_ip\_header{  u8\_theaderlen:4;  u8\_tversion:4;  u8\_tdiff\_services;  u16\_ttotal\_length;  u16\_tidentification;  u16\_tflags\_offset;  u8\_ttime\_to\_live;  u8\_tprotocol;  u16\_theader\_checksum;  structip\_addrsource\_address;  structip\_addrdestination\_address;  }; | ipv4包头结构 |
| structnetp\_tcp\_header | structnetp\_tcp\_header{  u16\_tsour\_port;  u16\_tdest\_port;  u32\_tseq\_num;  u32\_tack\_num;  u8\_treserved:4;  u8\_tdata\_offset:4;  u8\_ttcp\_flag;  u16\_twindow;  u16\_tchecksum;  u16\_turgent\_ptr;  u32\_topt\_padd;  }; | TCP报文头结构 |
| structnetp\_tcp\_pseudo | structnetp\_tcp\_pseudo{  u32\_tsour\_addr;  u32\_tdest\_addr;  u8\_tzero;  u8\_tprotocol;  u16\_tlength;  }; | TCP伪首部结构 |
| structin\_addr | structin\_addr{  u32\_ts\_addr;  }; | 32位地址 |
| TCP\_HEADER\_LEN | #defineTCP\_HEADER\_LEN24 | TCP数据报头长度 |
| NETP\_TCP\_PSEUDO\_LEN | #defineNETP\_TCP\_PSEUDO\_LEN  sizeof(structnetp\_tcp\_pseudo) | TCP伪首部结构长度 |
| NETP\_TCP\_CONN\_SER | #defineNETP\_TCP\_CONN\_SER1 | 作为TCP服务器 |
| NETP\_TCP\_CONN\_CLI | #defineNETP\_TCP\_CONN\_CLI2 | 作为TCP客户端 |
| TCP\_FLAG\_FIN | #defineTCP\_FLAG\_FIN0x01 | TCPFIN标志 |
| TCP\_FLAG\_SYN | #defineTCP\_FLAG\_SYN0x02 | TCPSYN标志 |
| TCP\_FLAG\_RST | #defineTCP\_FLAG\_RST0x04 | TCPRST标志 |
| TCP\_FLAG\_PSH | #defineTCP\_FLAG\_PSH0x08 | TCPPSH标志 |
| TCP\_FLAG\_ACK | #defineTCP\_FLAG\_ACK0x10 | TCPACK标志 |
| TCP\_FLAG\_URG | #defineTCP\_FLAG\_URG0x20 | TCPURG标志 |
| TCP\_FLAG\_ECE | #defineTCP\_FLAG\_ECE0x40 | TCPECE标志 |
| TCP\_FLAG\_CWR | #defineTCP\_FLAG\_CWR0x80 | TCPCWR标志 |
| PAYLOAD\_DATA | #definePAYLOAD\_DATA"Hello,World!" | TCP负载内容 |
| PAYLOAD\_LEN | #definePAYLOAD\_LENsizeof(PAYLOAD\_DATA) | TCP负载长度 |
| NETP\_UDP\_PSEUDO\_LEN | #defineNETP\_UDP\_PSEUDO\_LEN  sizeof(structnetp\_udp\_pseudo) | TCP伪首部长度 |
| ETH\_HEADER\_LEN | #defineETH\_HEADER\_LEN14 | 以太网帧头长度 |
| IP\_HEADER\_LEN | #defineIP\_HEADER\_LEN20 | IP包头长度 |
| netp\_current\_udp\_addr | u32\_tnetp\_current\_udp\_addr(); | 获取当前正在使用的网络接口的IP地址 |
| htons | u16\_thtons(u16\_tn); | 将16位数值由主机字节序转换为网络字节序 |
| htonl | u32\_thtonl(u32\_tn) | 将32位数值由主机字节序转换为网络字节序 |
| inet\_addr | u32\_tinet\_addr(constchar\*cp); | 将ASCII编码的Internet地址转换成为网络字节序地址 |
| inet\_chksum | u16\_tinet\_chksum(void\*dataptr,u16\_tlen); | 计算校验和 |
| do\_sleep | voiddo\_sleep(intms); | 在指定的时间内挂起当前线程 |
| netp\_ip\_payload\_output | err\_tnetp\_ip\_payload\_output(  structip\_addr\*src,structip\_addr\*dest,  u8\_tttl,u8\_tds,u8\_tproto,void\*buffer,  intbuff\_len,intadapterID); | 发送IP数据包接口函数，不需要构造IP头部 |

【实验步骤】

### 练习1TCP连接建立和释放

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。实验开始前，先单击“初始环境”。

在实验中主机的调试接口IP地址分别设为172.16.0.1n1、172.16.0.1n2、172.16.0.1n3、172.16.0.1n4、172.16.0.1n5、172.16.0.1n6(n为组别号，目的是为了避免IP地址冲突)，所有主机使用子网掩码255.255.255.0，默认网关设置为0.0.0.0。

以第一组为例，主机A将调试接口的IP地址设置为172.16.0.111、主机B使用处于连接状态的物理接口，将调试接口的IP设置为172.16.0.112、主机C将调试接口的IP地址设置为172.16.0.113、主机D使用处于连接状态的物理接口，将调试接口的IP地址设置为172.16.0.114、主机E使用处于连接状态的物理接口，将调试接口的IP地址设置为172.16.0.115、主机F将新接口的IP地址设置为172.16.0.116。所有主机使用子网掩码255.255.255.0，默认网关设置为0.0.0.0。

1. 阅读已存在代码

TCP协议的实现相对比较复杂，因此在编写实现代码之前，请同学们根据TCP状态转换图所示的TCP状态转换关系阅读已经存在的代码，尤其是函数netp\_tcp\_input的实现代码，理解TCP协议的工作机制。

2. 设置TCP连接终止方式

各主机使用VS2005软件打开实验平台安装目录下的JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_tcp\_student\netproto\_tcp\_student.sln文件，在头文件netproto\_tcpconn\_student.h内将宏\_\_TCP\_HALF\_CLOSE所定义的值修改为1，从而设定TCP连接采用四次握手方式终止。

注：若实验平台安装在C盘，则目录为：C:\Program Files\JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_tcp\_student\netproto\_tcp\_student.sln。

3. 编码实现TCP SYN报文的发送

(1) 当已经存在的代码阅读完毕以后，各主机使用VS2005软件打开实验平台安装目下的JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_tcp\_student\netproto\_tcp\_student.sln文件，在源函数netproto\_tcpconn\_student.c的函数netp\_tcpconn\_init\_student中添加TCP SYN报文的发送。

(2) 构造TCP报头结构。注意，确认号设置为0，标识为TCP\_FLAG\_SYN。

(3) 构造TCP伪首部结构。长度字段值为htons(TCP\_HEADER\_LEN)。

(4) 计算TCP校验和。使用inet\_chksum函数计算TCP校验和。

4. 所有主机打开协议分析器，开始捕获数据

5. 所有主机调试并运行程序

主机A充当TCP服务器角色，主机B充当TCP客户端角色，同组同学协商端口号进行连接。

6. 各主机停止数据捕获，观察实验现象

7. 各主机将\_\_TCP\_HALF\_CLOSE所定义的值修改为0，重新进行实验

● 两次连接断开时的方式有什么不同？

8.发送TCPSYN报文的参考代码如下：

//发送SYN报文  
//在Listen状态下收到syn报文，发送syn+ack  
printf("向%s(端口:%d)发送连接请求。\n",  
inet\_ntoa(\*(structin\_addr\*)&tcp\_conn\_pcb.remote\_ip),  
ntohs(tcp\_conn\_pcb.remote\_port));  
  
req\_tcp\_header.dest\_port=tcp\_conn\_pcb.remote\_port;  
req\_tcp\_header.sour\_port=tcp\_conn\_pcb.local\_port;  
req\_tcp\_header.seq\_num=htonl(tcp\_conn\_pcb.next\_seqno++);  
req\_tcp\_header.ack\_num=0;  
req\_tcp\_header.data\_offset=TCP\_DATA\_OFFSET;  
req\_tcp\_header.reserved=0;  
req\_tcp\_header.tcp\_flag=TCP\_FLAG\_SYN;  
req\_tcp\_header.window=tcp\_conn\_pcb.window;  
req\_tcp\_header.checksum=0;  
req\_tcp\_header.urgent\_ptr=0;  
  
req\_tcp\_pseudo.sour\_addr=tcp\_conn\_pcb.local\_ip.addr;  
req\_tcp\_pseudo.dest\_addr=tcp\_conn\_pcb.remote\_ip.addr;  
req\_tcp\_pseudo.zero=0;  
req\_tcp\_pseudo.protocol=IP\_PROTO\_TCP;  
req\_tcp\_pseudo.length=htons(TCP\_HEADER\_LEN);  
  
memcpy(check\_buff,&req\_tcp\_pseudo,NETP\_TCP\_PSEUDO\_LEN);  
memcpy(check\_buff+NETP\_TCP\_PSEUDO\_LEN,&req\_tcp\_header,  
TCP\_HEADER\_LEN);  
  
req\_tcp\_header.checksum=inet\_chksum(&check\_buff,  
NETP\_TCP\_PSEUDO\_LEN+TCP\_HEADER\_LEN);  
  
if(ERR\_OK==netp\_ip\_payload\_output(&tcp\_conn\_pcb.local\_ip,  
&tcp\_conn\_pcb.remote\_ip,tcp\_conn\_pcb.ttl,tcp\_conn\_pcb.ds,  
IP\_PROTO\_TCP,&req\_tcp\_header,TCP\_HEADER\_LEN,0)){  
tcp\_conn\_pcb.state=TCP\_SYN\_SENT;  
}

### 练习2TCP报文的上层投递的设计与实现

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。实验开始前，先单击“初始环境”。

在实验中主机的调试接口IP地址分别设为172.16.0.1n1、172.16.0.1n2、172.16.0.1n3、172.16.0.1n4、172.16.0.1n5、172.16.0.1n6(n为组别号，目的是为了避免IP地址冲突)，所有主机使用子网掩码255.255.255.0，默认网关设置为0.0.0.0。

以第一组为例，主机A将调试接口的IP地址设置为172.16.0.111、主机B使用处于连接状态的物理接口，将调试接口的IP设置为172.16.0.112、主机C将调试接口的IP地址设置为172.16.0.113、主机D使用处于连接状态的物理接口，将调试接口的IP地址设置为172.16.0.114、主机E使用处于连接状态的物理接口，将调试接口的IP地址设置为172.16.0.115、主机F将调试接口的IP地址设置为172.16.0.116。所有主机使用子网掩码255.255.255.0，默认网关设置为0.0.0.0。

1. 阅读已存在代码

同学们根据TCP状态转换图所示的TCP状态转换关系阅读已经存在的代码。

分析函数netp\_tcp\_input的实现代码与练习一中的函数netp\_tcp\_input相比较有什么不同之处。

2. 构造TCP头结构

各主机使用已经定义的变量tcp\_header编码构造TCP头结构，可以在tcp\_conn\_pcb中查找相应的数值。

3. 构造TCP伪首部

各主机使用已经定义的变量tcp\_pseudo构造TCP伪首部。在伪首部中，总长度字段值应该为htons(TCP\_HEADER\_LEN+buff\_len)。

4. 计算TCP校验和

各主机编码实现TCP校验和的计算，可以使用inet\_chksum函数。

5. 所有主机打开协议分析器，开始捕获数据

6. 所有主机调试并运行程序

(1) 主机A充当TCP服务器角色，主机B充当TCP客户端角色，同组同学协商端口号进行连接。

(2) 当TCP连接建立以后，双方发送消息。

● 你收到的信息是什么？

(3) 输入quit关闭TCP连接。

7. 各主机停止数据捕获，观察实验现象

8.函数netp\_send\_tcp的参考代码如下：

/\*\*  
\*\brief利用创建的tcp连接发送tcp数据  
\*  
\*\parambuff负载缓冲区  
\*\buff\_len负载长度  
\*  
\*\returnERR\_OK发送成功  
\*/  
err\_t  
netp\_send\_tcp(void\*buff,intbuff\_len)  
{  
structnetp\_tcp\_headertcp\_header;  
structnetp\_tcp\_pseudotcp\_pseudo;  
u8\_tcheck\_buff[TCP\_HEADER\_LEN+NETP\_TCP\_PSEUDO\_LEN+PAY\_BUFF\_LEN];  
u8\_tsend\_buff[TCP\_HEADER\_LEN+PAY\_BUFF\_LEN];  
  
//构造TCP头结构  
tcp\_header.sour\_port=tcp\_conn\_pcb.local\_port;  
tcp\_header.dest\_port=tcp\_conn\_pcb.remote\_port;  
tcp\_header.seq\_num=htonl(tcp\_conn\_pcb.next\_seqno);  
tcp\_header.ack\_num=htonl(tcp\_conn\_pcb.ack);  
tcp\_header.reserved=0;  
tcp\_header.data\_offset=TCP\_DATA\_OFFSET;  
tcp\_header.tcp\_flag=TCP\_FLAG\_ACK;  
tcp\_header.window=htons(tcp\_conn\_pcb.window);  
tcp\_header.checksum=0;  
tcp\_header.urgent\_ptr=0;  
tcp\_header.opt\_padd=0;  
  
//构造TCP伪首部结构  
tcp\_pseudo.sour\_addr=tcp\_conn\_pcb.local\_ip.addr;  
tcp\_pseudo.dest\_addr=tcp\_conn\_pcb.remote\_ip.addr;  
tcp\_pseudo.zero=0;  
tcp\_pseudo.protocol=IP\_PROTO\_TCP;  
tcp\_pseudo.length=htons(TCP\_HEADER\_LEN+buff\_len);  
  
//计算TCP校验和  
memcpy(check\_buff,&tcp\_pseudo,NETP\_TCP\_PSEUDO\_LEN);  
memcpy(check\_buff+NETP\_TCP\_PSEUDO\_LEN,&tcp\_header,TCP\_HEADER\_LEN);  
memcpy(check\_buff+NETP\_TCP\_PSEUDO\_LEN+TCP\_HEADER\_LEN,  
buff,buff\_len);  
tcp\_header.checksum=inet\_chksum(check\_buff,  
NETP\_TCP\_PSEUDO\_LEN+TCP\_HEADER\_LEN+(u16\_t)buff\_len);  
  
//将TCP头结构、TCP负载拷贝到发送缓冲区  
memcpy(send\_buff,&tcp\_header,TCP\_HEADER\_LEN);  
memcpy(send\_buff+TCP\_HEADER\_LEN,buff,buff\_len);  
tcp\_conn\_pcb.next\_seqno+=buff\_len;  
  
//如果TCP连接已经建立，使用IP接口发送  
if(TCP\_ESTABLISHED==tcp\_conn\_pcb.state){  
returnnetp\_ip\_payload\_output(&tcp\_conn\_pcb.local\_ip,&tcp\_conn\_pcb.remote\_ip,  
tcp\_conn\_pcb.ttl,tcp\_conn\_pcb.ds,IP\_PROTO\_TCP,send\_buff,  
TCP\_HEADER\_LEN+buff\_len,0);  
}else{  
 returnERR\_CONN;  
}

}

### 练习3TCP滑动窗口协议的设计与实现

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。实验开始前，先单击“初始环境”。

在实验中主机的调试接口IP地址分别设为172.16.0.1n1、172.16.0.1n2、172.16.0.1n3、172.16.0.1n4、172.16.0.1n5、172.16.0.1n6(n为组别号)，所有主机使用子网掩码255.255.255.0，默认网关设置为0.0.0.0。

以第一组为例，主机A将调试接口的IP地址设置为172.16.0.111、主机B使用处于连接状态的物理接口，将调试接口的IP设置为172.16.0.112、主机C将调试接口的IP地址设置为172.16.0.113、主机D使用处于连接状态的物理接口，将调试接口的IP地址设置为172.16.0.114、主机E使用处于连接状态的物理接口，将调试接口的IP地址设置为172.16.0.115、主机F将调试接口的IP地址设置为172.16.0.116。所有主机使用子网掩码255.255.255.0，默认网关设置为0.0.0.0。

1. 阅读已存在代码

同学们根据TCP状态转换图所示的TCP状态转换关系和TCP滑动窗口的相关原理，阅读已经存在的代码。

2. 更新TCP滑动窗口

各主机使用VS2005软件打开实验平台安装目录下的JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_tcp\_student\netproto\_tcp\_student.sln文件，在源函数netproto\_tcpwindow\_studnet.c的函数netp\_update\_wnd中进行编辑。根据接收到的ACK报文内容和当前滑动窗口信息，编写代码实现TCP滑动窗口的更新。

注：若实验平台安装在C盘，则目录为：C:\Program Files\JLCSS\ExpCNC\Work\NPL\ExpNPL\_student\netproto\_tcp\_student\netproto\_tcp\_student.sln。

3. 所有主机打开协议分析器，开始捕获数据

4. 所有主机调试并运行程序

(1) 主机A充当TCP服务器角色，主机B充当TCP客户端角色，同组同学协商端口号进行连接。

(2) 主机A、B设置接收窗口大小值，这个值应该比较小。

(3) 当TCP连接建立以后，双方发送消息。

● 你收到的信息是什么？

● 你的发送窗口大小是什么？这个值是由谁决定的？

(4) 输入quit关闭TCP连接。

5. 各主机停止数据捕获，观察实验现象

6.函数netp\_update\_wnd的参考代码如下：

/\*\*  
\*\brief更新滑动窗口  
\*/  
void  
netp\_update\_wnd()  
{  
if(wnd\_next!=ntohl(tcp\_conn\_pcb.remote\_ack)){  
printf("\n收到%s发送的ACK报文，确认号为：%d，需要调整滑动窗口。\n",  
inet\_ntoa(\*(structin\_addr\*)&tcp\_conn\_pcb.remote\_ip),  
ntohl(tcp\_conn\_pcb.remote\_ack));  
  
//更新滑动窗口左沿，右沿和下一个可发送字节位置  
wnd\_next=ntohl(tcp\_conn\_pcb.remote\_ack);  
wnd\_left=wnd\_next;  
wnd\_right=  
(wnd\_left+send\_wnd\_size<SEND\_BUFF\_SIZE)?  
wnd\_left+send\_wnd\_size:SEND\_BUFF\_SIZE;  
  
if(SEND\_BUFF\_SIZE==wnd\_right){  
printf("滑动窗口右沿已经到达发送缓冲区末尾，不能继续移动\n");  
wnd\_left=0;  
wnd\_right=0;  
return;  
}  
  
printf("\n滑动窗口左沿调整为：\t%d\n滑动窗口右沿调整为：\  
 \t%d\n下一个可发送字节位置：\t%d\n",  
wnd\_left,wnd\_right,wnd\_next);  
}  
}

## 实验9简单网络管理协议(SNMP)

【实验目的】

1.掌握SNMP的报文格式

2.掌握SMI定义的规则

3.掌握MIB定义的结构

4.理解SNMP工作原理

【学时分配】

2学时

【实验环境】

该实验采用网络结构一

【实验原理】

### 一.SNMP简介

SNMP（简单网络管理协议）是专门用于在IP网络中管理网络节点的一种标准协议。它用于在SNMP代理和SNMP管理器之间传送管理信息。SNMP使网络管理员能够管理网络，发现并解决网络问题以及规划网络增长。

### 二.SNMP报文格式

SNMP的5种报文，它们封装在UDP数据报中，它们都有公共SNMP首部，然后是不同的PDU（其中get，get-next，set的PDU部分是相同的）。管理进程发出的get，get-next，set操作采用UDP端口161，代理进程发出的trap操作采用UDP的162端口。另外，SNMP报文的编码采用ASN.1和BER。如下图所示。

公共SNMP首部格式如下：

|  |  |  |
| --- | --- | --- |
| 版本 | 共同体 | PDU类型 |

get，get-next，setPDU格式如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 请求标识 | 0 | 0 | 变量绑定 |

get-responsePDU格式如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 请求标识 | 差错状态 | 差错索引 | 变量绑定 |

trapPDU格式如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 企业 | 代理地址 | Trap类型 | 特定代码 | 时间戳 | 变量绑定 |

变量绑定字段格式如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称1 | 值1 | 名称2 | 值2 |  |

图9-1SNMP报文格式

●版本：SNMP版本，值为0，是通过SNMP版本号减去1得到的。

●共同体：SNMP代理和一些SNMP应用程序实体的任意集合的组合。是一个字符串，默认值为public。

●PDU类型：PDU类型如下表所示：

表9-1PDU类型

|  |  |
| --- | --- |
| PDU类型 | 名称 |
| 0 | get |
| 1 | get-next |
| 2 | get-response |
| 3 | set |
| 4 | trap |

●请求标识：通过给每个请求提供一个独有的ID来区分处理的请求。

●变量绑定：变量名称和相应取值的列表。

●差错状态：用于表示在处理请求时出现的异常。

●差错索引：当差错状态是非零时，可能由列表中导致异常的变量来提供附加的信息。

●企业：产生trap的对象的类型。

●代理地址：产生trap的对象的地址。

●Trap类型：通用trap类型。

●特定代码：具体trap代码。

●时间戳：上次初始化网络实体和产生trap之间的所持续的时间。

### 三.SNMP管理器和代理

SNMP使用管理器和代理的概念。管理器（通常是主机）控制和监视一组代理（通常是路由器）。如下图所示：

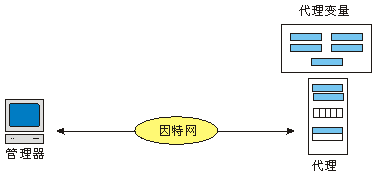


图9-2管理器和代理

管理器是运行SNMP客户程序的主机。代理是运行SNMP服务器程序的路由器（或主机）。管理是通过在管理器和代理之间的简单交互来实现的。

代理在数据库中保存了性能信息。管理器可以使用这个数据库中数值。例如，路由器可以把已收到的和已转发的数据包数存储成适当的变量。管理器可以读取和比较这两个变量，以便发现路由器是否拥塞。

管理器还可以使路由器完成某些动作。例如，路由器定期地检查重新引导计数器的值，看它何时应当重新引导自己。当计数器的值为0时就应当重新引导自己。管理器可以随时使用这个特性从远程重新引导这个代理。它只要发送一个数据包，迫使这个计数器的值为0即可。

代理也可以参加到管理过程中。在代理上运行的服务器程序可以检查环境，若发现有异常现象可以向管理器发送告警报文（叫做陷阱trap）。

总之，使用SNMP的管理是基于以下3个基本思想：

●管理器检查代理的方法是发出请求能够反映代理的行为的信息。

●管理器可以用重新设置在代理数据库中的某些值的方法，强迫代理完成某个任务。

●代理参与管理过程的方法是向管理器发出对异常情况的告警。

### 四.SNMP管理构件

为了完成管理任务，SNMP使用另外两个协议：管理信息结构（SMI）和管理信息库（MIB）。如下图所示：

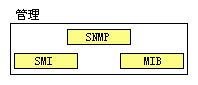


图9-3SNMP管理构件

#### 1.SNMP的作用

SNMP在网络管理中起着非常特殊的作用。它定义了从管理器发送到代理以及从代理发送到管理器的数据包格式。它还解释结果和产生统计。所交换的数据包包含对象（变量）名和它们的状态（值）。SNMP负责读取和改变这些数值。

#### 2.SMI的作用

要使用SNMP，就需要命名对象的规则和定义对象类型的规则。SMI是定义这些规则的协议。SMI只是定义了这些规则，它并没有定义在一个实体中可以管理多少个对象或哪个对象使用哪一种类型。SMI是许多通用规则的集合，这些规则用来命名对象和列出它们的类型清单。对象和类型的关联并不是SMI应当做的事。

#### 3.MIB的作用

MIB协议定义了对象的数目，按照SMI定义的规则给这些对象命名，并且将对象和一种类型联系起来。

### 五.管理信息结构SMI

#### 1.SMI

SMIv2（管理信息结构版本2）是用在网络管理中的一个构件。它的功能是：

●定义给对象命名的规则。

●定义可在对象中存储的数据类型。

●给出如何对网络上传输的数据进行编码的方法。

SMI是SNMP的一个指南。它强调处理对象的3个属性：名字、数据类型和编码方法，如下图所示：

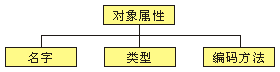


图9-4对象属性

#### 2.名字

SMI要求一个被管对象具有惟一的名字。为了在全局给对象命名，SMI使用对象标识符，它是基于树结构的分层次的标识符。如下图所示：

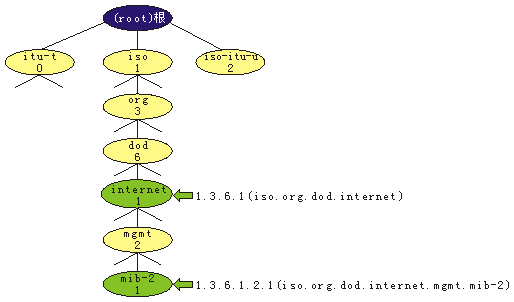


图9-5对象标识符

树结构从不命名的根开始。每一个对象可用点分隔开的整数序列定义。这种树结构也可以使用点分隔开的文本名字序列来定义对象。名字——点的表示方法是用户使用的。

SNMP使用的各对象都位于mib-2对象下面的，它们的标识符永远从1.3.6.1.2.1开始。

#### 3.类型

对象的第二个属性是对象中存储的数据的类型。SMI使用ASN.1（抽象语法1）的基本定义定义了数据类型，同时也增加了几个新的类型。SMI既是ASN.1的一个子集，也是ASN.1的一个超集。

SMI使用两大数据类型：简单的和结构化的。

(1)简单类型

简单数据类型是原子数据类型。这些类型中的一些是直接取自ASN.1的；另一些是SMI增加的。下表给出了一些重要的类型，前5个取自ASN.1；后7个是SMI定义的。

表9-2SMI简单类型

| 类型 | 大小 | 说明 |
| --- | --- | --- |
| INTEGER | 4字节 | 在－～－1之间的整数 |
| Interger32 | 4字节 | 和INTEGER相同 |
| Unsigned32 | 4字节 | 在0～－1之间的无符号数 |
| OCTETSTRING | 可变 | 不超过65535字节长的字节串 |
| OBJECTIDENTIFIER | 可变 | 对象标识符 |
| IPAddress | 4字节 | 由4个整数组成的IP地址 |
| Counter32 | 4字节 | 可从0增加到的整数；当它到达最大值时就返回到0 |
| Counter64 | 8字节 | 64位的计数器 |
| Gauge32 | 4字节 | 与COUNTER32相同，但当它到达最大值时不返回；它保持在这个数值到复位 |
| TimrTicks | 4字节 | 记录时间的计数值，以1/100s为单位 |
| BITS |  | 位串 |
| Opaque | 可变 | 不解释的串 |

(2)结构化类型

把简单的和结构化的数据类型组合起来，就可以构成新的结构化数据类型。SMI定义了两种结构化数据类型：sequence和sequenceof。

Sequence：sequence数据类型是一些简单数据类型的组合，但不必都是相同的类型。它和C语言中使用的struct或record的概念相似。

Sequenceof：sequenceof数据类型是所有相同类型的简单数据类型的组合，或所有相同类型的sequence数据类型的组合。它和C语言中使用的array的概念相似。

#### 4.编码方法

SMI编码方法使用BER（基本编码规则），把数据编码后在网络上传输。BER将每一块数据编码成三元组格式：标记、长度和值，如下图所示：

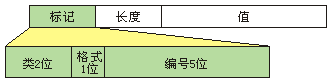


图9-6编码格式

(1)标记

标记是定义数据类型的1字节字段。它由3个子字段组成：类字段（2位）、格式字段（1位）和编号字段（5位）。类字段定义数据的作用域（scope）。共定义了4个类：通用类（00）、应用类（01）、特定上下文类(10)和专用类(11)。通用数据类型是来自ASN.1的。应用数据类型是由SMI增加的。有5种特定上下文数据类型，它们的意义随着协议的不同而不同。专用数据类型是特定厂商使用的。格式字段指出数据是简单的（0）还是结构化的(1)。编号字段将简单的或结构化的数据进一步划分为一些子组（subgroup）。例如在通用类的简单格式，INTEGER的值是2，OCTETSTRING的值是4，等等。

(2)长度

长度字段是1或多字节。若它是1字节，则最高位必定为0，其余的7位定义数据长度。若大于1字节，则第一字节的最高位必定为1。第一个字节的其余7位则定义所需的字节数。下图说明了长度字段的意义：

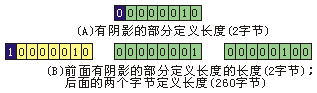


图9-7长度的格式

(3)值

值字段按照在BER中定义的规则把数据的值进行编码。

### 六.管理信息库MIB

MIB2（管理信息库版本2）是网络管理中的第二个构件。每一个代理有它自己的MIB2，这是管理器能够管理的所有对象的集合。在MIB2中的对象分成10个不同的组：system（系统）、interface（接口）、addresstranslation（地址转换）、ip、icmp、tcp、udp、egp、transmission（传输）和snmp。这些组都在对象标识符树中mib-2对象的下面，如下图所示。每一个组定义一些变量和表。

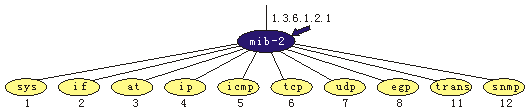


图9-8MIB

下面是一些对象的简单描述。

sys：系统定义有关这个节点（系统）的一般信息，如名字、位置和寿命。

if：接口定义有关这个节点的所有接口的信息，如接口号、物理地址和IP地址。

at：地址转换定义有关ARP表的信息。

ip：这个对象定义有关IP的信息，如路由表和IP地址。

icmp：这个对象定义有关ICMP的信息，如已发送和已收到的数据包数，以及产生的差错总数。

tcp：这个对象定义有关TCP的信息，如连接表、超时值、端口号，以及已发送和已收到的数据包数。

udp：这个对象定义有关UDP的信息，如端口号、已发送和已收到的数据包数。

snmp：这个对象定义有关SNMP本身的信息。

#### 1.访问MIB变量

下面以udp组作为例子来说明如何访问不同的变量。在udp组有4个简单变量和一个记录序列（表）。下图给出了这些变量和表。



图9-9udp组

下面说明如何访问每一个实体。

访问简单变量时使用组的id（1.3.6.1.2.1.7）后面跟着该变量的id。下面给出了如何访问每一个变量。

udpInDatagrams->1.3.6.1.2.1.7.1

udpNoPorts->1.3.6.1.2.1.7.2

udpInErrors->1.3.6.1.2.1.7.3

udpOutDatagrams->1.3.6.1.2.1.7.4

但是，这些对象标识符定义的是变量而不是内容。要给出每一个变量的内容，应该增加内容的后缀。简单变量的内容后缀就是零。例如：

udpInDatagrams->1.3.6.1.2.1.7.1.0

udpNoPorts->1.3.6.1.2.1.7.2.0

udpInErrors->1.3.6.1.2.1.7.3.0

udpOutDatagrams->1.3.6.1.2.1.7.4.0

MIB使用表id标志表。如下图所示，udp只有一个表（id是5）。

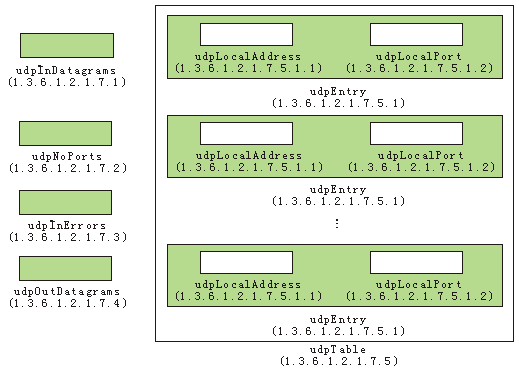


图9-10udp的变量和表

要访问这个表，应该使用如下的表id：

udpTable->1.3.6.1.2.1.7.5

但是，这个表还是在树结构的树叶级。因此不能访问这个表，下面定义在这个表（id是1）中的项目：

udpEntry->1.3.6.1.2.1.7.5.1

这个项目也不是树叶，因此不能访问它。还需要定义这个项目中的每一个实体（字段）。

udpLocalAddress->1.3.6.1.2.1.7.5.1.1

udpLocalPort->1.3.6.1.2.1.7.5.1.2

这时两个变量都是在树叶上了。这个表对每一个“本地地址/本地端口对”可以有不同的值。要访问表中的特定行，应当给以上的id加上索引。在MIB中，数组的索引不是整数。这些索引是基于在这些项目中的一个或多个字段的值。在例子中，udpTable的索引基于本地地址和本地端口号。

### 七.SNMP通信过程

SNMP在两个熟知端口161和162上使用UDP的服务。熟知端口161由代理使用，而熟知端口162由管理器使用。

代理在端口161上发出被动打开。然后它就等待从管理器来的连接。管理器使用短暂端口发出主动打开。客户向服务器发送请求报文，使用短暂端口作为源端口而熟知端口161作为目的端口。服务器向客户发送响应报文，使用熟知端口161作为源端口而短暂端口作为目的端口。

管理器在端口162发出被动打开。然后它就等待从代理来的连接。代理只要有Trap报文要发送，就使用短暂端口发出主动打开。这个连接只是单向的，从服务器到客户，如下图所示：

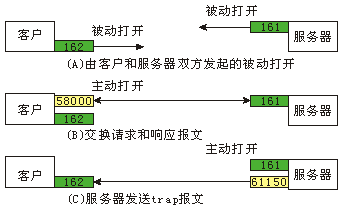


图9-11SNMP的端口号

在SNMP中的客户/服务器机制与其它协议的不同。这里的客户和服务器都使用熟知端口。此外，客户和服务器都必须无限制地运行下去。这个原因就是请求报文是由管理器发出的，但Trap报文则是由代理发出的。

【实验步骤】

### 练习1获取代理服务器信息

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。

其中主机B作为SNMP代理服务器，主机A作为SNMP管理器。主机B在实验前应当通过管理工具中的“服务”工具查看是否已经安装了名为“SNMPService”服务，如果没有安装，应当参考实验附录C完成“SNMPService”服务的安装。

1.主机B启动SNMP服务，并创建具有“只读”权利的团体“public”接受来自任何主机的SNMP数据包。配置方法如下：

(1)启动“服务”管理器，找到“控制面板/管理工具/服务”程序，双击启动。

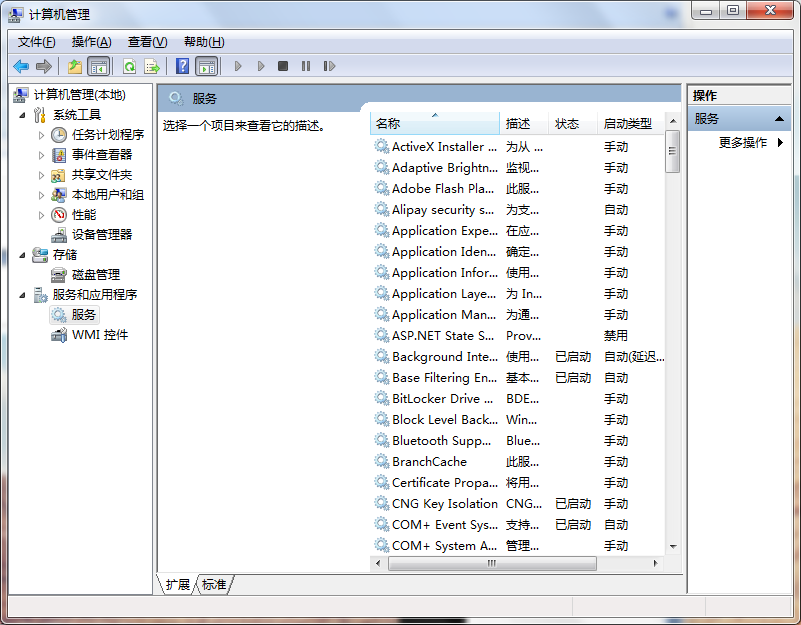


图9-12服务

(2)启动“SNMPService”和“SNMPTrap”。

①在服务程序列表中找到“SNMPService”和“SNMPTrap”。

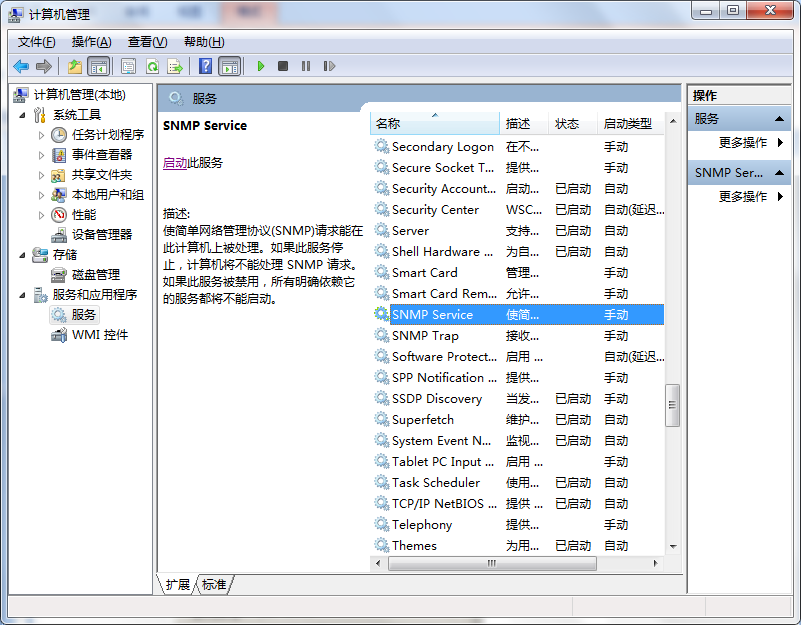


图9-13SNMP服务

②选中“SNMPService”条目，单击鼠标右键，选择“属性”菜单项，并修改“启动类型”为“手动”，点击[确定]按钮保存设置。

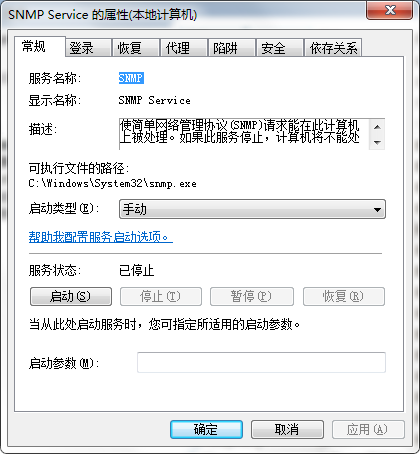


图9-14更改服务属性

③单击“服务”管理器菜单栏上的[启动]按钮启动服务。

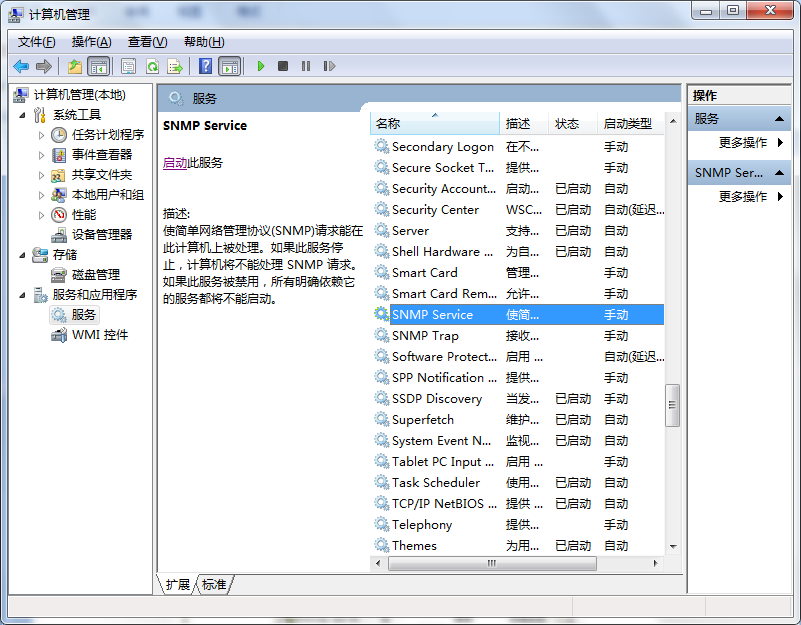


图9-15启动服务

④按同样的方法启动“SNMPTrap”。启动后的状态如下图所示：

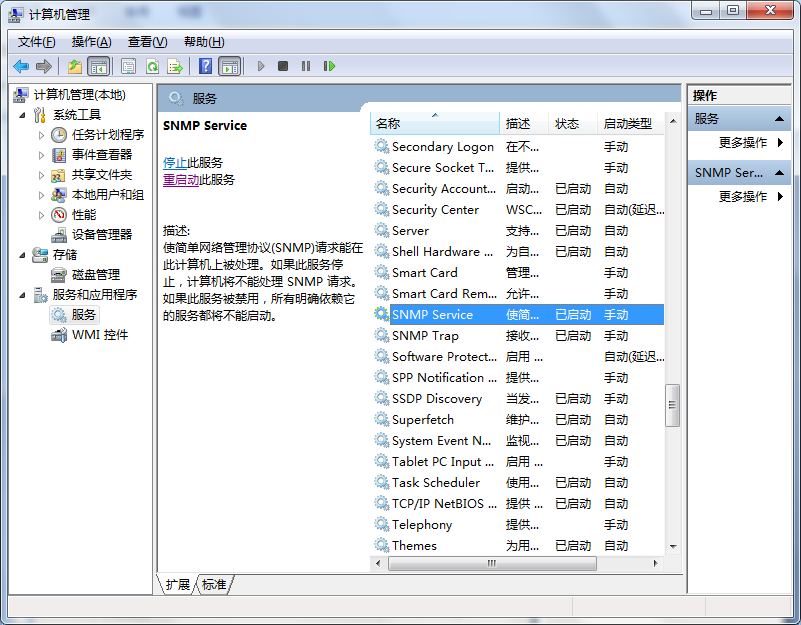


图9-16启动服务

(3)设置“代理”属性页。

选中“SNMPService”条目，单击鼠标右键，选择“属性”菜单项。在属性页集合中找到“代理”属性页，并按照下图所示设置：

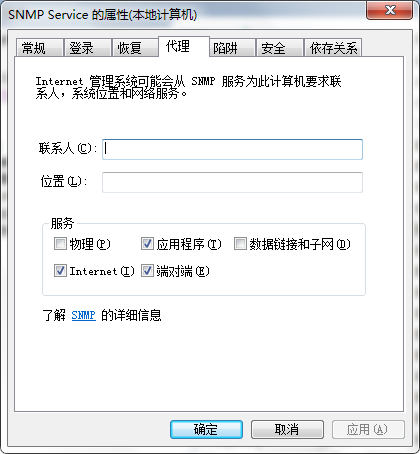


图9-17SNMPService属性

(4)设置“陷阱”属性页。

在属性页集合中找到“陷阱”属性页，并按照下图所示设置：

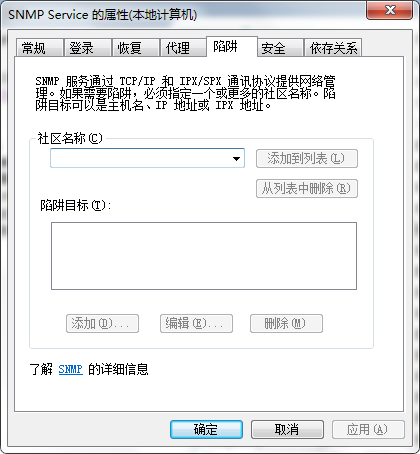


图9-18SNMPService属性

(5)设置“安全”属性页。

在属性页集合中找到“安全”属性页，并按照下图所示设置：

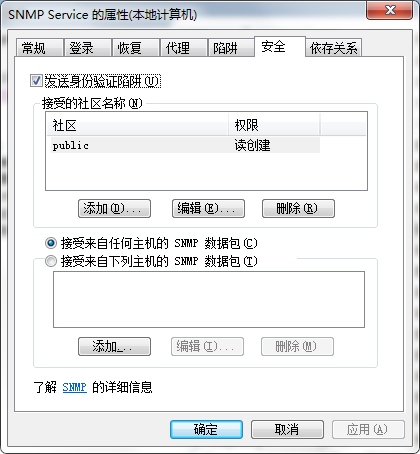


图9-19SNMPService属性

2.主机B启动协议分析器捕获数据，并设置过滤条件（提取SNMP协议）。

3.主机A扫描SNMP主机，填入主机B的IP地址。

主机A启动“实验平台工具栏中的SNMP工具”在“SNMP扫描”页签下点击[开始扫描SNMP主机]按钮，在列表中找到主机B的IP地址，双击该地址，使其添加到工具栏中的“IP”文本框中。

4.在主机A上，展开MIB树，通过双击树节点来获取代理服务器信息。

●按照返回的代理服务器信息，填写下表：

表9-3实验结果

|  |  |  |  |
| --- | --- | --- | --- |
| 代理服务器信息 | OID | 返回值类型 | 返回值 |
| 操作系统类型  (sysDescr) |  |  |  |
| 网卡数  (ifNumber) |  |  |  |
| 物理地址  (ifPhysAddress) |  |  |  |
| IP默认TTL值  (ipDefaultTTL) |  |  |  |

●通过对代理服务器信息的获取，推测该代理服务器的路由表。

5.主机B停止捕获数据。通过分析捕获到的报文，回答以下问题：

●为加深对SMI（管理信息结构）的理解，现给出某一报文中SNMP协议的数据。

302602010004067075626c6963a11902020a52020100020100300d300b06072b0601020101010500

结合SNMP报文格式和SMI定义的规则，绘制出树形的结构图（用树来表现sequence和sequenceof的关系）。

6.主机B重新启动数据捕获。

7.在主机A上，用鼠标选中

“iso.org.dod.internet.mgmt.mib-2.udp.udptable”节点，单击工具条上的[获取子树]按钮（按钮图标为说明: 1235614909328_pic）。

●通过察看代理服务器返回的结果OID列表和主机B上捕获的SNMP报文类型，简述字典式排序在SNMP查询方式中的意义。

●列出你所熟悉的代理服务器开放的UDP端口和UDP服务名。

8.关闭SNMP工具。

### 练习2设置代理服务器信息

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。

1.主机B修改SNMP服务配置，为团体“public”开放“读/写”权利。

选中“SNMPService”条目，单击鼠标右键，选择“属性”菜单项。在属性页集合中找到“安全”属性页，并按照下图所示设置：

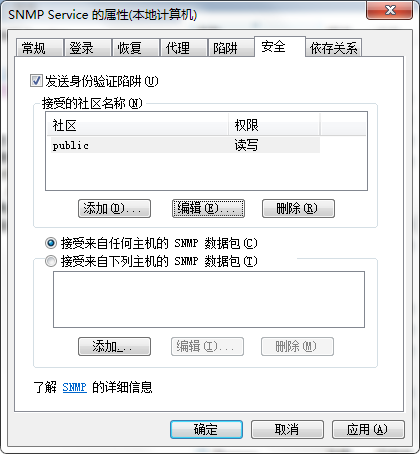


图9-20SNMPService属性

2.主机B启动协议分析器开始捕获数据并设置过滤条件（提取SNMP协议）。

3.主机A更改主机B的sysName节点值。

(1)主机A启动“实验平台工具栏中的SNMP工具”。

(2)单击“SNMP扫描”页签后点击[开始扫描SNMP主机]按钮，在列表中找到主机B的IP地址，双击该地址，使其添加到工具栏中的“IP”文本框中。

(3)单击“MIB树”页签获得树列表，在列表中找到。

“iso.org.dod.internet.mgmt.mib-2.system.sysName”节点，双击获得该值信息。

(4)在工具条上，点击[设置SNMP主机的相关信息]按钮，设置“公共体名称”下拉框为“public”；在“节点值”文本框中输入任意字符串；“变量绑定类型”单选按钮选择[OctetString]，如下图所示。点击[确定]按钮。



图9-21SNMP工具

4.主机A点击工具条上的[显示SNMP主机的相关信息]按钮，

察看“iso.org.dod.internet.mgmt.mib-2.system.sysName”节点的值，确定该值是否被更新。

5.主机B停止捕获数据，分析捕获到的数据。

6.关闭SNMP工具。

### 练习3代理服务器的事件报告

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。

1.主机B修改SNMP服务配置，为团体“public”设置“陷阱”。

(1)选中“SNMPService”条目，单击鼠标右键，选择“属性”菜单项。在属性页集合中找到“陷阱”属性页；添加“团体名称”为public；点击[添加到列表]按钮；点击[添加(D)...]按钮来设置“陷阱目标”（注意陷阱目标使用主机A的IP地址）。如下图所示设置：

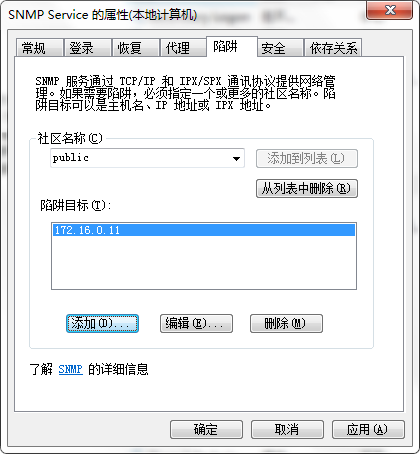


图9-22SNMPService属性

(2)在属性页集合中找到“安全”属性页；选中“接受来自这些主机的SNMP包”项；点击[添加(D)...]按钮来设置SNMP管理器的IP地址（注意此地址使用一个非组内主机的IP地址）。如下图所示设置：

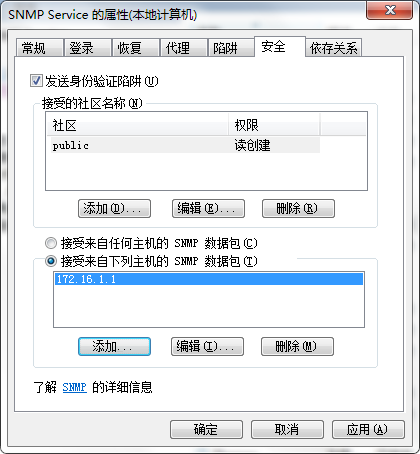


图9-23SNMPService属性

2.主机B启动协议分析器开始捕获数据并设置过滤条件（提取SNMP协议）。

3.主机A启动“实验平台工具栏中的SNMP工具”，在“IP”文本框中输入主机B的IP地址。

4.主机A通过SNMP工具尝试获取主机B（SNMP代理服务器）的信息。

主机B停止捕获数据，并分析捕获到的数据。

5.关闭SNMP工具。

思考问题：

1.SNMP使用UDP协议进行封装，分析为什么不使用TCP进行封装？

2.为什么SNMP的管理进程使用探询掌握全网状态属于正常情况，而代理进程用陷阱向管理进程报告属于较少发生的异常情况？

3.假如你是网络管理人员，你能否通过SNMP协议和以前所学的知识，实现网络拓扑发现？

## 实验10动态主机配置协议(DHCP)

【实验目的】

1.掌握DHCP的报文格式

2.掌握DHCP的工作原理

【学时分配】

2学时

【实验环境】

该实验采用网络结构一

【实验原理】

### 一.DHCP简介

DHCP（动态主机配置协议）提供了一种动态指定IP地址的机制。DHCP主要用于大型网络环境和配置IP比较困难的地方。DHCP服务器自动为客户端指定IP地址，使得网络上的计算机通信变得方便而且容易实现。DHCP使IP地址可以租用，租期从1分钟到100年不定，当租期到了的时候，服务器可以把这个IP地址分配给别的主机使用。

DHCP使用UDP协议封装，使用UDP的熟知端口67和68。

### 二.DHCP报文格式

下图为DHCP的报文格式：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 操作码（1字节） | 硬件地址类型  （1字节） | | 硬件地址长度  （1字节） | 跳数（1字节） |
| 事务ID（4字节） | | | | |
| 秒数（2字节） | | 标志（2字节） | | |
| 客户端IP地址（4字节） | | | | |
| 你（客户端）的IP地址（4字节） | | | | |
| 服务器IP地址（4字节） | | | | |
| 代理服务器IP地址（4字节） | | | | |
| 客户端硬件地址（16字节） | | | | |
| 服务器主机名称（64字节） | | | | |
| 引导文件名（128字节） | | | | |
| 选项（64字节） | | | | |

图10-1DHCP报文格式

●操作码：规定报文是请求报文还是应答报文。1代表请求报文，2代表应答报文。

●硬件地址类型：表示正在使用的硬件技术的类型。1代表以太网。

●硬件地址长度：规定硬件地址的字节长度。在以太网中，该字段的值为6。

●跳数：规定了报文所能经过的最大跳数。

●事务ID：携带一个由客户端使用的整数，来匹配应答和请求。它的值由客户端随机生成，服务器在发送应答时返回给客户端同一个值。当客户端检查请求数据包并在发送的事务ID中发现差异时，客户端的硬件地址就可以用来确保该响应是发给该客户端的。

●秒数：它表示从客户端开始引导起到目前为止，共经过的秒数。

●标志：该字段强制服务器将他的应答广播给客户端。客户端不能处理单播应答，因为它不能识别自己的IP地址。

●客户端IP地址：如果该客户已经知道自身的IP地址，它将写入“客户IP地址”字段。否则，它将该字段设置为0。

●你（客户端）的IP地址：服务器用该客户的IP地址写入“你的IP地址”字段。

●服务器IP地址：由服务器填写，该服务器的IP地址。

●代理服务器的IP地址：如果使用了某个代理服务器，则该代理服务器就填写“代理服务器的IP地址”字段。

●客户端硬件地址：指定客户端的硬件地址。

●服务器主机名称：该字段是一个空值终止串，由服务器填写。

●引导文件名：服务器在该字段填入包括用于系统引导的文件名及其所在位置的路径全名。

●选项：该字段包含额外的子字段，比如指定DHCP服务器和客户端之间接口类型的字段，以及表示租借时间长短的字段。下图所示为一种子字段的格式，该子字段规定了在客户端和服务器间交换的报文。

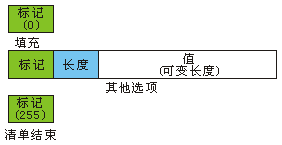


图10-2选项格式

### 三.静态地址分配与动态地址分配

#### 1.静态地址分配

静态地址分配方法将物理地址与IP地址绑定在一起，DHCP服务器将这个绑定文件存放在静态数据库中。当有主机请求DHCP服务器分配IP时，DHCP服务器首先检查静态数据库。若静态数据库存在所请求的物理地址条目，则将相应的IP地址返回给客户。

#### 2.动态地址分配

DHCP服务器还有第二个数据库，它拥有可用IP地址池。当一个DHCP客户请求临时的IP地址时，DHCP服务器就查找可用IP地址池，然后指派在可协商的期间内有效的IP地址。

从DHCP服务器获得的动态IP地址是临时地址。DHCP发出一个租用，指明了租用的时间。当租用时间到了，客户就更新租用或者停止使用这个IP地址。服务器对更新可选择同意或不同意。若服务器不同意，客户就停止使用这个地址。

### 四.状态转换

#### 1.转换状态

DHCP客户端可以从一个状态转换到另一个状态，这取决于收到的报文和发送的报文，如下图所示：

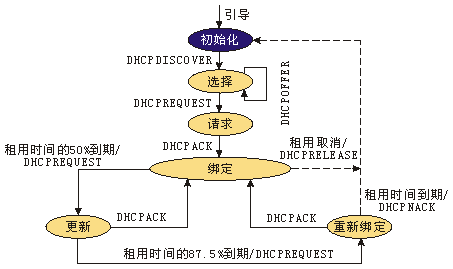


图10-3DHCP的转换图

#### 2.初始化状态

当DHCP客户端首次启动时，它处于初始化状态。客户使用UDP端口67广播DHCPDISCOVER报文（带有DHCPDISCOVER选项的请求报文）。

#### 3.选择状态

在发送DHCPDISCOVER报文后，客户就进入选择状态。接收到DHCPDISCOVER的DHCP服务器，用DHCPOFFER进行响应。在这些报文中，服务器提供了IP地址。它们还提供了租用时间，其默认值是1小时。发送DHCPOFFER报文的服务器，把提供的IP地址锁定，使这个地址不会再提供给任何其它的客户。客户选择所提供的地址中的一个，并向所选择的服务器发送DHCPREQUEST报文。然后就进入请求状态。如果客户没有收到DHCPOFFER报文，它就再尝试4次，每一次间隔2s。如果对这些DHCPDISCOVER中的任何一个都没有回答，客户就睡眠300s后再试。

#### 4.请求状态

客户继续留在请求状态，直到它从服务器收到DHCPACK报文为止，这个报文在客户的物理地址和它的IP地址之间进行了绑定。收到DHCPACK报文后，客户进入绑定状态。

#### 5.绑定状态

在这种状态下，客户在租用时间到期之前可以使用这个IP地址。当到达租用时间的50%时，客户就发送另一个DHCPREQUEST报文请求更新。客户进入更新状态。在绑定状态时，客户可以取消租用，并进入到初始化状态。

#### 6.更新状态

在更新状态下，如果客户收到更新租用协定的DHCPACK报文，客户就把计时器复位，然后回到绑定状态。如果没有收到DHCPACK报文，并且到达租用时间的87.5%时，那么客户就进入重新绑定状态。

#### 7.重新绑定状态

在重新绑定状态下，如果客户收到DHCPNACK报文或租用时间到期，则回到初始化状态，并尝试得到另一个IP地址。如果客户收到DHCPACK，它就进入到绑定状态，并把计时器复位。

### 五.DHCP运行过程

(1)发现阶段：

即DHCP客户寻找DHCP服务器的阶段。DHCP客户以广播方式（因为DHCP服务器的IP地址对于客户来说是未知的）发送DHCPDISCOVER发现报文来寻找DHCP服务器，如下图所示：

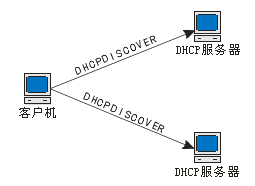


图10-4DHCP发现阶段

(2)提供阶段：

即DHCP服务器为DHCP客户端提供IP地址的阶段。在网络中接收到DHCPDISCOVER发现报文的DHCP服务器都会做出响应，它从尚未出租的IP地址中挑选一个分配给DHCP客户，向DHCP客户发送一个包含出租的IP地址和其它设置的DHCPOFFER提供报文，如下图所示：

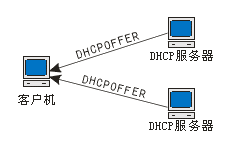


图10-5DHCP提供阶段

(3)选择阶段：

即DHCP客户端选择某台DHCP服务器提供的IP地址的阶段。如果有多台DHCP服务器向DHCP客户端发来的DHCPOFFER提供报文，则DHCP客户只接受第一个收到的DHCPOFFER提供报文，然后它就以广播方式回答一个DHCPREQUEST请求报文，该报文中包含向它所选定的DHCP服务器请求IP地址的内容。之所以要以广播方式回答，是为了通知所有的DHCP服务器，他将选择某台DHCP服务器所提供的IP地址，如下图所示：

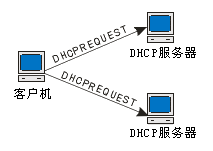


图10-6DHCP选择阶段

(4)确认阶段：

即DHCP服务器确认所提供的IP地址的阶段。当DHCP服务器收到DHCP客户回答的DHCPREQUEST请求报文之后，它便向DHCP客户发送一个包含它所提供的IP地址和其它设置的DHCPACK确认报文，告诉DHCP客户可以使用它所提供的IP地址。然后DHCP客户便将其TCP/IP协议与网卡绑定，另外，除DHCP客户选中的服务器外，其它的DHCP服务器都将收回曾提供的IP地址，如下图所示：

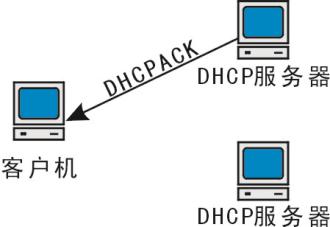


图10-7DHCP确认阶段

(5)重新登录：

以后DHCP客户端每次重新登录网络时，就不需要再发送DHCPDISCOVER发现报文了，而是直接发送含有前一次所分配的IP地址的DHCPREQUEST请求报文。当DHCP服务器收到这一报文后，它会尝试让DHCP客户继续使用原来的IP地址，并回答一个DHCPACK确认报文。如果此IP地址已无法再分配给原来的DHCP客户使用时（比如此IP地址已分配给其它DHCP客户使用），则DHCP服务器给DHCP客户回答一个DHCPNACK否认报文。当原来的DHCP客户收到此DHCPNACK否认报文后，它就必须重新发送DHCPDISCOVER发现报文来请求新的IP地址。

(6)更新租约：

DHCP服务器向DHCP客户出租的IP地址一般都有一个租借期限，期满后DHCP服务器便会收回出租的IP地址。如果DHCP客户要延长其IP租约，则必须更新其IP租约。DHCP客户启动时和IP租约期限过一半时，DHCP客户都会自动向DHCP服务器发送更新其IP租约的报文。

【实验步骤】

### 练习1使用DHCP获取IP地址

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习每台主机为一组。现仅以主机A所在组为例，其它组的操作参考主机A所在组的操作。

1.记下本机的IP地址，在命令行方式下，输入下面的命令：

“netshinterfaceipsetaddressname=“本机可用网卡的接口名”source=dhcp”。

2.启动协议分析器捕获数据，并设置过滤条件（提取DHCP协议）。

3.在命令行方式下，输入命令“ipconfig-release”。

4.在命令行方式下，输入命令“ipconfig-renew”。

5.察看DHCP会话分析，填写下表。

表10-1实验结果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 报文序号 | 操作码的值 | DHCP消息类型的值 | 租借时间  的值(若有) | 源IP地址 | 目的IP地址 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

6.等待时间超过租用时间（上表中的“租借时间”的值）的50%后，察看捕获的数据包。

●各报文中字段“操作码”、“DHCP消息类型”的值分别是多少？该请求报文的作用是什么？

### 练习2模拟重新登录

本练习一人一组，现仅以主机A为例，其它主机参考主机A的操作。

1.主机A启动协议编辑器，编辑一个DHCPRequest数据包，其中：

MAC层：

源MAC地址：本机MAC地址

目的MAC地址：服务器MAC地址

IP层：

源IP地址：本机IP地址

目的IP地址：服务器IP地址（默认为172.16.0.253）

总长度：IP层及其上层协议长度

校验和：在其它所有字段填充完毕后计算并填充

UDP层：

源端口：68

目的端口：67

有效负载长度：UDP层及其上层协议长度

计算校验和，其它字段默认

DHCP层：

操作码：1

标志：0000

客户端IP地址：主机B的IP地址（产生分配冲突）

你的IP地址：0.0.0.0

客户端硬件地址：本机的MAC地址

追加选项块：

选项代码：53

长度：1

DHCP消息类型：3

2.主机A启动协议分析器捕获数据并设置过滤条件（提取DHCP协议）。

3.发送主机A编辑好的数据包。

4.察看主机A捕获的数据。

●各报文中字段“操作码”、“DHCP消息类型”的值分别是多少？

思考问题：

1.DHCP协议适合于什么情况下使用？请举例说明。

2.DHCP协议为何使用67、68两个熟知端口进行UDP通信？

## 实验11域名服务(DNS)

【实验目的】

1.掌握DNS的报文格式

2.掌握DNS的工作原理

3.掌握DNS域名空间的分类

4.理解DNS高速缓存的作用

【学时分配】

2学时

【实验环境】

该实验采用网络结构一

【实验原理】

### 一.域名空间

在域名空间中，名字被定义在一个根在顶部的树型结构中。这个树结构最多有128层：第0层为根，如下图所示：

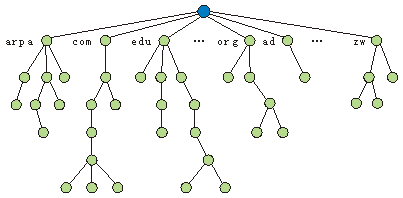


图11-1域名空间

#### 1.标号

树上的每一个节点都有一个标号，标号是一个最多有63个字符的字符串。根节点的标号是空字符串。每一个节点的子节点都具有不同的标号，这样就保证了域名是惟一的。

#### 2.域名

一个完全的域名是用点“.”分隔开的标号序列。域名总是从节点标号向上读到根节点标号的。因为最后一个标号是根节点的标号，所以一个完全的域名总是以空标号结束。因为空字符串表示什么也没有，所以域名的最后一个字符是一个点。下图给出了一个域名示例。

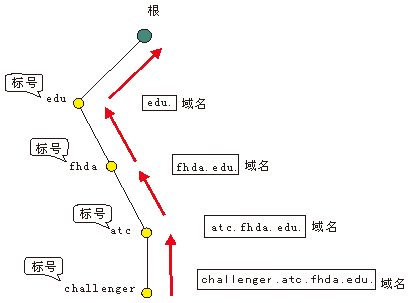


图11-2域名和标号

(1)完整域名

若域名以空字符串结束，那么这个域名就叫做完整域名（FQDN）。完整域名是包括主机全名的域名。它包括从最具体的到最一般的所有标号，并惟一地定义了该主机的名字。例如，域名：challenger.atc.fhda.edu是名为challenger的计算机的完整域名。

(2)不完整域名

若一个域名不是以空字符串结束，则它就叫做不完整域名（PQDN）。不完整域名从一个节点开始，它没有到达根节点。它适用于这样一种情况：当要被解析的域名和客户属于同一个场所时，解析程序可以自动加上缺少的部分，以便创建完整域名。例如，如果在场所atc.fhda.edu上的用户想得到计算机challenger的IP地址，用户就可以定义一个不完整域名：challenger。DNS客户在把地址传递给DNS服务器之前，会加上后缀atc.fhda.edu。

#### 3.域

域（domain）是域名空间中的子树。域的名字就是这个子树顶部节点的域名。下图给出了一些域。域本身又可划分为若干个域（有时也称它们为子域）。

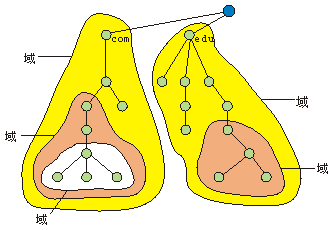


图11-3域

### 二.DNS协议简介

DNS（域名服务）是一种能够完成从域名到地址或从地址到域名的映射系统。使用DNS，计算机用户可以间接的通过域名来完成通信。Internet中的DNS被设计成为一个联机分布式数据库系统，采用客户/服务器方式工作。分布式的结构使DNS具有很强的容错性。

### 三.DNS的域名分类

在Internet中，域名空间被划分为3个部分：类属域、国家域和反向域，如下图所示：

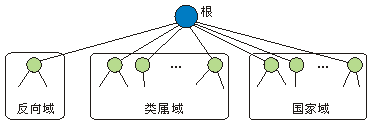


图11-4Internet中使用的DNS

#### 1.类属域

类属域按照主机的类属行为分类。树中的每一个节点定义一个域，它是到域名空间数据库的一个索引，如下图所示：

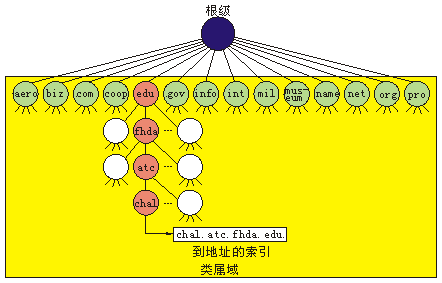


图11-5类属域

在类属域的第一级允许有14个标号。这些标号描述了不同的机构类型，如下表所示：

表11-1第一级类属域

| 标号 | 说明 | 标号 | 说明 |
| --- | --- | --- | --- |
| aero | 航空和航天公司 | int | 国际机构 |
| biz | 企业或商行（与“com”相似） | mil | 军事机构 |
| com | 商业机构 | museum | 博物馆和其它非盈利组织 |
| coop | 协作的企业组织 | name | 人名字（个人的） |
| edu | 教育机构 | net | 网络支持中心 |
| gov | 政府机构 | org | 非盈利机构 |
| info | 信息服务提供者 | pro | 专业个体组织 |

#### 2.国家域

国家域使用两字符的国家或地区的缩写（例如，用cn代表中国）。第二级标号可以是机构的标号，或者是国家指定的标号。

#### 3.反向域

反向域用来把一个地址映射为域名。例如，有时服务器会收到来自客户的请求，要完成一个任务。但是服务器不能确定这个客户是否在授权的客户列表中，因为只有客户的IP地址（从收到的IP数据包中提取出来的）被列出。要确定这个客户是否在授权列表中，服务器可以使用它的解析程序向DNS发送查询，并请求把地址映射为名字。

这种类型的查询叫做反向查询或指针（PTR）查询。要处理反向查询，在域名空间中要增加反向域，且其第一级节点叫做arpa（由于历史原因）。第二级节点叫做in-addr（表示反向地址）。域的其余部分为IP地址。

处理反向域的服务器也是分级的。这就表示地址的网络号部分要比子网号部分的等级高，而子网号部分要比主机号部分的等级高。在与类属域和国家域相比较时，反向域看起来是反过来的，如132.34.45.121的IP地址在读出时应为121.45.34.132.in-addr.arpa。下图是反向域配置的说明。

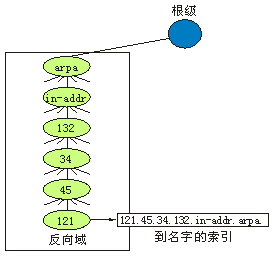


图11-6反向域

### 四.DNS报文格式

DNS报文由12字节长的首部和4个长度可变的字段组成，如下图所示：

|  |  |
| --- | --- |
| 标识（16位） | 标志（16位） |
| 问题记录数（16位） | 应答记录数（16位） |
| 授权记录数（16位） | 附加记录数（16位） |
| 查询问题 | |
| 回答（资源记录数可变） | |
| 授权（资源记录数可变） | |
| 额外信息（资源记录数可变） | |

图11-7DNS报文格式

(1)标识：该字段占两个字节，由客户程序设置并由服务器返回。客户程序通过它来确定响应与查询是否匹配。

(2)标志：16位的标志字段被划分为若干子字段，如下图所示：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| QR | Opcode | AA | TC | RD | RA | 保留 | AD | CD | rcode |
| 1位 | 4位 | 1位 | 1位 | 1位 | 1位 | 1位 | 1位 | 1位 | 4位 |

图11-8DNS标志字段

标志字段的各子字段含义如下：

●QR（查询/响应）：该位为0时是查询报文；为1时是响应报文。

●Opcode：该位为0时是标准查询；为1时是反向查询；为2时是服务器状态请求。

●AA（授权回答）：这是1位字段。当它设置为1时，表示名字服务器是权限服务器。它只用在响应报文中有效。

●TC（截断的）：该位只在响应报文中有效，它表示响应报文被切割，因为响应报文过大而无法适用于数据包的数据部分。例如，如果响应包含大量名称服务器，数据包可能会超过允许的MTU。这时，数据包将被切割，并且将TC域位设置为1。

●RD（要求递归）：如果目标名称服务器不包含所请求的信息，该域表示客户端请求递归查询。

●RA（递归可用）：该域在响应中有效，它表示响应名称服务器是否提供递归查询。

●AD（可信数据）：这位用来指定所有的数据已经被服务器认证。

●CD（验证无效）：这位指定了没有被认证的数据对于询问者来说是可以接受的。

●Rcode：该域长度为4位，用于DNS响应中，表示是否出现错误。

(3)问题记录数：该字段占两个字节，查询问题部分包含的条目数量。

(4)应答记录数：该字段占两个字节，表示回答部分包含的回答记录数。在查询报文中它的值是0。

(5)授权记录数：该字段占两个字节，包含在响应报文的授权部分的授权记录数。在查询报文中它的值是0。

(6)附加记录数：该字段占两个字节，包含在响应报文的附加部分的附加记录数。在查询报文中它的值是0。

(7)查询问题：DNS查询或响应报文中会有查询部分。查询部分中每个问题的格式如下图所示：

|  |  |
| --- | --- |
| 查询名 | |
| 查询类型 | 查询类 |

图11-9DNS查询问题字段

●查询名：表示要查找的名字，它是一个或多个标识符序列。

●查询类型：表示查询问题时的类型。最常用的查询类型是A类型，表示期望获得查询名的IP地址；而一个PTR查询则请求获得一个IP地址对应的域名。

●查询类：表示查询的类别。其值通常是1，表示Internet类型。

(8)回答（资源记录数可变）：DNS响应报文中会有回答部分。回答部分包括从服务器到客户（解析程序）的回答。其资源记录的格式如下图所示：

|  |  |
| --- | --- |
| 域名 | |
| 类型 | 类 |
| 生存时间 | |
| 资源数据长度 | 资源数据 |

图11-10DNS回答字段

●域名：表示记录中资源数据对应的名字。它的格式和查询名字字段格式相同。

●类型：表示资源记录的类型。它的值和查询类型的值是一样的。

●类：表示资源记录的类别。它的值和查询类的值是一样的。

●生存时间：表示客户程序保留该资源记录的秒数。

●资源数据长度：表示资源数据的数量。该数据的格式依赖于类型字段的值。

●资源数据：表示该资源数据的内容。

(9)授权（资源记录数可变）：DNS响应报文中会有授权部分。授权部分为该查询给出关于一个或多个授权服务器的信息（域名）。其资源记录的格式如下图所示：

|  |  |
| --- | --- |
| 域名 | |
| 类型 | 类 |
| 生存时间 | |
| 资源数据长度 | 资源数据 |

图11-11DNS授权字段

●域名：表示记录中资源数据对应的名字。它的格式和查询名字字段格式相同。

●类型：表示资源记录的类型。它的值和查询类型的值是一样的。

●类：表示资源记录的类别。它的值和查询类的值是一样的

●生存时间：表示客户程序保留该资源记录的秒数。

●资源数据长度：表示资源数据的数量。该数据的格式依赖于类型字段的值

●资源数据：表示该资源数据的内容。

(10)额外信息（资源记录数可变）：DNS响应报文中会有额外信息部分。额外信息部分提供有助于解析程序的附加信息。其资源记录格式如下图所示：

|  |  |
| --- | --- |
| 域名 | |
| 类型 | 类 |
| 生存时间 | |
| 资源数据长度 | 资源数据 |

图11-12DNS额外信息字段

●域名：表示记录中资源数据对应的名字。它的格式和查询名字段格式相同。

●类型：表示资源记录的类型。它的值和查询类型的值是一样的。

●类：表示资源记录的类别。它的值和查询类的值是一样的。

●生存时间：表示客户程序保留该资源记录的秒数。

●资源数据长度：表示资源数据的数量。该数据的格式依赖于类型字段的值。

●资源数据：表示该资源数据的内容。

### 五.正向解析与反向解析

#### 1.解析程序

DNS解析程序是客户/服务器模式的应用程序。需要把地址映射为域名或把域名映射为地址的主机要调用DNS解析程序。解析程序用映射请求找到最近的DNS服务器。若DNS服务器有这个信息，则满足解析程序的要求；否则，或者让解析程序找其它的服务器，或者再请其它服务器提供这个信息。

当解析程序收到响应后，就解释这个响应，看它是正确的解析还是错误的解析，最后把解析结果交给请求映射的进程。

#### 2.正向解析

通常，解析程序把域名交给服务器，请求服务器给出相应的地址，服务器检查类属域或国家域并查找映射。

如果需要查询的域名是类属域名或国家域名，解析程序就把这个需要查询的域名发送到本地DNS服务器进行解析。若本地服务器不能解析这个域名，它就让解析程序再找其它的DNS服务器，或者直接询问其它DNS服务器。

#### 3.反向解析

有时客户会要求将IP地址映射到相应的域名，客户把IP地址发送到DNS服务器并请求服务器映射出相应的域名，这种查询叫做反向解析，也叫做PTR查询。要回答这类查询，DNS使用反向域。在请求中，IP地址需要反过来，同时还要附上两个标号in-addr和arpa，以创建可以被反向域部分所接受的域。例如，若解析程序收到的IP地址是132.34.45.121，解析程序首先把地址反过来，并在发送前加上两个标号。发送出的域名是“121.45.34.132.in-addr.arpa”，它由DNS服务器接受和解析。

### 六.递归解析与迭代解析

#### 1.递归解析

客户（解析程序）可以向域名服务器请求递归回答。这就表示解析程序期望服务器提供最终解答。若服务器是这个域名的权限服务器，就检查它的数据库并作出响应。若服务器不是权限服务器，它就将请求发送给另一个服务器（通常是父服务器）并等待响应。若父服务器是权限服务器，则响应；否则，就将查询再发送给另一个服务器。当查询最终被解析时，响应就返回，直到最后到达发出请求的客户。下图给出了这个过程。

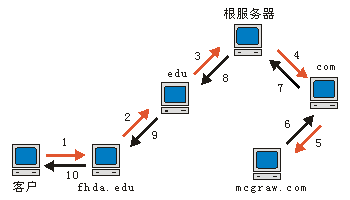


图11-13递归解析

#### 2.迭代解析

客户（解析程序）可以向域名服务器请求迭代回答。若服务器是这个域名的权限服务器，它就发送解答。若不是，就返回它认为可以解析这个查询的服务器的IP地址。客户就向第二个服务器重复查询。若新找到的服务器能够解决这个问题，就回答这个查询；否则，就向客户返回一个新的服务器的IP地址。现在客户必须向第三个服务器重复查询。这个过程称为迭代，客户向多个服务器重复发送同样的查询。在下图中，客户在从mcgraw.com服务器获得解答之前，查询了4个服务器。

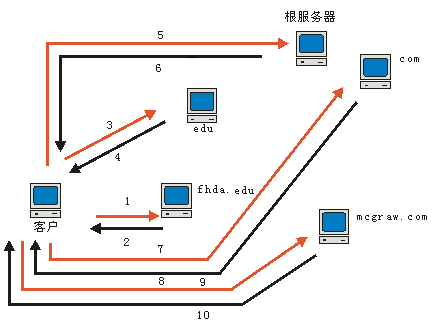


图11-14迭代解析

### 七.高速缓存

如果DNS服务器收到的需要解析的域名不在它的域中，它就会以递归的或迭代的解析方式请求父服务器。为提高效率，DNS使用高速缓存机制。当服务器向另一个服务器请求映射并收到它的响应时，在把响应发送给客户之前，DNS服务器先将这个响应存储在它的高速缓存中。若另一个客户请求同样的映射，它就检查高速缓存并解析这个域名。这时，DNS服务器把响应标记为未授权的，用来通知客户这个响应是来自高速缓存而不是来自一个授权的DNS服务器。这种高速缓存技术加速了解析过程。

地址与域名的映射关系是会发生变化的，如果DNS服务器将一个映射长期保存在高速缓存中，这个映射可能会过期，它可能把过期的映射发送给客户。目前有两种方法解决这个问题。第一种方法是权限服务器把生存时间（TTL）信息添加在映射上，生存时间定义了接收映射的服务器可以在高速缓存中保存本条映射的时间（以秒为单位）。经过这段时间后，这个映射就变成为无效的。第二种方法是DNS服务器对每一个高速缓存的映射保留一个TTL计数器。当TTL计数器到期时就清除这条映射。

### 八.压缩

当域名重复出现时，DNS就需要用偏移指针来替换。例如，在资源记录中，域名通常是问题记录的域名的重复。为了避免发生这样的重复，DNS定义了2字节的偏移指针，它指向前一次出现的该域或该域的一部分。这个指针的格式如下图所示：

|  |  |
| --- | --- |
| 11 | 开始字节的地址 |
| 2位 | 14位 |

图11-15偏移指针

例如，解析程序向本地服务器发送查询报文，要找出主机“chal.fhda.edu”的IP地址。这里将分开讨论查询报文和响应报文。

下图给出了解析程序发送出的查询报文。前两个字节是标识符（0x1333），它使响应报文与查询报文关联起来。因为解析程序可以向同一个DNS服务器发送多个查询，标识符有助于将不同的响应与查询对应起来。下一个字节是标志，其值为0x0100。查询报文如下图所示：

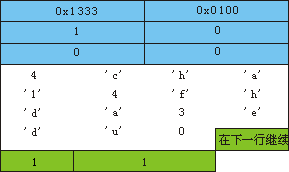


图11-16查询报文

QR位定义报文是查询报文。OpCode是0000，表示是标准查询。这个报文只包含一个问题记录。域名是4chal4fhda3edu0。下一个2字节定义查询类型是IP地址；最后两个字节定义类是Internet。

下图给出了服务器的响应报文。

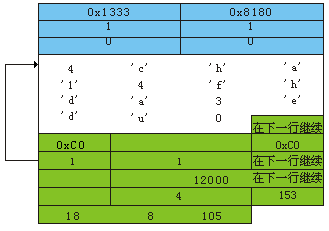


图11-17响应报文

除了标志不同的回答记录数是1，标志的值是0x8180以外，响应报文与查询报文相似。标志的值是0x8180，用二进制表示是1000000110000000，把它划分为以下的几个字段：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| QR | Opcode | AA | TC | RD | RA | 保留 | AD | CD | rcode |
| 1 | 0000 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0000 |

图11-18响应报文的标志字段

QR位定义报文是响应。OpCode是0000，它表示是标准的查询。递归可用（RA）和RD位的值设置为1。报文只包含一个问题记录和一个回答记录。问题记录是与查询报文一致的。回答记录有一个指针0xC00C（分成两行），它指向问题记录而不是重复这个域名。下一个字段定义域的类型（地址）。再下面一个字段定义类（Internet）。值为12000的字段是TTL（12000s）。再下面的字段是资源数据长度字段，资源数据是IP地址（153.18.8.105）。

### 九.DNS封装

DNS可以使用UDP封装，也可以使用TCP封装。服务器使用的是熟知端口号53。当响应报文的长度小于512字节时就使用UDP，因为大多数UDP封装受512字节的数据包长度限制。若响应报文的长度超过512字节，就要使用TCP封装。

若解析程序事先知道响应报文的长度超过512字节，就要使用TCP封装。例如，若次名字服务器（作为客户）需要主服务器传送数据，它就必须使用TCP连接，因为要传送的信息的长度通常是超过512字节的。

若解析程序不知道响应报文的长度，它可以使用UDP封装。若响应报文的长度超过512字节，服务器就截断这个报文，并把TC位置1。解析程序之后就打开TCP连接，并重复这个请求，以便从服务器得到完整的响应。

【实验步骤】

练习1Internet域名空间的分类

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习一人一组，现仅以主机A为例，其它主机的操作参考主机A。

「说明」本实验要求主机能上广域网。

1.类属域

将主机A的“首选DNS服务器”设置为公网DNS服务器，目的是能够访问Internet。

(1)主机A启动协议分析器开始捕获数据并设置过滤条件（提取DNS协议）。

(2)主机A在命令行下运行“nslookupwww.python.org”命令。

(3)主机A停止捕获数据。分析主机B捕获到的数据及主机A命令行返回的结果，回答以下问题：

●“www.python.org”对应的IP地址是什么？

●“www.python.org”域名的顶级域名的含义是什么？

2.国家域

(1)主机A启动协议分析器开始捕获数据并设置过滤条件（提取DNS协议）。

(2)主机A在命令行下运行“nslookupwww.jl.gov.cn”命令。

(3)主机A停止捕获数据。分析主机B捕获到的数据及主机A命令行返回的结果，回答以下问题：

●“www.jl.gov.cn”对应的IP地址是什么？

●“www.jl.gov.cn”域名的顶级、二级、三级域名的含义是什么？

3.反向域

(1)将主机A的“首选DNS服务器”设置为服务器的IP地址（默认为172.16.0.253）。

(2)主机A启动协议分析器开始捕获数据并设置过滤条件（提取DNS协议）。

(3)主机A在命令行下运行“nslookup172.16.0.253”命令。

(4)主机A停止捕获数据。分析主机A捕获到的数据及主机A命令行返回的结果，回答以下问题：

●172.16.0.253对应的域名是什么？

●反向域的顶级、二级域名分别是什么？

思考问题：

1.Internet的域名结构是怎样的？它与目前的电话网的号码结构有何异同之处？

练习2DNS正向查询

本练习一人一组，现仅以主机A为例，其它主机的操作参考主机A。

「说明」

本练习中要求每台主机配置DNS服务器，（DNS服务器的IP地址即服务器的IP地址）其IP地址以172.16.0.253为例。

各组主机IP地址配置如下：

第一组六台主机IP地址依次为172.16.0.11，172.16.0.12……172.16.0.16；

第二组六台主机IP地址依次为172.16.0.21，172.16.0.22……172.16.0.26；

其它各组以此类推。

1.在主机A执行命令“nslookup本机的IP地址”获取本机的域名，记录下来。

2.主机A启动协议编辑器，编写一个DNS正向查询报文。其中：

MAC层：

源MAC地址：本机MAC地址

目的MAC地址：服务器的MAC地址

IP层：

源IP地址：本机IP地址

目的IP地址：服务器的IP地址（默认为172.16.0.253）

总长度：IP层及其上层协议总长度

校验和：IP层字段全部编辑完成后，计算IP层校验和

UDP层：

源端口：大于1024的端口

目的端口：53

有效负载长度：UDP层及其上层协议总长度

校验和：所有字段编辑完成后，计算校验和

DNS层：

标志：0100

问题记录数：1

域名循环体：选中第一个“域名循环体”项，点击右边按钮[B]来追加域名块。按格式要求填写步骤1获取的主机的域名。

例如：设步骤1中获取的域名为host11.Netlab，则追加2块。选中“域名循环体”下的“长度”项，修改长度值；出现“域标记”项，选中“域标记”项，点击右边按钮[E]输入相应的值。最后一块“长度”字段为0。

查询类型：1

查询类别：1

设置如图所示：

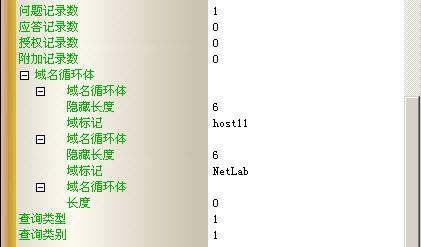


图11-19DNS帧的编辑

3.主机A启动协议分析器开始捕获数据，并设置过滤条件（提取DNS协议）。

4.主机A发送已编辑好的报文。

5.主机A停止捕获数据。在捕获到的数据中查找DNS响应报文。

● 在响应报文中提取对方主机的IP地址。

思考问题：

1.域名的IP地址是否只有一个？

2.域名服务协议的主要功能是什么？域名服务协议中的根服务器和授权服务器有何区别？授权服务器与管辖区有何关系？

练习3DNS反向查询

本练习一人一组，现仅以主机A为例，其它主机的操作参考主机A。

1.该练习中，DNS服务器及各主机IP地址配置同练习二。

2.主机A启动协议编辑器，编写一个DNS反向查询报文。其中：

MAC层：

源MAC地址：本机MAC地址

目的MAC地址：服务器的MAC地址

IP层：

源IP地址：本机IP地址

目的IP地址：服务器的IP地址（默认为172.16.0.253）

总长度：IP层及其上层协议总长度

校验和：IP层字段全部编辑完成后，计算IP层校验和

UDP层：

源端口：大于1024的合法端口号

目的端口：53

有效负载长度：UDP层及其上层协议总长度

校验和：所有字段编辑完成后，计算校验和

DNS层：

标志：0100

问题记录数：1

域名循环体：选中“域名循环体”项，点击右边按钮[B]来追加域名块。按格式要求填写主机反向域域名（反转IP+.in-addr.arpa）。例如：设主机A的IP地址为172.16.0.11，则它的反向域为11.0.16.172.in-addr.arpa，这需要追加6个块，其中最后一个块“长度”字段为0，如图所示：

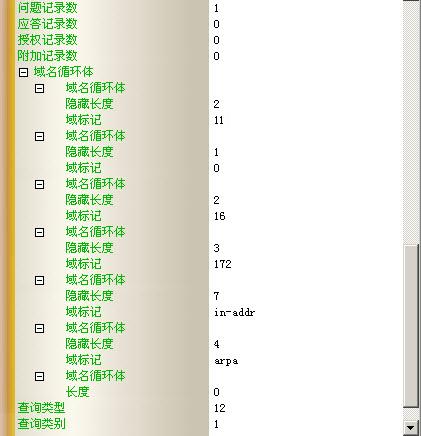


图11-20DNS帧的编辑

查询类型：12

查询类别：1

3.主机A启动协议分析器开始数据捕获，设置过滤条件（提取DNS协议）。

4.主机A发送已编辑好的报文。

5.主机A停止捕获数据。在捕获到的数据中查找DNS响应报文。

●在响应报文中提取主机A的域名地址。

练习4DNS的应用及高速缓存

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。

1.该练习中，DNS服务器及各主机IP地址配置同练习二。

2.主机A、B分别在命令行下执行“ipconfig/flushdns”命令来清空DNS高速缓存。

3.主机A、B分别启动协议分析器开始捕获数据并设置过滤条件（提取DNS协议和ICMP协议）。

4.主机A、B分别在命令行下执行“ping对方的域名”命令，然后执行“ipconfig/displaydns”命令来显示DNS高速缓存。在缓存中找到对方的域名所对应的记录。

5.主机A、B在命令行下再次执行“ping对方主机的域名”命令。

6.主机A、B停止捕获，分析其捕获的数据及对方的DNS高速缓存中的内容，回答问题：

●简述在使用域名完成的通信中，DNS协议所起到的作用。

●简述DNS高速缓存的作用。

●参考“会话分析”视图的显示结果，绘制此次访问过程的报文交互图（包括ICMP协议）。

7.恢复网络环境，将“首选DNS服务器”清空。

## 实验12网络地址转换(NAT)

【实验目的】

1.理解NAT的转换机制

2.理解NAT转换表的作用

3.了解静态地址转换的原理及作用

4.了解动态地址转换的原理及作用

【学时分配】

2学时

【实验环境】

该实验采用网络结构二

【实验原理】

### 一.NAT简介

NAT（网络地址转换）是将专用网络地址（如企业内部网intranet）转换为公用地址（如互联网Internet）的一种技术，它对外界隐藏了内部网络的IP地址。通过在内部使用专用IP地址，并将它们转换为一小部分外部网络地址，从而减少了IP地址注册的费用以及节省了目前越来越缺乏的IP地址。同时这也隐藏了内部拓扑结构，从而降低了内部网络受到攻击的风险。

NAT功能通常被集成到路由器、防火墙、单独的NAT设备中，NAT设备（或软件）维护一个状态表，用来把内部网络的专用IP地址映射到外部网络地址上去。每个数据包在NAT设备（或软件）中都被转换成正确的IP地址发往下一级。与普通路由器不同的是，NAT设备是对报头进行修改，将内部网络的源地址变为NAT设备自己的外部网络地址；而普通路由器仅在将数据包转发到目的地之前读取源地址和目的地址。

### 二.NAT转换表

NAT使用一个转换表来记录内部专用地址和外部地址之间的映射。

#### 1.使用一个IP地址

最简单的形式是一个转换表只有两列；专用地址和外部地址。当这个路由器对出数据包的源地址进行转换时，它也记下来目的地址。当响应从这个目的地址返回时，路由器就使用这个数据包的源地址（作为外部地址）来找出这个数据包的专用地址。这里描述的NAT机制要求通信是由专用网发起的。下图给出了使用一个IP地址的情况。



图12-1使用一个IP地址的NAT转换表

#### 2.使用IP地址池

使用一个IP地址的NAT，只允许一个专用网主机连接一个外部主机。要去掉这个限制，NAT路由器可以使用全球地址池。例如，不是仅使用一个全球地址（200.24.5.8），NAT路由器可以使用4个地址（100.24.5.8、100.24.5.9、100.24.5.10、100.24.5.11）。在这种情况下，4个专用网主机可同时接入到同一个外部主机，因为每一对地址定义一条连接。但是，这种方法仍然有一些缺点。到同一个目的地址的连接不能超过4条。没有一个专用网主机可以同时访问两个外部服务器程序。同样地，两个专用网主机也不能同时访问同一个外部服务器程序。

#### 3.同时使用IP地址和端口号

要允许在专用网主机和外部服务器程序之间有多对多的关系，就需要在转换表中有更多的信息。例如，假定在专用网内有两个主机，地址为172.18.3.1和172.18.3.2，它们需要访问主机25.8.3.2上的HTTP服务器。如果转换表有5列而不是两列，即包括源端口号和目的端口号，以及传输层协议，那么这种二义性就消失了。下表给出了转换表的例子。

表12-1同时使用IP地址和端口号

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 专用地址 | 专用端口 | 外部地址 | 外部端口 | 传输层协议 |
| 172.18.3.1 | 1400 | 25.8.3.2 | 80 | TCP |
| 172.18.3.2 | 1401 | 25.8.3.2 | 80 | TCP |
| …… | …… | …… | …… | …… |

当从HTTP返回响应时，源地址（25.8.3.2）和目的端口号（1400）的组合定义了这个响应应当指向专用网主机。要使这样的转换能够正常的工作，临时端口号必须是惟一的。

### 三.NAT的地址转换方式

#### 1.静态地址转换

所谓静态地址转换是指专用地址和外部网络地址之间是静态一一映射的。这种转换通常用在内部网上的主机需要对外提供服务（如Web、E-mail服务等）的情况下。

#### 2.动态地址转换

在动态地址转换的方式下，一组专用地址与一个外部网络地址池之间建立起一种动态的一一映射关系。这种地址转换形式下，内部主机可以访问外部网络，外部主机也能对内部网络进行访问，但必须是在内网专用IP地址与外部网络地址之间存在映射关系时才能成功，并且这种映射关系是动态的。

【实验步骤】

练习1静态地址转换

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A、B、C、D、E、F作为一组进行实验。

本练习中主机B作为NAT服务器（主机B的b1接口连接到Internet，b2接口连接到内部局域网），主机A、C、D作为Internet上的主机，主机E、F作为内部服务器。  
1. 主机F启动实验平台工具栏中的“UDP工具”，作为UDP服务器端来监听2929端口。  
2.为主机B启动静态NAT服务，配置方法如下：

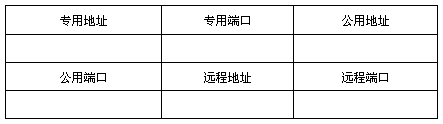
（1）首先点击工具栏中的安装VxGate按钮，安装VxGate驱动，安装时选型“2”，然后按任意键完成安装。

（2）在主机B上启动VxGate，并进行配置；

（3）主机B点击VxGate配置按钮，点击WAN 01选项，选择“启用该网络接口”，网络类型选择“外网直连”，网络设置中选择 "b1" 网卡，在外网直连设置中选择IP地址为172.16.1.1.，网关填写为172.16.1.2。

（4）主机B点击VxGate配置按钮，点击LAN 01选项，选择“启动该网络接口”，网络设置中选择"b2"，IP地址选择172.16.0.1，子网掩码255.255.255.0。服务范围填写172.16.0.1-172.16.0.255，高级设置中，虚拟本地网关填写172.16.0.1。点击确定完成配置。

（5）点击视图，选择“端口映射表”，在表上点击右键，选择添加，在WAN接口中选择“b1”网卡，在LAN接口中选择“b2”接口，客户机IP填写为172.16.0.3，点击MAC地址后会自动生成客户机的MAC地址。客户机端口填写2929映射到6001，协议选择UDP。配置完成后点击启动按钮启动VxGate。  
3.主机A、B、C、F启动协议分析器开始捕获数据并设置过滤条件（提取UDP协议）。  
4.主机A启动实验平台工具栏中的“UDP工具”并向主机B（172.16.1.1）的6001端口发送一条数据。  
5.主机A、B、C、F停止捕获数据，分析捕获到的数据。  
  ● 分析主机B捕获到的数据，结合静态NAT的原理，试填写会话映射表：

表12-2  实验结果  


● 结合本练习的结果，绘制第4步发送的UDP数据包在网络中的传输路径图。

6.主机B关闭VxGate，并点击工具栏中的卸载驱动程序，还原实验环境。

练习2动态地址转换

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。本练习将主机A、B、C、D、E、F作为一组进行实验。

本练习中主机B作为NAT服务器（主机B的b1接口连接到Internet，b2接口连接到内部局域网），主机A、C、D作为Internet上的主机，主机E、F作为内部服务器。  
1.主机A启动实验平台工具栏中的“UDP工具”，作为UDP服务器端来监听2828端口。  
2.为主机B启动静态NAT服务，配置方法如下：

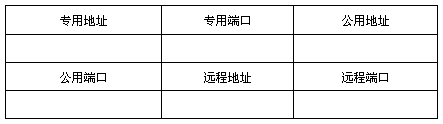
（1）首先点击工具栏中的安装VxGate按钮，安装VxGate驱动，安装时选型“2”，然后按任意键完成安装。

（2）在主机B上启动VxGate，并进行配置；

（3）主机B点击VxGate配置按钮，点击WAN 01选项，选择“启用该网络接口”，网络类型选择“外网直连”，网络设置中选择 "b1" 网卡，在外网直连设置中选择IP地址为172.16.1.1.，网关填写为172.16.1.2。

（4）主机B点击VxGate配置按钮，点击LAN 01选项，选择“启动该网络接口”，网络设置中选择"b2"，IP地址选择172.16.0.1，子网掩码255.255.255.0。服务范围填写172.16.0.1-172.16.0.255，高级设置中，虚拟本地网关填写172.16.0.1。点击确定完成配置。配置完成后点击启动按钮，启动VxGate。  
3.主机A、B、C、F启动协议分析器开始捕获数据并设置过滤条件（提取UDP协议）。  
4.主机F启动实验平台工具栏中的“UDP工具”并向主机A（172.16.1.2）的2828端口发送一条数据。  
5.主机A、B、C、F停止捕获数据，分析捕获到的数据。

●分析主机B捕获到的数据，结合静态NAT的原理，试填写会话映射表：

表12-2  实验结果  


● 结合本练习的结果，绘制第4步发送的UDP数据包在网络中的传输路径图。  
6.主机B关闭VxGate，并点击工具栏中的卸载驱动程序，还原实验环境。

## 实验13超文本传输协议(HTTP)

【实验目的】

1.掌握HTTP的报文格式

2.掌握HTTP的工作原理

3.掌握HTTP常用方法

【学时分配】

4学时

【实验环境】

该实验采用网络结构一

【实验原理】

### 一.体系结构

万维网（WWW）服务是分布式的客户/服务器服务，客户使用浏览器能够得到服务器提供的服务，这种服务的提供是分布在许多网站中的，如下图所示：

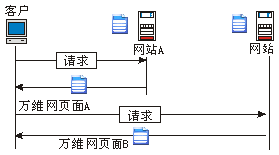


图13-1分布式的服务

每一个网站保存有一个或多个文档，叫做万维网页面。在图13-1所示的例子中，客户需要查看网站A的某些信息。浏览器用来读取万维网上的文档。它向网站A发送一个请求，这个请求包含了网站A和网站中万维网页面的地址，这个地址叫做统一资源定位符（URL）。网站A收到请求后，将指定的文档发送给这个客户。它用同样的方法与网站B通信。

#### 1.客户（浏览器）

许多厂商提供商用浏览器，可以解释和显示万维网页面。每一个浏览器通常由3个部分组成：控制程序、客户程序及解释程序。控制程序从键盘或鼠标接收输入，使用客户程序访问要浏览的文档。在文档找到后，控制程序就使用解释程序（可以是HTML、Java或JavaScript等）中的一个，把文档显示在屏幕上。

#### 2.服务器

万维网页面存储在服务器上。每当有客户请求到达时，对应的文档就发送给客户。为了提高效率，服务器通常在其高速缓存中存储被请求过的文档；对高速缓存进行访问要比磁盘快得多。通过多线程或多进程可使服务器的效率更加提高，服务器在同一时间可回答多个请求。

#### 3.统一资源定位符（URL）

客户要访问万维网页面就需要这个页面地址。为了方便地访问文档，HTTP协议使用统一资源定位符（URL）。URL是Internet上指定信息的标准。URL由4部分组成：协议、主机、端口和路径，如下图所示：

说明: 1316245444395_pic

图13-2URL

●协议：协议指定了用这个URL的客户/服务器程序。例如，HTTP协议、FTP协议和TELNET协议等。

●主机：主机指明了信息所存放的地址，可以是逻辑地址也可以是相应的域名。存放万维网页面的计算机通常使用以字符“WWW”开始的域名，但这不是强制性的。

●端口：端口指定了使用主机的某个端口，端口是可选的。如果包含了端口，那么端口就插入在主机和路径之间，和主机用冒号分隔开。

●路径：路径指定了文件存放的位置。路径本身可以包含斜线，用于将目录与子目录和文件分隔开。

### 二.万维网文档

万维网文档可分为3类：静态文档、动态文档和活动文档。这种分类是基于文档内容被确定的时间。

#### 1.静态文档

静态文档是固定内容的文档，它由服务器创建，并存储在服务器中。客户只能得到文档的一个副本。换言之，文件的内容是在文件被创建时就确定的，而不是在它被使用时。当然，在服务器中的内容是可以改变的，但用户不能改变它。当客户访问文档时，文档的一个副本就被发送出去。用户可以使用浏览程序显示这个文档，如下图所示：

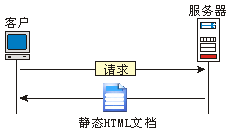


图13-3静态文档

#### 2.动态文档

动态文档是在浏览器请求该文档时才由万维网服务器创建出来。当请求到达时，万维网服务器就运行创建动态文档的应用程序。服务器返回这个程序或脚本的输出，把它作为对请求该文档的浏览器的响应。因为对每一个请求都创建出新的文档，因此每一个请求得到的动态文档的内容就会不同。

#### 3.活动文档

对于许多应用，需要程序能够在客户端运行，这就叫做活动文档。当浏览器请求活动文档时，服务器就发送这个文档的一个副本或脚本。然后这个文档就在客户（浏览器）端运行。

### 三.HTTP协议简介

HTTP（超文本传输协议）主要用于访问万维网上的数据。协议以普通文本、超文本、音频、视频等格式传输数据。之所以称为超文本协议，原因是在应用环境中，它可以快速的在文档之间跳转。HTTP在熟知端口80上使用TCP服务。

### 四.HTTP报文格式

HTTP报文有两种一般的类型：请求和响应。这两种类型的报文格式几乎是相同的。报文格式如下图所示：

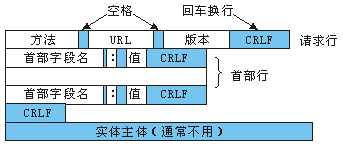


图13-4HTTP报文格式

### 五.HTTP方法

HTTP报文中的方法是客户端向服务器端发出的实际命令和请求。常用HTTP方法如下表所示：

表13-1HTTP方法

|  |  |
| --- | --- |
| 方法 | 动作 |
| GET | 从服务器请求一个文档 |
| HEAD | 请求关于一个文档的信息，但不是这个文档本身 |
| POST | 从客户向服务器发送一些信息 |
| PUT | 从服务器向客户发送一些信息 |
| TRACE | 把到达的请求回送 |
| CONNECT | 用于使用代理连接 |
| OPTION | 询问关于可用的选项 |

### 六.HTTP状态码

在响应报文中，请求行被替换为状态行，由3位数字组成，表示请求是否被理解或被满足。HTTP协议的状态码如下表所示：

表13-2HTTP状态码

| 代码 | 短语 | 说明 |
| --- | --- | --- |
| 提供信息 | | |
| 100 | Continue | 请求的开始部分已经收到，客户可以继续它的请求 |
| 101 | Switching | 服务器同意客户请求，切换到在更新首部中定义的协议 |
| 成功 | | |
| 200 | OK | 请求成功 |
| 201 | Created | 新的URL被创建 |
| 202 | Accepted | 请求被接受，但还没有马上起作用 |
| 204 | Nocontent | 主体中没有内容 |
| 重新定向 | | |
| 301 | Movedpermanently | 服务器已经不再使用所请求的URL |
| 302 | Movedtemporarily | 所请求的URL已暂时地删除了 |
| 304 | Notmodified | 文档还没有被修改 |
| 客户差错 | | |
| 400 | Badrequest | 在请求中有语法差错 |
| 401 | Unauthorized | 请求缺少适当地授权 |
| 403 | Forbidden | 服务被拒绝 |
| 404 | Notfound | 文档未找到 |
| 405 | Methodnotallowed | 这个方法不被这个URL所支持 |
| 406 | Notacceptable | 所请求的格式不可接受 |
| 服务器差错 | | |
| 500 | Internalservererror | 在服务器端有差错，如崩溃 |
| 501 | Notimplemented | 所请求的动作不能完成 |
| 503 | Serviceunavailable | 服务暂时不可用，但可能在以后能够被请求 |

### 七.持久与非持久连接

在版本1.1以前的HTTP协议默认是非持久连接的，而在版本1.1中，持久连接是默认的连接。

#### 1.非持久连接

在非持久连接中，对每一个请求/响应都要建立一次TCP连接。非持久连接的步骤如下：

●客户打开TCP连接，并发送请求。

●服务器发送响应，并关闭连接。

●客户读取数据，直到它遇到文件结束标记；然后它关闭连接。

使用非持久连接时，对于在不同文件中的N个不同图片的请求，连接必须打开和关闭N次。非持久连接策略给服务器造成了很大的开销，因为服务器需要N个不同的缓存，而每次打开连接时都要使用慢开始过程。

#### 2.持久连接

HTTP版本1.1指明持久连接是默认的策略。在使用持久连接时，服务器在发送响应后，让连接继续为一些请求打开。服务器可以在客户发送关闭请求时等待或关闭这个连接。发送端通常在每一个响应中都发送数据长度。但是，有时发送端并不知道数据的长度（例如动态文档或活动文档），这时服务器就把长度不知道这件事通知客户，并在发送数据后关闭这个连接，这样客户就知道数据结束的地方已经到了。

### 八.HTTP代理服务器

HTTP支持代理服务器（proxyserver）。代理服务器保留对最近请求的响应的副本。在有代理服务器的情况下，HTTP客户把请求发送给代理服务器。代理服务器检查它的高速缓存。如果在高速缓存中有这个响应，代理服务器就直接应答客户的请求；反之，如果在高速缓存中没有这个响应，代理服务器就把请求发送给相应的HTTP服务器，当HTTP服务器的响应到达代理服务器时，代理服务器将该响应转发给客户，同时将该响应存储到高速缓存中，以便以后为其它客户的请求使用。

代理服务器减少了原始服务器的负荷，减小了通信量，也减小了延时。但是，由于使用了代理服务器，客户必须配置成接入到代理服务器而不是那个目标服务器。

【实验步骤】

练习1页面访问

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习一人一组，现仅以主机A为例，其它主机参考主机A的操作。

1.主机A清空IE缓存。

2.主机A启动协议分析器开始捕获数据，并设置过滤条件（提取HTTP协议）。

3.主机A启动IE浏览器，在“地址”框中输入http://*服务器的ip*/experiment，并连接，服务器的ip默认为172.16.0.253。

4.主机A停止捕获数据，分析捕获到的数据，并回答以下问题：

●本练习使用HTTP协议的哪种方法？简述这种方法的作用。

●根据本练习的报文内容，填写下表。

表13-3实验结果

|  |  |
| --- | --- |
| 主机名 |  |
| URL |  |
| 服务器类型 |  |
| 传输文本类型 |  |
| 访问时间 |  |

●参考“会话分析”视图显示结果，绘制此次访问过程的报文交互图（包括TCP协议）。

●简述TCP协议和HTTP协议之间的关系。

思考问题：

1.一个主页是否只有一个连接？

练习2页面提交

本练习一人一组，现仅以主机A为例，其它主机参考主机A的操作。

1.主机B启动协议分析器开始捕获数据，并设置过滤条件（提取HTTP协议）。

2.主机A启动IE浏览器，在“地址”框中输入“http://*服务器的ip*/experiment/post.html”，并连接，服务器的ip默认为172.16.0.253。在返回页面中，填写“用户名”和“密码”，点击[确定]按钮。

3.主机A停止捕获数据，分析捕获到的数据，并回答以下问题：

●本练习的提交过程使用HTTP协议的哪种方法？简述这种方法的作用。

●此次通信分几个阶段？每个阶段完成什么工作？

●参考“会话分析”视图显示结果，绘制此次提交过程的报文交互图（包括TCP协议）。

练习3获取页面信息

本练习一人一组，现仅以主机A为例，其它主机参考主机A的操作。

1.主机A启动实验平台工具栏中的“TCP工具”。

2.主机A启动协议分析器开始捕获数据，并设置过滤条件（提取HTTP协议）。

3.主机A在“TCP工具”上，选中“客户端”单选框，设置“IP地址”为服务器IP（默认为172.16.0.253）；设置“端口”为80；单击[连接]按钮来和服务器建立连接。

4.主机A在“TCP工具”上，设置“发送数据（文本）”为以下内容：

HEAD/experiment/HTTP/1.1<CRLF>

Host:172.16.0.253<CRLF>

<CRLF>

点击[发送]按钮。（注：<CRLF>是回车换行）

点击[断开]按钮，断开TCP连接（由于不同http版本所遵循的规范不同，有些HTTP服务器不需要断开操作）。

5.主机A在“TCP工具”上的“显示数据（文本）”中察看服务器返回信息。

6.主机A停止捕获数据，分析捕获到的数据。

思考问题：

1.同时打开多个浏览器窗口并访问一个WEB站点的不同页面时，系统是根据什么把返回的页面正确地显示到相应窗口的？

练习4较复杂的页面访问

本练习一人一组，现仅以主机A为例，其它主机参考主机A的操作。

1.本练习中要求各主机设置DNS服务器地址，（DNS服务器的IP地址即主控中心平台的IP地址）其IP地址以172.16.0.253为例。

2.主机A使用“ipconfig/flushdns”命令清空DNS高速缓存。

3.主机A启动协议分析器开始捕获数据并设置过滤条件（提取DNS、HTTP协议）。

4.主机A启动IE浏览器，在地址框中输入http://JServer.NetLab/complexpage.htm。

5.主机A停止捕获数据，察看相关会话，分析捕获到的数据，并回答以下问题：

●结合本次实验结果，简述浏览器是如何处理一个访问请求的。

6.恢复网络环境，将“首选DNS服务器”清空。

思考问题：

1.为什么HTTP不保持与客户端的TCP连接？

## 实验14远程登录与文件传送协议(TELNET与FTP)

【实验目的】

1.掌握TELNET的工作过程

2.理解TELNET选项协商

3.掌握FTP的工作原理

4.掌握FTP一些常用命令的使用方法及用途

【学时分配】

4学时

【实验环境】

该实验采用网络结构一

【实验原理】

### 一.分时系统

在分时系统中，计算机支持多个用户。用户使用由键盘、鼠标和监视器组成的终端与分时系统进行交互。中央计算机进行所有的计算。当用户在键盘上键入字符时，这个字符发送到计算机，同时回送到监视器。分时系统使用户有专用计算机的感觉。用户可以运行程序，使用系统资源，从一个程序切换到另一个程序，等等。

### 二.本地登录与远程登录

在分时系统中，用户是系统的一部分，具有使用资源的权利。每一个授权用户都有一个标识，也可能还有一个口令。用户标识定义用户是系统的一部分。要接入到系统，用户要使用用户标识或登录名字登录到系统。系统对用户口令进行检查以防止非授权用户使用资源。

#### 1.本地登录

用户登录到本地的分时系统叫做本地登录。用户在终端上进行操作，该操作被终端驱动程序接受。接着操作系统就解释字符的组合，并调用所需的应用程序，如下图所示：

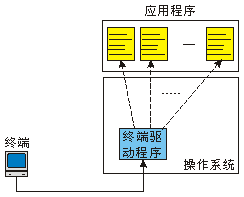


图14-1本地登录

#### 2.远程登录

用户登录到远程主机并使用远程主机的应用程序，这叫做远程登录。远程登录一般要使用TELNET客户程序和服务器程序。用户把自己的击键发送给终端驱动程序，本地操作系统接受这些字符，但并不解释它们。这些字符被送到TELNET客户。然后，TELNET客户把这些字符转换成叫做网络虚拟终端字符的通用字符集，再把它放入本地TCP/IP栈，如下图所示：

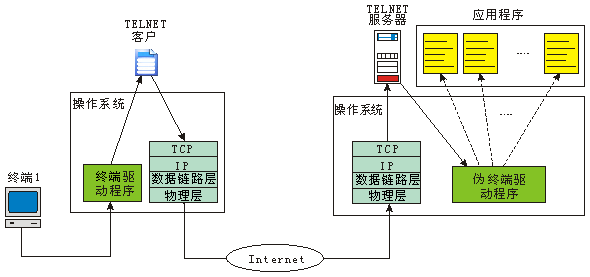


图14-2远程登录

网络虚拟终端形式的命令或文字，通过Internet到达远程主机的TCP/IP栈。字符被交付给操作系统，然后递交给TELNET服务器，TELNET服务器再把这些字符转换为远程计算机可理解的相应字符。这些字符不能直接交给操作系统，远程的操作系统不能接收来自TELNET服务器的字符（它只能接收来自终端驱动程序的字符）。TELNET服务器使用伪终端驱动程序将这些字符伪装成好像是从一个终端发来的。然后操作系统将这些字符传递给适当的应用程序。

### 三.网络虚拟终端

接入到远程计算机的过程是很复杂的，这是因为不同的操作系统接受特殊的字符组合是不同的。例如，在Windows操作系统的计算机中，文件结束标记是Ctrl+z，而在UNIX操作系统中则是Ctrl+d。

TELNET使用网络虚拟终端（NVT）字符集来处理异构系统的远程登录问题。网络虚拟终端（NVT）字符集是一个通用接口，通过这个接口，客户TELNET把来自本地终端的字符（数据或命令）转换成NVT形式，然后交付给网络。而服务TELNET把来自NVT形式的字符（数据或命令）转换成计算机可接受的形式。下图给出了NVT字符集的概念。

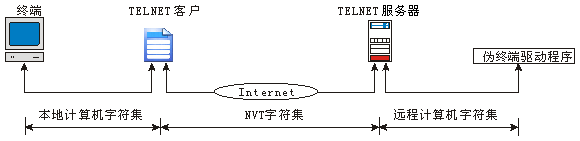


图14-3NVT的概念

### 四.TELNET简介

TELNET是一种终端仿真技术，它提供了一种通过网络操作远程主机方法。TELNET使用面向字节的双向通信，服务器通常使用TCP的23端口，客户端使用动态端口。TELNET协议可以工作在任何主机和任何终端之间，它使用TCP/IP在远程计算机上登录并执行命令。

### 五.NVT字符集

NVT使用两个字符集，一个为数据字符集，另一个为远程控制字符集。

#### 1.数据字符

对于数据，NVT通常使用NVTASCII。这是8位字符集，其中低7位和USASCII是一样的，但是最高位是0。虽然这也可以使用8位的ASCII（最高位可以是0或1），但这必须在客户和服务器之间使用选项协商取得一致，如下图所示：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 |  |  |  |  |  |  |  |

图14-4数据字符的格式

#### 2.远程控制字符

远程控制字符用于在客户与服务器之间发送控制字符，NVT使用8位字符集，其最高位置为1。如下图所示：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  |  |  |  |  |

图14-5控制字符的格式

下表列举了一些远程控制字符及其意义。

表14-1某些NVT远程控制字符

| 字符 | 十进制 | 十六进制 | 意义 |
| --- | --- | --- | --- |
| Echo | 1 | 0x01 | 回显 |
| Suppressgo\_ahead | 3 | 0x03 | 向前抑制 |
| Status | 5 | 0x05 | 状态 |
| Terminal\_type | 24 | 0x18 | 终端类型 |
| Negotiateaboutwindowsize | 31 | 0x1f | 窗口尺寸协商 |
| Terminalspeed | 32 | 0x20 | 终端速度 |
| Remoteflowcontrol | 33 | 0x21 | 远程溢出控制 |
| Xdisplaylocation | 35 | 0x23 | X显示定位 |
| authentication | 37 | 0x25 | 认证 |
| encrypt | 38 | 0x26 | 数据加密 |
| Newenvironment | 39 | 0x27 | 新环境 |
| EOF | 236 | 0xEC | 文件结束 |
| EOR | 239 | 0xEF | 记录结束 |
| SE | 240 | 0xF0 | 子选项结束 |
| NOP | 241 | 0xF1 | 无操作 |
| DM | 242 | 0xF2 | 数据标记 |
| BRK | 243 | 0xF3 | 断开 |
| IP | 244 | 0xF4 | 中断过程 |
| AO | 245 | 0xF5 | 异常终止输出 |
| AYT | 246 | 0xF6 | 对方是否还在运行 |
| EC | 247 | 0xF7 | 擦除字符 |
| EL | 248 | 0xF8 | 擦除行 |
| GA | 249 | 0xF9 | 前进 |
| SB | 250 | 0xFA | 子选项开始 |
| WILL | 251 | 0xFB | 同意允许（enable）选项 |
| WONT | 252 | 0xFC | 拒绝允许选项 |
| DO | 253 | 0xFD | 认可选项请求 |
| DONT | 254 | 0xFE | 拒绝选项请求 |
| IAC | 255 | 0xFF | 解释（下一个字符）为控制 |

### 六.TELNET选项协商

TELNET允许客户与服务器在使用服务之前或使用服务过程中进行选项协商。对于复杂的终端用户，选项协商能提供额外的特性。对于较简单的终端用户，选项协商只能使用较少的特性。一些远程控制字符可用来定义选项。下表给出了一些常用的选项。

表14-2TLENET协商选项

| 代码 | 选项 | 意义 |
| --- | --- | --- |
| 0 | 二进制 | 使用8位二进制传输 |
| 1 | 回显 | 将在一端收到的数据回显到另一端 |
| 3 | 抑制前进 | 抑制数据后面的前进信号 |
| 5 | 状态 | 请求TELNET的状态 |
| 6 | 定时标记 | 定义定时标记 |
| 24 | 终端类型 | 设置终端类型 |
| 32 | 终端速率 | 设置终端速率 |
| 34 | 行方式 | 改变到行方式 |

下面是选项的描述：

●二进制：这个选项允许将收到的除IAC之外的每个8位字符解释为二进制数据。当收到IAC时，它的下一个或下几个字符被解释为命令。如果收到的两个连续的IAC字符，则丢弃第一个IAC字符，然后把第二个解释为数据。

●回显：这个选项允许服务器回显收到的来自客户的数据。客户向发送器发送的每一个字符都将回显到客户终端的屏幕上。这时，用户终端通常在字符被键入时并不回显，而要等到服务器收到它们后才回显。

●抵制前进：这个选项抑制前进字符（GA）。

●状态：这个选项允许用户或客户机器上运行的进程得到在服务器端被允许的选项状态。

●定时标记：这个选项允许一方发出定时标记，指出所有以前收到的数据都已被处理。

●终端类型：这个选项允许客户发送它的终端类型。

●终端速率：这个选项允许客户发送它的终端速率。

●行方式：这个选项允许客户切换到行方式。

#### 1.选项协商

要使用前面介绍的选项，就需要在客户与服务器之间进行选项协商。选项协商需要使用四种控制字符。这些字符如下表所示：

表14-3选项协商控制字符

| 字符 | 十进制 | 二进制 | 意义 |
| --- | --- | --- | --- |
| WILL | 251 | 11111011 | (1)提供允许选项  (2)接受请求允许选项 |
| WONT | 252 | 11111100 | (1)拒绝请求允许选项  (2)提供禁止选项  (3)接受禁止选项 |
| DO | 253 | 11111101 | (1)同意提供允许选项  (2)请求允许选项 |
| DONT | 254 | 11111110 | (1)不同意提供允许选项  (2)同意提供禁止选项  (3)请求禁止选项 |

(1)允许选项

一些选项仅能由服务器允许，另一些仅能由客户允许，还有一些则可由服务器或客户允许。选项要被允许需通过提供允许或请求允许来实现。

●提供允许

如果一方有权限，它可以提供允许选项。对方可以同意或不同意这个提供。提供方发送WILL命令，表示“我能允许这个选项吗？”另一方可以发送DO命令，表示“同意”或发送DONT命令，表示“不同意”。如下图所示：

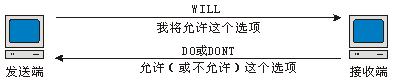


图14-6提供允许选项

●请求允许

一方可以请求另一方允许选项。另一方可以接收或拒绝这个请求。请求方发送DO命令，表示“请允许这个选项”。另一方可以发送WILL命令，表示“同意”，或发送WONT命令，表示“我不同意”。如下图所示：

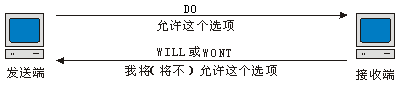


图14-7请求允许选项

(2)禁止选项

已经被允许的选项可以被一方禁止掉。可以通过提供禁止或请求禁止来禁止选项。

●提供禁止

一方可以提供禁止选项。另一方必须同意这个提供。提供方发送WONT命令，表示“我不再使用这个选项”。回答必须是DONT命令，表示“不要再使用这个选项”。如下图所示：

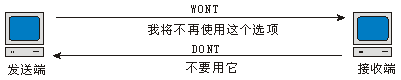


图14-8提供禁止一个选项

●请求禁止

一方可以请求另一方禁止选项。另一方必须接受这个请求。请求方发送DONT命令。表示“请不要再使用这个选项”。回答必须是WONT命令，表示“我不再使用它”。如下图所示：

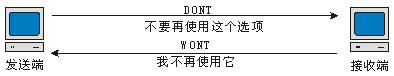


图14-9请求禁止选项

#### 2.子选项协商

一些选项需要附加的信息。例如，要定义终端的类型或速率，协商就要包括字符串或数字来定义类型或速率。下表所示的两个子选项字符是用来进行选项协商的。

表14-4子选项协商的NVT字符集

|  |  |  |  |
| --- | --- | --- | --- |
| 字符 | 十进制 | 二进制 | 意义 |
| SE | 240 | 11110000 | 子选项结束 |
| SB | 250 | 11111010 | 子选项开始 |

下图所示的例子中，客户将终端的类型设置为VT。



图14-10子选项协商的例子

### 七.FTP协议简介

FTP（文件传输协议）提供了一种通过TCP传送文件的方法，可以将一个文件从一个系统复制到另一个系统中。FTP使用两种TCP连接：一种是控制连接，一种是数据连接。控制连接一直持续到客户端和服务器端进程间的通信完成为止，用于传输控制命令，服务器使用21端口；数据连接根据通信的需要随时建立和释放，用于数据的传输，服务器常使用20端口。FTP的连接模式有两种：主动模式（PORT）和被动模式（PASV）。

### 八.FTP连接、通信与传送

#### 1.连接

FTP的控制连接和数据连接使用不同的方法和不同的端口号。

(1)控制连接

控制连接的创建步骤如下：

●服务器在熟知端口21发出被动打开，等待客户。

●客户使用临时端口发出主动打开。

在整个过程中这个连接一直是打开的。这是客户和服务器的交互式连接，所以IP协议使用的服务类型是最小延迟。用户键入命令并期望收到的响应延时不太大。下图给出了服务器和客户之间的初始连接：

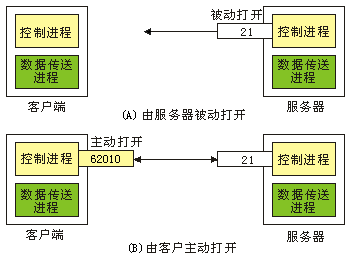


图14-11打开控制连接

(2)FTP数据连接的两种模式

FTP的数据连接存在两种模式：主动模式（PORT）和被动模式（PASV）。主动模式是从服务器端向客户端发起连接；被动模式是客户端向服务器端发起连接。

当FTP被设置为主动模式时，它的连接过程如下：首先客户端向服务器的FTP端口（默认是21）发送连接请求，服务器接受连接，建立一条控制连接。当需要传输数据时，客户端在控制连接上用PORT命令告诉服务器：“我打开了XXXX端口，你来连接我”。于是服务器从20端口向客户端的XXXX端口发送连接请求，最后建立一条数据连接来传输数据。

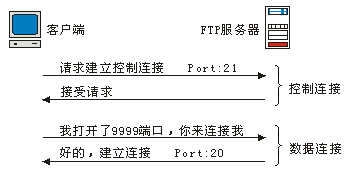


图14-12FTP的主动模式

当FTP被设置为被动模式时，它的连接过程如下：首先客户端向服务器的FTP端口（默认是21）发送连接请求，服务器接受连接，建立一条控制连接。当需要传输数据时，服务器在命令链路上用PASV命令告诉客户端：“我打开了XXXX端口，你来连接我”。于是客户端向服务器的XXXX端口发送连接请求，最后建立一条数据连接来传输数据。

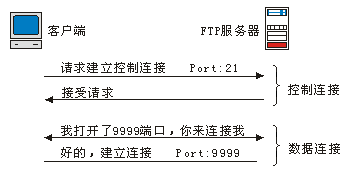


图14-13FTP的被动模式

当进行FTP连接时，InternetExplore通常被设置为被动模式，而FTP客户端软件（如：FlashFXP、CutFTP）一般为主动模式。如果服务器和客户端之间存在防火墙，主动模式经常会引起一些麻烦。例如：客户端位于防火墙之后，通常防火墙允许所有内部向外部的连接通过，但是对于外部向内部发起的连接却存在很多限制。在这种情况下，客户可以正常地和服务器建立控制连接，而如果使用主动模式的数据连接，一些数据传输命令就很难成功运行，因为防火墙会阻塞从服务器向客户端发起的数据传输连接。因此在使用主动模式的FTP数据连接时，防火墙上的配置会比较麻烦。

#### 2.通信

(1)在控制连接上的通信

FTP使用NVTASCII字符集在控制连接上通信，如下图所示。通信过程使用命令和响应来完成。命令和响应都是一个短行，因此不必考虑它的文件结构。每一行以回车和换行组成的结束记号结束。



图14-14使用控制连接

(2)在数据连接上的通信

数据连接用来传送数据，在传送数据之前，客户需要使用控制连接来做传输的准备。客户需要定义要传送的文件类型、数据结构以及传输方式。如下图所示：

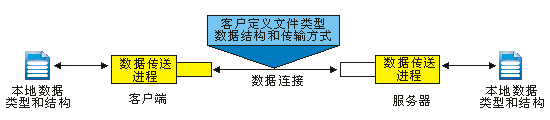


图14-15使用数据连接

①文件类型

FTP能够在数据连接上传送下列文件类型中的一种：

●ASCII文件：这是传送文本文件的默认格式。每一个字符使用NVTASCII进行编码。发送端把文件从它自己的表示转换成NVTASCII字符，而接收端从NVTASCII字符转换成它自己的字符。

●EBCDIC文件：若连接的一端或两端使用EBCDIC编码，则可使用EBCDIC编码传送文件。

●图像文件：这是传送二进制文件的默认格式。这种文件是作为连续的位流传送而没有任何解释或编码。在大多数情况下它用来传送二进制文件，如已编译的程序。

②数据结构

在数据连接上传送文件时，FTP可以使用下列数据结构中的一种：

●文件结构（默认）：这种文件没有结构。它是连续的字节流。

●记录结构：这种文件划分为一些记录，这只能用于文本文件。

●页面结构：这种文件划分为一些页面，每一个页面有页面号和页面首部。页面可以随机地或顺序地进行存取。

(3)传输方式

FTP可以使用下列传输方式之一在数据连接上传送文件：

●流方式：这是默认方式。数据作为连续的字节流从FTP交付给TCP。TCP负责把数据划分为适当大小的报文。若数据是简单的字节流（文件结构），就不需要文件结束符。若数据划分为记录（记录结构），则每一个记录有1字节的记录结束（EOR）字符，而在文件的结束处有文件结束（EOF）字符。

●块方式：数据可以按块从FTP交付给TCP。每一个块的前面有3字节首部。第一个字节叫做块描述符，后两个字节定义块的大小，以字节为单位。

●压缩方式：若文件很大，数据可进行压缩。通常使用的压缩方法是游程长度编码。数据单元的连续出现数可以用一个“出现”和“重复数”来替换。在文本文件中，这通常是空格。在二进制文件中，空字符常常被压缩。

#### 3.文件传送

在控制连接命令的控制下，在数据连接上进行文件传送，如下图所示：

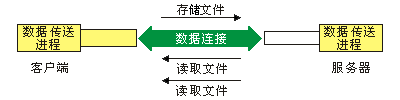


图14-16文件传送

FTP的文件传送有三种动作：

●从服务器把文件复制到客户端叫做读取文件。读取文件是在RETR命令的监督下完成的。

●从客户端把文件复制到服务器叫做存储文件。存储文件是在STOR命令的监督下完成的。

●从服务器向客户端发送目录列表或文件名。这是在LIST命令的监督下完成的。FTP把目录或文件名列表当作文件，在数据连接上发送。

### 九.FTP命令与响应

FTP使用控制连接在客户进程和服务器进程之间建立连接进行通信。在通信时，客户向服务器发送命令，服务器给客户返回响应，如下图所示：

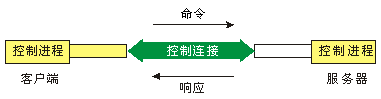


图14-17命令的处理

#### 1.命令

由FTP客户控制进程发送的命令形式是ASCII大写字符，后面的变量可以有，也可以没有。命令可以粗略地划分为6组：接口命令、文件管理命令、数据格式化命令、端口定义命令、文件传送命令以及杂项命令。关于这些命令的细节，请参考RFC765、RFC775和RFC959文档。

●接口命令：这些命令使用户能够接入到远程系统。下表列举了这个组的常用命令。

表14-5接口命令

| 命令 | 变量 | 说明 |
| --- | --- | --- |
| USER | 用户标识符 | 用户信息 |
| PASS | 用户口令 | 口令 |
| ACCT | 应付费的帐务 | 帐务信息 |
| REIN |  | 重新初始化 |
| QUIT |  | 向系统注销 |
| ABOR |  | 前面地命令异常终止 |

●文件管理命令：这些命令使用户接入到远程计算机的文件系统，允许用户使用目录结构、创建新的目录、删除文件，等等。下表给出了这个组的常用命令。

表14-6文件管理命令

| 命令 | 变量 | 说明 |
| --- | --- | --- |
| CWD | 目录名 | 改变到另一个目录 |
| CDUP |  | 改变到父目录 |
| DELE | 文件名 | 删除文件 |
| LIST | 目录名 | 列出子目录或文件 |
| NLIST | 目录名 | 列出子目录名或无其它属性的文件 |
| MKD | 目录名 | 创建新目录 |
| PWD |  | 显示当前目录名 |
| RMD | 目录名 | 删除目录 |
| XMKD | 目录名 | 创建新目录 |
| XRMD | 目录名 | 删除目录 |
| XPWD |  | 显示当前目录名 |
| XCUP |  | 改变到父目录 |
| RNFR | 文件名（旧文件名） | 标志要重新命名的文件 |
| RNTO | 文件名（新文件名） | 重新命名文件 |
| SMNT | 文件系统名 | 安装文件系统 |

●数据格式化命令：这些命令让用户定义数据结构、文件类型以及传输方式。所定义的格式可由文件传送命令来使用。下表给出了这个组的常用命令。

表14-7数据格式化命令

|  |  |  |
| --- | --- | --- |
| 命令 | 变量 | 说明 |
| TYPE | A（ASCII），E（EBCDIC），I（图像），  N（非打印），或T（TELNET） | 定义文件类型和当需要时定义打印格式 |
| STRU | F（文件），R（记录），或P（页面） | 定义数据的组织 |
| MODE | S（流），B（块），或C（压缩） | 定义传输方式 |

●端口定义命令：这些命令定义了客户端的数据连接的端口号。

使用PORT命令，客户可选择一个短暂端口号，并使用这个端口号被动打开，然后把该端口号发送给服务器。服务器就使用这个端口号创建主动打开。

使用PASV命令，客户仅要求服务器先选择一个端口号。服务器在那个端口进行被动打开，并在响应中发送端口号。客户使用端口号发出主动打开。下表给出了这个组的常用命令。

表14-8端口定义命令

|  |  |  |
| --- | --- | --- |
| 命令 | 变量 | 说明 |
| PORT | 6个数字的标识符 | 客户选择端口 |
| PASV |  | 服务器选择端口 |

●文件传送命令：这些命令可以让用户传送文件。下表给出了这个组的常用命令。

表14-9文件传送命令

| 命令 | 变量 | 说明 |
| --- | --- | --- |
| RETR | 文件名 | 读取文件；文件从服务器传送到客户。 |
| STOR | 文件名 | 存储文件；文件从客户传送到服务器。 |
| APPE | 文件名 | 与STOR相似，若文件存在就必须附加上数据。 |
| STOU | 文件名 | 与STOR相似，除了文件名在目录中必须是惟一的。但是，已存在的文件不能被重写。 |
| ALLO | 文件名 | 在服务器为文件分配存储空间。 |
| REST | 文件名 | 在指明的数据点给文件标记确定位置。 |
| STAT | 文件名 | 返回文件的状态。 |

●杂项命令：这些命令把信息交付给客户端的FTP用户。下表给出了这个组的常用命令。

表14-10杂项命令

|  |  |  |
| --- | --- | --- |
| 命令 | 变量 | 说明 |
| HELP |  | 询问关于服务器的信息 |
| NOOP |  | 检查服务器是否工作 |
| SITE | 命令 | 指明特定场所的命令 |
| SYST |  | 询问服务器使用的操作系统 |

●安全命令：这些命令提供了对认证，完整性和机密性的支持。下表给出了这个组的常用命令。

表14-11安全命令

| 命令 | 变量 | 说明 |
| --- | --- | --- |
| AUTH |  | 认证/安全机制 |
| ADAT |  | 认证/安全数据 |
| PROT |  | 数据信道保护等级 |
| PBSZ |  | 保护缓冲区大小 |
| CCC |  | 清除命令通道 |
| MIC |  | 完整性保护命令 |
| CONF |  | 保密性保护命令 |
| ENC |  | 私密性保护命令 |

#### 2.响应

每一个FTP命令都应该有响应。响应由两个部分组成：3个数字以及跟随在后面的文本。数字部分定义代码，文本部分定义所需的参数或额外的解释。数据部分中的每一个数字都有特殊的意义，将这3个数字记为xyz，下面描述每一个数字的意义。

(1)第一个数字定义命令的状态：

●1yz（正面初步回答）：动作已经开始。服务器在接受另一个命令之前将发送另一个回答。

●2yz（正面完成回答）：动作已经完成。服务器将接受另一个命令。

●3yz（正面中间回答）：命令已经接受，但进一步的信息是需要的。

●4yz（过渡负面完成回答）：动作没有发生，但差错是暂时的。同样的命令后面可以发送。

●5yz（永久负面完成回答）：命令没有接受，不能够再试。

(2)第二个数字定义命令的状态：

●x0z（语法）。

●x1z（信息）。

●x2z（连接）。

●x3z（鉴别和账号）。

●x4z（未指明）。

●x5z（文件系统）。

(3)第三个数字提供附加信息。下表给出了可能的响应的简短列表。

表14-12FTP响应

| 代码 | 说明 |
| --- | --- |
| 正面初步回答 | |
| 120 | 服务不久即将就绪 |
| 125 | 数据连接打开；数据传送不久即将开始 |
| 150 | 文件状态是OK；数据连接不久即将打开 |
| 正面完成回答 | |
| 200 | 命令OK |
| 211 | 系统状态或求助回答 |
| 212 | 目录状态 |
| 213 | 文件状态 |
| 214 | 求助报文 |
| 215 | 命名系统类型（操作系统） |
| 220 | 服务就绪 |
| 221 | 服务关闭 |
| 225 | 数据连接打开 |
| 226 | 关闭数据连接 |
| 227 | 进入被动方式；服务器发送它的IP地址和端口号 |
| 230 | 用户登录OK |
| 250 | 请求文件动作OK |
| 正面中间回答 | |
| 331 | 用户名OK；需要更多的信息 |
| 332 | 需要登录帐号 |
| 350 | 文件动作在进行中；需要更多的信息 |
| 过渡负面完成回答 | |
| 425 | 不能打开数据连接 |
| 426 | 连接关闭；不能识别的命名 |
| 450 | 未采取文件动作；文件不可用 |
| 451 | 动作异常终止；本地差错 |
| 452 | 动作异常终止；存储器不足 |
| 永久负面完成回答 | |
| 500 | 语法差错；不能识别的命令 |
| 501 | 参数或变量的语法差错 |
| 502 | 命令未实现 |
| 503 | 不良命令序列 |
| 504 | 命令参数未实现 |
| 530 | 用户未登录 |
| 532 | 存储文件需要帐号 |
| 550 | 动作未完成；文件不可用 |
| 552 | 请求的动作异常终止；超过存储器的分配 |
| 553 | 未采取请求动作；文件名不允许 |

【实验步骤】

练习1运行TELNET命令，捕获数据并分析

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习一人一组，现仅以主机A为例，其它主机参考主机A的操作。

1.主机A启动协议分析器进行数据捕获，并设置过滤条件（提取TELNET协议）。

2.主控中心平台（IP地址：172.16.0.253）上的telnet服务已经启动，使主控中心平台为本小组提供的帐号，其用户名：group1\_1，密码：group1\_1。

[注]用户名、密码相同，生成规则是：groupx\_y（x是组索引，y是主机索引，eg第一组的主机C使用的用户名和密码为：group1\_3）。

主机A在命令行提示符下运行：

(1)telnet172.16.0.253。

(2)在“Login：”提示符后输入用户名（group1\_1）。在“Password：”提示符后输入密码（group1\_1）。

(3)在虚拟终端上进行一些简单的操作（可不作）。

(4)按“CTRL+]”回到telnet提示符下。

(5)输入“quit”退出telnet。

[注] telnet如果本机未安装，请点击开始→控制面板→程序和功能→打开或关闭Windows功能，将前面对号打钩，点击确定便可。

3.察看主机A捕获的数据，分析TELNET的工作过程。

●TELNET使用的TCP端口是\_\_\_\_\_。

●找出与选项协商有关的数据包，分析客户端与服务器端进行选项协商的过程。

●在捕获的数据中，查找用户名和密码。

●结合分析结果，绘制TELNET交互图。

●理解TELNET明文传输的不安全性，了解网络监听的可能。

练习2TELNET选项协商的过程

本练习一人一组，现仅以主机A为例，其它主机参考主机A的操作。

1.主机A启动协议分析器进行数据捕获，并设置过滤条件（提取TELNET协议）。

2.主机A首先要与TELNET服务器建立一个TCP连接：

(1)主机A上启动实验平台工具栏中的“TCP工具”。

(2)选中“客户端”单选框，在“地址”中填入服务器IP地址(默认172.16.0.253)。

(3)在“端口”中填入TELNET协议的端口号23。

(4)点击[连接]按钮进行连接。

3.使用TELNET的NVT字符集实现选项协商。

在发送数据（十六进制）窗口编辑并发送以下数据：

FFFB18FFFB1F点击[发送]按钮；

FFFC20FFFC23FFFB27点击[发送]按钮；

FFFD03点击[发送]按钮；

FFFB01FFFE05FFFC21点击[发送]按钮；

4.点击[断开]按钮，断开主机A与服务器的TCP连接。

5.察看主机A捕获到的数据，分析选项协商的过程。

●写出步骤3中每个十六进制字符对应的NVT字符。

●说明步骤3中发送的每个数据各自的作用和服务器端对其应答的内容。

思考问题：

1.远程登录TELNET的主要特点是什么？什么叫做虚拟终端NVT？

练习3FTP的工作过程

本练习一人一组，现仅以主机A为例，其它主机参考主机A的操作。

1.主机A启动协议分析器进行数据捕获并设置过滤条件（提取FTPdata和FTPcontrol协议）。

2.主机A登录FTP服务器：

在实验环境中的服务器（默认为172.16.0.253）已经启动，并提供一个公共帐号，用户名是：anonymous，口令：无。

在命令行提示符下运行：

(1)C:\>ftp172.16.0.253

(2)在“User：”提示符后输入用户名：anonymous

(3)在“Password：”提示符后输入密码：无

(4)在客户端上运行一个简单的操作，如：ftp>dir

(5)在FTP提示符下输入“quit”退出FTP

3.察看主机A捕获的数据，保存会话命令（方法：会话交互视图/单击右键/保存会话命令菜单，保存为FTP.txt），并分析FTP的工作过程：

●FTP使用的TCP端口有哪些\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

●分析FTP报文格式。指出在捕获数据报文中含有用户名、密码的报文，是否可以看到用户名和密码？说明FTP的安全性？

●分析FTP的工作过程，分析控制连接和数据连接是如何工作的，绘制出控制连接和数据连接的交互过程图。

思考问题：

1.文件传送协议FTP的主要工作过程是怎样的？主进程和从属进程各起什么作用？

练习4使用TCP连接工具与服务器进行命令交互

本练习一人一组，现仅以主机A为例，其它主机参考主机A的操作。

1.主机A启动协议分析器开始捕获数据并设置过滤条件（提取TCP协议）。

2.主机A启动TCP工具连接FTP服务器。

(1)主机A启动“实验平台工具栏中的TCP工具”。

①选中“客户端”单选框。

②在“地址”文本框中填入FTP服务器的IP地址。

③在“端口”文本框中填入主机FTP服务器进程的端口号21。

④点击“连接”按钮，建立与FTP服务器的TCP连接。

(2)连接成功（将该次连接记为w\_cmd），在接收窗口会显示成功连接的信息；若不成功，再次尝试进行连接，直到成功。如图：



图14-18TCP连接工具界面

3.使用TCP连接工具与服务器进行命令交互：

「注」

①<CRLF>是回车换行；

②文件名的生成规则是：file\_x(x是组索引，eg：第一组使用的文件名为：file\_1)。

③用户名、密码相同，生成规则是：groupx\_y（x是组索引，y是主机索引，eg第一组的主机C使用的用户名和密码为：group1\_3）。

(1)w\_cmd的发送窗口：USER用户名<CRLF>点击“发送”；

●服务器回复的信息？

(2)w\_cmd的发送窗口：PASS密码<CRLF>点击“发送”；

●服务器回复的信息？

进入FTP服务器，与FTP服务器进行命令交互；

(3)w\_cmd的发送窗口：SYST<CRLF>点击“发送”；（询问服务器使用的操作系统）

●服务器回复的信息？

(4)w\_cmd的发送窗口：PWD<CRLF>点击“发送”；（显示当前路径）

●服务器回复的信息？

(5)w\_cmd的发送窗口：TYPEA<CRLF>点击“发送”；（定义文件类型为ASCII）

●服务器回复的信息？

(6)w\_cmd的发送窗口：PORTx1,x2,x3,x4,x5,x6<CRLF>点击“发送”；

「注意」上面x1-x4的取值的规则为x1,x2,x3,x4携带的是本机的IP地址，x5,x6为端口值，计算方法为x5\*256+x6。例如本机的IP地址为172.16.0.35，则x1、x2、x3、x4分别为172、16、0、35，端口号为1025，则x5为4，x6为1。本实验使用端口为2570，所以x5为10，x6为10。

●服务器回复的信息？

●PORT命令对端口信息是怎样传递的？由上面发送的信息可知客户端使用的端口值为？

●解释一下PORT的格式，及该命令的用途。

●FTP服务器是使用什么方式创建数据连接的？

(7)再次运行TCP连接工具，按下图的内容填写数据，按“创建”键，进入等待远程连接的侦听状态中（将该次创建的连接记为w\_data1）；



图14-19以服务器方式进行连接

(8)w\_cmd的发送窗口：STOR文件名<CRLF>点击“发送”；（存储文件）

●服务器回复的信息？

(9)w\_data1的发送窗口：filedata点击“发送”，再按“断开”键关闭w\_data1；

(10)w\_cmd的发送窗口：PASV<CRLF>点击“发送”；

●服务器回复的信息？

●说明该命令的用途。

●FTP服务器是使用什么方式创建数据连接的？

●在服务器回复的信息中可知服务器端使用的端口为？（记为port）。

(11)再次运行TCP工具，将其端口值21改为port的值，按“连接”键，进入FTP数据传输窗口（将该次连接记为w\_data2）；

(12)w\_cmd的发送窗口：RETR文件名<CRLF>点击“发送”；（读取文件）

●服务器回复的信息？

(13)察看w\_data2返回信息，并将其关闭；

●其内容是步骤(9)输入的信息吗？

(14)w\_cmd的发送窗口：QUIT<CRLF>点击“发送”；（退出—终止命令连接）

●服务器回复的信息？

4.分析端停止数据捕获，并分析捕获的数据：

察看“FTP会话”各会话中，建立TCP连接、数据传输、释放TCP连接的过程。

(1)找出含有“PORT”命令的数据包，结合上下文理解该命令的作用。

(2)找出含有“PASV”命令的数据包，结合上下文理解该命令的作用。

(3)找出步骤2中涉及的数据连接（在“FTP会话”中，由步骤3的(10)步port的值及20端口所区分的两个会话），分析数据传输的过程。

5.结合该实验的操作过程及实验结果，分别绘制出主动模式下、被动模式下FTP上传数据、下载数据的会话交互图。

思考问题：

1.FTP的数据连接存在两种模式：主动模式和被动模式，说明各自的工作过程。如果服务器和客户端之间存在防火墙，使用哪种模式会引起一些麻烦？

## 实验15电子邮件协议(SMTP、POP3和IMAP)

【实验目的】

1.掌握邮件服务的工作原理

2.掌握SMTP、POP3、IMAP的工作过程

3.了解SMTP、POP3、IMAP协议的命令和使用方法

【学时分配】

4学时

【实验环境】

该实验采用网络结构一

【实验原理】

### 一.电子邮件简介

电子邮件（ElectronicMail，简称E-mail）是一项普遍使用的服务。电子邮件简单的说就是通过网络来邮寄的信件，是传统邮件的电子化。电子邮件的成本比邮寄普通信件低得多；而且投递速度很快，不管多远，只要几秒钟就能完成投递；另外，电子邮件使用起来也很方便，无论何时何地，都可以通过Internet发送电子邮件，或者打开自己的信箱阅读别人发来的邮件。

电子邮件地址的形式如“abc@xyz.com”，符号@是电子邮件地址的专用标识符，它前面的部分是信箱名称，后面的部分是信箱所在的位置。这就好比信箱abc放在“邮局”xyz.com里。当然这里的邮局是Internet上的一台用来收信的计算机，当收信人取信时，就把自己的电脑连接到这个“邮局”，打开自己的信箱，取走自己的信件。

电子邮件是一种“存储转发式”服务，而不是“终端到终端”的服务。这正是电子信箱系统的核心，利用存储转发可进行非实时通信，属异步通信方式。即信件发送端可随时随地发送邮件，不要求接收者同时在场，即使对方现在不在，仍可将邮件立刻送到对方的信箱内，且存储在对方的电子邮箱中。接收者可在他认为方便的时候读取信件，不受时空限制。在这里，“发送”邮件意味着将邮件放到收件人的信箱中，而“接收”邮件则意味着从自己的信箱中读取信件，信箱实际上是由文件管理系统支持的—个实体。

### 二.电子邮件传输过程

从发件人到收件人之间的邮件传输过程由三个阶段构成：

第一阶段：电子邮件从用户代理进入本地服务器。邮件并不是直接传送到远程服务器中的，因为远程服务器不能保证始终可用。所以，邮件在发送前会一直保存在本地服务器中。用户代理使用SMTP客户端软件，而本地服务器使用SMTP服务器软件。

第二阶段：电子邮件由本地服务器中继传递。在这一阶段，远程服务器作为SMTP服务器，而本地服务器作为SMTP客户端。电子邮件分发给远程服务器，而不是远程用户代理。原因是SMTP报文必须由始终处于运行状态的服务器接收，服务器必须不间断运行的原因是由于邮件可能随时到达。电子邮件由这台邮件服务器接收，存储在用户的邮箱中，便于以后检索。

第三阶段：远程用户代理使用邮件访问协议，如POP3或者IMAP，来访问邮箱并获取邮件。

### 三.SMTP协议简介

SMTP（简单邮件传输协议）是Internet中发送电子邮件的标准协议。SMTP是一个推送协议，它负责将邮件从客户推送到邮件服务器。SMTP报文被TCP协议封装，使用熟知端口25进行通信。

### 四.SMTP会话命令与响应

SMTP使用一些命令和响应在MTA（报文传送代理）客户和MTA服务器之间传送报文。每一个命令或响应都以回车和换行组成的行结束标记终止。如下图所示：

说明: 15-1

图15-1命令和响应

#### 1.命令

命令是从客户发送到服务器的。命令的格式如下图所示：

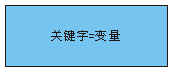


图15-2命令的格式

命令包括关键词，后面跟着零个或多个变量。SMTP定义了21个命令。前5个是强制性的，每一种SMTP实现都必须支持这5种命令。之后4种是常用的，并且是高度推荐的；最后12种很少使用。SMTP的命令如下表所示：

表15-1SMTP命令

| 关键词 | 变量 |
| --- | --- |
| HELO | 发送方的主机名 |
| MAILFROM | 发件人 |
| RCPTTO | 预期的收件人 |
| DATA | 邮件的主体 |
| QUIT |  |
| RSET |  |
| VRFY | 需要验证的收件人名字 |
| NOOP |  |
| AUTH |  |
| TURN |  |
| EXPN | 需要扩展的邮件发送清单 |
| HELP | 命令名 |
| SENDFROM | 预期的收件人 |
| SMOLFROM | 预期的收件人 |
| SMALFROM | 预期的收件人 |
| DELIVERBY |  |
| PIPELINING |  |
| CHUNKING |  |
| STARTTLS |  |
| NO-SOLICITING |  |
| CONPERM |  |

#### 2.响应

响应是从服务器发送到客户端的。响应是三个数字，后面可以跟着附加的文本信息。第一位数字的意义如下：

●2yz（正面完成回答）：若第一位数字是2（数字1现在已不使用），就表示所请求的命令已经成功地完成了，新命令可以开始。

●3yz（正面中间回答）：若第一位数字是3，就表示所请求的命令已被接受，但在完成请求的命令之前，收件人需要更多的一些信息。

●4yz（过渡负面完成回答）：若第一位数字是4，就表示所请求的命令已被拒绝，但差错条件是暂时的。这个命令可重新发送。

●5yz（永久负面完成回答）：若第一位数字是5，就表示所请求的命令已被拒绝。这个命令不能重新发送。

第二位和第三位数字提供关于响应的更详细的内容。下表列举了某些响应。

表15-2SMTP响应

| 代码 | 说明 |
| --- | --- |
| 正面完成回答 | |
| 211 | 系统状态或求助回答 |
| 214 | 求助报文 |
| 220 | 服务就绪 |
| 221 | 服务关闭传输信道 |
| 250 | 请求命令完成 |
| 251 | 用户不是本地的；报文将被转发 |
| 正面中间回答 | |
| 354 | 开始邮件输入 |
| 过渡负面完成回答 | |
| 421 | 服务不可用 |
| 450 | 邮箱不可用 |
| 451 | 命令异常终止；本地差错 |
| 452 | 命令异常终止；存储器不足 |
| 永久负面完成回答 | |
| 500 | 语法差错；不能识别的命令 |
| 501 | 语法的参数或变量差错 |
| 502 | 命令未实现 |
| 503 | 命令序列不正确 |
| 504 | 命令暂时未实现 |
| 550 | 命令未执行；邮箱不可用 |
| 551 | 用户非本地的 |
| 552 | 所请求的动作异常终止；存储位置超过 |
| 553 | 所请求的动作未发生；邮箱名不允许用 |
| 554 | 事务失败 |

### 五.SMTP运行过程

邮件报文的传送共有3个阶段：连接建立、报文传送和连接终止。

#### 1.连接建立

当客户与服务器的熟知端口25建立了TCP连接后，SMTP服务器就开始它的连接阶段。这个阶段包括以下3个步骤，如下图所示：

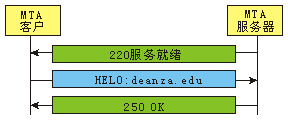


图15-3连接建立

(1)服务器发送代码220（服务就绪）告诉客户它已准备好接收邮件。若服务器未就绪，它就发送代码421（服务不可用）。

(2)客户HELO报文，使用它的域名地址标志自己，用来把客户的域名通知服务器。在TCP的连接建立阶段，发送方和接收方是通过它们的IP地址来知道对方的。

(3)服务器响应代码250（请求命令完成）或根据情况使用其它一些代码。

#### 2.报文传送

在SMTP客户与服务器之间建立连接后，发件人就可以与一个或多个收件人交换报文了。这个阶段包括8个步骤。若收件人超过一个，则步骤(3)和(4)将重复进行，如下图所示：

(1)客户发送MAILFROM报文介绍报文的发送者。报文中包括发件人的邮件地址和域名。在报文中包括发件人的邮件地址是因为这个地址可以被服务器使用，用于返回差错报文或报告报文。

(2)服务器发送响应代码250或其它适当的代码。

(3)客户发送RCPT（收件人）报文，包括收件人的邮件地址。

(4)服务器发送响应代码250或其它适当的代码。

(5)客户发送DATA报文对报文的传送进行初始化。

(6)服务器发送响应代码354（开始邮件输入）或其它适当的报文。

(7)客户用连续的行发送报文的内容。每一行以二字符的行结束标记（回车和换行）终止。整个报文以仅有一个点的行结束。

(8)服务器发送响应代码250（OK）或其它适当的代码。

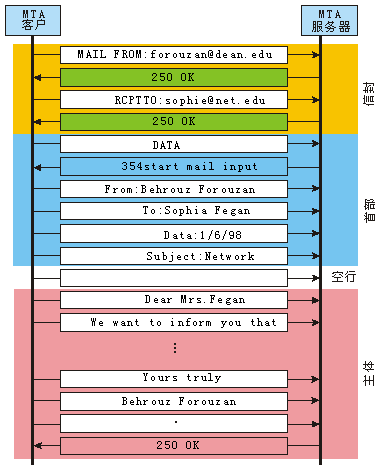


图15-4报文的传送

#### 3.连接终止

在报文传送成功后，客户端就终止连接。这个阶段包括两个步骤。如下图所示：

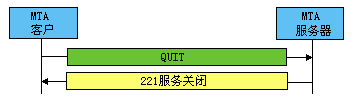


图15-5连接终止

(1)客户发送QUIT命令。

(2)服务器响应代码221或其它适当的代码。

在SMTP连接终止后，TCP连接也就关闭了。

### 六.POP3和IMAP简介

POP3（邮局协议版本3）协议和IMAP（因特网邮件访问协议版本4）协议是拉取协议，操作由收件人发起。邮件在收件人检索之前保存在邮件服务器的邮箱中。POP3报文被TCP协议封装，使用熟知端口110进行通信。IMAP报文也被TCP协议封装，使用熟知端口143进行通信。

### 七.POP3会话命令

表15-3POP3会话命令

|  |  |
| --- | --- |
| POP3命令 | 意义 |
| USER | 用户 |
| PASS | 密码 |
| APOP | Digest是MD5消息摘要 |
| STAT | 请求服务器发回关于邮箱的统计资料，如邮件总数和总字节数 |
| UIDL | 返回邮件的惟一标识符，POP3会话的每个标识符都将是惟一的 |
| LIST | 返回邮件数量和每个邮件的大小 |
| RETR | 返回由参数标识的邮件的全部文本 |
| DELE | 服务器将由参数标识的邮件标记为删除，由quit命令执行 |
| RSET | 服务器将重置所有标记为删除的邮件，用于撤销DELE命令 |
| TOP | 服务器将返回由参数标识的邮件前n行内容 |
| NOOP | 服务器返回一个肯定的应答 |
| CAPA | 返回POP3服务器支持的功能列表 |
| QUIT | 退出操作 |

### 八.POP3运行过程

当用户需要阅读电子邮件时，就从邮件服务器的邮箱中下载电子邮件。客户使用TCP端口110与服务器建立连接，然后通过用户名和口令访问邮箱。用户可以列出清单，然后逐个读取邮件报文，下图给出了使用POP3协议读取邮件的例子：

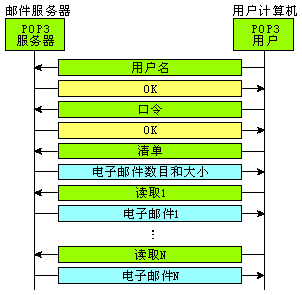


图15-6POP3运行过程

POP3有两种方式：删除方式和保存方式。删除方式在每一次读取邮件后就把邮箱中的邮件删除。保存方式是在读取邮件后仍然在邮箱中保存这个邮件。删除方式通常用在用户使用固定计算机工作的情况，用户在读取或回答邮件后可以保存或整理所收到的邮件。保存方式通常用在用户离开计算机时来读取邮件，邮件读取后还保存在服务器中，供日后读取和整理。

### 九.IMAP会话命令

IMAP协议的命令与POP3协议的命令是不同的，在IMAP中每条命令都有一个由客户指定的标签。客户发出的每条命令都有不同的标签，服务器使用命令的标签作为应答的标签。这样IMAP客户就可以同时送出多个命令，而服务器可以并发地处理这些命令，不必等待上一个命令执行完毕才处理下一个。

表15-4IMAP会话命令

| 标签 | 命令 | 变量 |
| --- | --- | --- |
| TAG | LOGIN | 用户名和口令 |
| TAG | CREATE | 可以创建指定名字的文件夹 |
| TAG | DELETE | 删除指定名字的文件夹 |
| TAG | RENAME | 重命名前的文件夹的名称，重命名后的文件夹的名称。 |
| TAG | LIST | 起始的路径位置和需要列出的文件夹所符合的特征。 |
| TAG | APPEND | 新邮件的属性、大小，随后是邮件数据。 |
| TAG | SELECT | 文件夹的名称。 |
| TAG | FETCH | FETCH命令是IMAP协议里最复杂的命令。FETCH的命令参数很多、很复杂，难以一一列举，但基本的特征是允许将邮件按照MIME结构拆解为零碎的部件来提取。例如，可以利用FETCH命令提取邮件头、某一个附件、或某一邮件附件头部的某一字段，等等。 |
| TAG | STORE | 指定邮件的属性。 |
| TAG | CLOSE |  |
| TAG | EXPUNGE |  |
| TAG | LOGOUT |  |
| TAG | UNSELECT |  |
| TAG | SETACL |  |
| TAG | EXPUNGE |  |

### 十.IMAP运行过程

IMAP客户首先建立一个到IMAP服务器143端口的TCP连接，服务器返回初始问候消息，然后客户与服务器就可以交互了。客户与服务器的交互与POP3协议类似，不过要丰富些。IMAP服务器在会话期间会处于以下4种状态之一：未认证（nonauthenticated）、已认证（authenticated）、已选择（selected）和注销（1ogout）。未认证状态是连接刚建立时的初始状态，这种状态下，用户必须提供一个用户名和口令才能发出更多的命令。在已认证状态下，用户必须选择一个邮件夹才能发出作用于邮件消息的命令。在已选择状态下，用户可以发出作用于邮件消息的任何命令（获取、转移、删除、获取多部分消息的某个部分，等等）。最后的注销状态是会话即将终止时的状态。如下图所示：

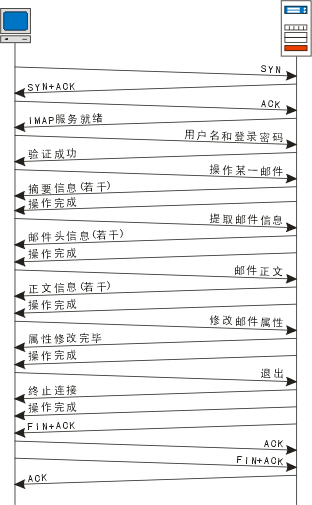


图15-7IMAP的运行过程

### 十一.POP3协议与IMAP协议的区别

POP3提供了快捷的邮件下载服务，用户可以利用POP3把邮箱里的信下载到主机上进行离线阅读。一旦邮件进入主机的本地硬盘，就可以选择把邮件从服务器上删除，然后脱离Internet的连接并选择在任何时候阅读已经下载的邮件。

IMAP同样提供了方便的邮件下载服务，让用户服务能进行离线阅读，但IMAP能完成的却远远不只这些。首先，IMAP提供的摘要浏览功能可以让你阅读完邮件的到达时间、主题、发件人、大小等信息后才作出是否下载的决定。你不必等所有的邮件都下载完毕后才知道邮件里都有些什么。如果你根据摘要信息就可以决定某些邮件对你毫无用处，你就可以直接在服务器上把这些邮件删除掉。

【实验步骤】

练习1使用Live Mail发送电子邮件

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习一人一组，现仅以主机A为例，其它主机参考主机A的操作。

1.主机A启动协议分析器进行数据捕获，并设置过滤条件（提取SMTP协议）。

2.主机A打开邮件工具Live Mail并设置一个邮件帐号，邮件接收服务器为POP3，编辑并发送一封电子邮件给主机A。

「注」依次点击：开始-所有程序-Windows Live,启动邮件工具。Live Mail使用方法及具体配置见附录。

3.察看主机A捕获的数据，保存会话命令（方法：会话交互视图/单击右键/保存会话命令菜单，保存为Smtp.zdt）。

(1)SMTP使用的TCP端口是\_\_\_\_\_。

(2)找出传输用户帐号及密码的数据包，能不能看到用户密码？

(3)结合Smtp.zdt的SMTP传输过程，填写下表。

表15-5实验结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SMTP连接建立过程 | 源IP | 目的IP | 报文摘要和参数 | 报文作用 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| 邮件传输过程 | 源IP | 目的IP | 报文摘要和参数 | 报文作用 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| SMTP连接释放过程 | 源IP | 目的IP | 报文摘要和参数 | 报文作用 |
|  |  |  |  |
|  |  |  |  |

(4)找到传输邮件内容的第一个报文，即包含很多HeaderField字段的Smtp报文：

●邮件的内容首部包括一些关键信息，如From、Subject、To等，请分析邮件首部格式。

●分析在邮件的首部字段中有关MIME的信息。

练习2使用Live Mail接收电子邮件

本练习一人一组，现仅以主机A为例，其它主机参考主机A的操作。

1.使用POP3协议接收

(1)主机A启动协议分析器进行数据捕获，并设置过滤条件（提取POP3协议）。

(2)主机A接收练习一中发送的邮件。

(3)察看主机A捕获的数据，保存会话（方法:会话交互视图/单击右键/保存会话命令菜单，保存为POP3.zdt）。

●POP3使用的TCP端口是\_\_\_\_\_。

●找出传输用户帐号及密码的数据包，能不能看到用户密码？

●结合POP3.zdt的内容，分析POP3协议的工作过程并画出邮件接收过程简图，比较它与SMTP协议的不同点。

●找到接收邮件内容的报文（命令RETR的响应报文），察看是否与发送的内容一致？

2.使用IMAP协议接收

(1)主机A重新进行数据捕获，并设置过滤条件（提取IMAP协议）。

(2)主机A使用Live Mail设置一个邮件账号，邮件接收服务器为IMAP。发送一封邮件并对其接收。

(3)察看主机A捕获的数据，保存会话(会话交互视图/单击右键/保存会话命令菜单，保存为IMAP.zdt）。

●IMAP使用的TCP端口是\_\_\_\_\_。

●找出传输用户帐号及密码的数据包，能不能看到用户密码？

●结合IMAP.zdt的内容，分析IMAP协议的工作过程并画出邮件接收过程简图，比较它与POP3协议的异同点。

练习3使用TCP工具和SMTP命令实现邮件发送

本练习一人一组，现仅以主机A为例，其它主机参考主机A的操作。

「说明」邮件服务器提供给主机A的帐号和密码均为group1\_1。

1.主机A启动协议分析器进行数据捕获，并设置过滤条件（提取SMTP协议）。

2.主机A首先要与邮件服务器建立一个TCP连接（实验室已建立一个邮件服务器，地址默认为172.16.0.253，邮件服务器主机名：JServer，邮件服务器域名：NetLab）。

(1)主机A上启动实验平台工具栏中的“TCP工具”。选中“客户端”单选框，在“地址”文本框中填入服务器IP地址（默认为172.16.0.253）；在“端口”文本框中填入SMTP协议端口号（25）；点击[连接]按钮进行连接。

(2)若连接成功，在显示数据窗口会显示成功连接的信息：220。

(3)若不成功，查看IP地址和端口号是否有错，再次尝试进行连接，直到成功。

3.用SMTP命令编辑并发送邮件。

(1)在发送窗口编辑发送SMTP协议的命令，在接收窗口会返回服务器端信息：

「注」<CRLF>是回车换行；[服务器返回的文本信息] 是邮件服务器返回的信息，不同的时候可能会有所不同，但只要返回代码对应正确即说明操作成功。

发送窗口：helogroup1\_1<CRLF>点击[发送]按钮；

接收窗口：250[服务器返回的文本信息]

发送窗口：mailfrom:<group1\_1@JServer.NetLab><CRLF>点击[发送]按钮；

接收窗口：250[服务器返回的文本信息]

发送窗口：rcptto:<group1\_1@JServer.NetLab><CRLF>点击[发送]按钮；

接收窗口：250[服务器返回的文本信息]

发送窗口：data<CRLF>点击[发送]按钮；

接收窗口：354[服务器返回的文本信息]

发送窗口：myemail<CRLF>（Myemail是邮件内容）点击[发送]按钮；

接收窗口：

发送窗口：.<CRLF>点击[发送]按钮；

接收窗口：250[服务器返回的文本信息]

发送窗口：quit<CRLF>点击[发送]按钮。

接收窗口：221[服务器返回的文本信息]

(2)观察TCP连接客户端的接收信息窗口中返回的信息。

4.察看主机A捕获的数据。

●会话分析的过程与练习一的过程相同吗？

●加深理解步骤3所使用的SMTP命令的使用方法和用途。

●在编写邮件内容时，我们只是简单发送“myemail”字符串，能否尝试添加邮件的内容首部的一些关键信息，如From、Subject、To等及有关MIME的信息。

练习4使用TCP工具和POP3命令实现邮件接收

本练习一人一组，现仅以主机A为例，其它主机参考主机A的操作。

「说明」邮件服务器提供给主机A的帐号和密码均为group1\_1。

1.主机A启动协议分析器进行数据捕获，并设置过滤条件（提取POP3协议）。

2.主机A与邮件服务器建立一个TCP连接，在“地址”中填入服务器IP地址（默认为172.16.0.253）；在“端口”中填入POP3协议端口号（110）。

3.用POP3命令实现邮件的接收。在发送数据窗口编辑发送POP协议的命令，在显示数据窗口察看服务器端返回的信息：

「注」<CRLF>是回车换行；[服务器返回的文本信息]文本出现出均为邮件服务器返回的信息，不同的时候可能会有所不同，但只要返回代码对应正确即说明操作成功。

发送窗口：usergroup1\_1<CRLF>点击[发送]按钮；

接收窗口：+OK[服务器返回信息]

发送窗口：passgroup1\_1<CRLF>点击[发送]按钮；

接收窗口：+OK[服务器返回信息]

发送窗口：stat<CRLF>点击[发送]按钮；

接收窗口：+OK[服务器返回信息]

发送窗口：uidl<CRLF>点击[发送]按钮；

接收窗口：+OK[服务器返回信息]

发送窗口：list<CRLF>点击[发送]按钮；

接收窗口：+OK[服务器返回信息]

发送窗口：retr1<CRLF>（查看第一封信件内容）点击[发送]按钮；

接收窗口：+OK[服务器返回信息]

在接收窗口查看第1封信的内容；

发送窗口：dele1<CRLF>点击[发送]按钮；

接收窗口：+OK[服务器返回信息]

发送窗口：quit<CRLF>点击[发送]按钮；

接收窗口：+OK[服务器返回信息]

4.察看主机A捕获的数据，加深理解步骤3所使用的POP3命令的使用方法和用途。

思考问题：

1.电子邮件系统使用TCP协议传送邮件。为什么有时我们会遇到邮件发送失败的情况？为什么有时对方会收不到我们发送的邮件？

2.通过实验说明你的电子邮件在网络上传输是安全的吗？为什么？你认为实现邮件安全传输的最好的办法是什么？

练习5使用TCP工具和IMAP命令实现邮件接收

本练习一人一组，现仅以主机A为例，其它主机参考主机A的操作。

「注」本练习开始前应确保主机A使用的账号group1\_1建立在IMAP邮件服务器上。

1.主机A使用live mail编辑并发送一封电子邮件给主机A

2.主机A启动协议分析器进行数据捕获，并设置过滤条件（提取IMAP协议）。

3.主机A与邮件服务器建立一个TCP连接，在“地址”中填入服务器IP地址（默认为172.16.0.253）；在“端口”中填入IMAP协议端口号（143）。

4.用IMAP命令实现邮件的接收。

在发送数据窗口编辑发送IMAP协议的命令，在显示数据窗口察看服务器端返回的信息。

「注」<CRLF>是回车换行。

发送窗口：00A1CAPABILITY<CRLF>点击[发送]按钮；

发送窗口：00A2LOGINgroup1\_1group1\_1<CRLF>点击[发送]按钮；

发送窗口：00A3SELECT"INBOX"<CRLF>点击[发送]按钮。记录服务器返回信息中的UIDNEXT后面的数字为“x”。

发送窗口：00A4UIDFETCH1:xFLAGS<CRLF>点击[发送]按钮。其中x为上一步记录的“x”，记录服务器返回结果的第一行尾UID（邮件唯一标记）后面的数字为“y”。

发送窗口：00A5UIDFETCHy(BODY.PEEK[]UID)<CRLF>点击[发送]按钮。其中y为上一步记录的“y”，取UID为“y”的邮件全部内容。

发送窗口：ZZZZLOGOUT<CRLF>点击[发送]按钮；

5.察看主机A捕获的数据，加深理解步骤3所使用的IMAP命令的使用方法和用途。

在本练习中是将邮件所有内容一次全部取出，尝试利用FETCH命令提取邮件头或某一邮件附件头部的某一字段。

## 实验16NetBIOS应用及SMB/CIFS协议

【实验目的】

1.掌握NetBIOS报文格式

2.了解NetBIOS使用的端口

3.理解WINS协议

4.了解SMB/CIFS协议

【学时分配】

2学时

【实验环境】

该实验采用网络结构一

【实验原理】

一.NetBIOS简介

NetBIOS（网络基本输入输出系统）是由IBM公司开发的。NetBIOS定义了一种软件接口以及在应用程序和连接介质之间提供通信接口的标准方法。NetBIOS是一种会话层协议，应用于各种局域网（Ethernet、TokenRing等）和广域网环境。

NetBIOS使得应用程序无需了解包括差错恢复（会话模式）在内的网络细节。NetBIOS请求以NCB（网络控制块）的形式提供，NCB中包含了信息存放位置和目标名称等信息。

NetBIOS提供开放系统互联（OSI）模型中的会话层和传输层服务，但不支持标准帧或数据格式的传输。NetBIOS扩展用户接口（NetBEUI）支持标准帧格式，它为NetBIOS提供网络层和传输层服务支持。NetBIOS支持两种通信模式：会话（session）模式和数据报（datagram）模式。会话模式是指两台计算机为“对话”建立一个连接，允许处理大量信息，并支持差错监测和恢复功能。数据报模式面向“无连接”（信息独立发送）操作，发送的信息较小，由应用程序提供差错监测和恢复功能。此外数据报模式也支持将信息广播到局域网中的每台计算机上。

NetBIOS名字为16字节长（必要情况下使用填充位填满），对使用的字节值几乎没有限制。对于不执行路由的小型网络，将NetBIOS名字映射到IP地址上有三种方法：

IP广播：当目标地址不在本机缓存上时，广播一个包含目标计算机NetBIOS名字的数据包。目标计算机返回其IP地址；

lmhosts文件：这是一个负责映射IP地址和NetBIOS计算机名称的文件；

NBNS：NetBIOS命名服务器负责将NetBIOS名称映射到IP地址上。该服务由Linux环境下的后台程序（nmbddaemon）执行。

NetBIOS服务与TCP/IP所提供端口对应关系如下：

表16-1NetBIOS服务端口号

|  |  |
| --- | --- |
| TCP/UDP端口 | NetBIOS服务 |
| UDP137端口 | NetBIOS名称服务 |
| UDP138端口 | NetBIOS数据报服务 |
| TCP139端口 | NetBIOS会话服务 |

二.NetBIOS/NetBEUI报文结构

NetBIOS/NetBEUI报文结构如下图所示：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 长度  （2字节） | 分隔符  （2字节） | 命令  （1字节） | 选项数据1  （1字节） | 选项数据2  （2字节） | Xmi  t（2字节） | RespCor  （2字节） |
| 目的名字（16字节） | | | | | | |
| 源名字（16字节） | | | | | | |

图16-1NetBIOS/NetBEUI报文结构

●长度：NetBIOS长度。

●分隔符：表示为实现NetBIOS功能定义并发数据。

●命令：特定协议命令，表示帧功能类型。

●选项数据1：每个特定命令分配1字节可选数据，0x00为保留。

●选项数据2：每个特定命令分配2字节可选数据，0x0000为保留。

●Xmit/RespCor（Xmit/responsecorrelatorresponsecorrelator）：用于联系接收到的对应传输请求的响应。

●目的名字：在非对话帧中该字段包含16字符名称。

●源名字：在非对话帧中该字段包含16字符源名，在对话帧中，该字段包含1字节源对话号码。

三.WINS简介

WINS（Windows网际名字服务）为NetBIOS名字提供名字注册、更新、释放和转换服务，这些服务允许WINS服务器维护一个将NetBIOS名链接到IP地址的动态数据库，大大减轻了对网络的负担。

在默认状态中，网络上的每一台计算机的NetBIOS名字是通过广播的方式来提供更新的，假如网络上有n台计算机，那么每一台计算机就要广播n-1次，对于小型网络来说，这似乎并不影响网络交通，但是对大型网络来说，加重了网络的负担。因此WINS对大中型企业来说尤其重要。

四.WINS名字注册、更新、释放及解析

#### 1.名字注册

名字注册就是客户端从WINS服务器获得信息的过程，在WINS服务中，名字注册是动态的。当一个客户端启动时，它向所配置的WINS服务器发送一个名字注册信息（包括了客户端的IP地址和计算机名），如果WINS服务器正在运行，并且没有其它客户计算机注册了相同的名字，服务器就向客户端计算机返还一个成功注册的消息（包括了名字注册的存活期——TTL）。与IP地址一样，每个计算机都要求有惟一的计算机名，否则就无法通信。如果名字已经被其它计算机注册了，WINS服务先验证该名字是否正在使用。如果该名字正在被使用则注册失败（发回一个负确认的信息），否则就可以继续注册。

#### 2.名字更新

因为客户端被分配了一个TTL（存活期），所以它的注册也有一定的期限，过了这个期限，WINS服务器将从数据库中删除这个名字的注册信息。它的过程是这样的：

●在过了存活期的1/8后，客户端开始不断试图更新它的名字注册，如果收不到任何响应，WINS客户端就会每过2分钟重复更新一次，直到存活期过了一半。

●当存活期过了一半时，WINS客户端将尝试与次选WINS服务器更新它的租约，它的过程与首选WINS服务器一样。

●如果时间过了一半后仍然没有成功的话，该客户端又回到它的首选WINS服务器了。

在该过程中，不管是与首选还是次选WINS服务器，一旦名字注册成功之后，该WINS客户端的名字注册将被提供一个新的TTL值。

#### 3.名字释放

在客户端的正常关机过程中，WINS客户端向WINS服务器发送一个名字释放的请求，以请求释放其映射在WINS服务器数据库中的IP地址和NetBIOS名字。收到释放请求后，WINS服务器验证一下在它的数据库中是否存在该IP地址和NetBIOS名，如果有就可以正常释放了，否则就会出现错误（WINS服务器向WINS客户端发送一个差错报文）。如果计算机没有正常关闭，WINS服务器将收不到释放的请求，则该名字直到WINS名字注册记录过期后才会失效。

#### 4.名字解析

当客户端在多个网络操作中需要WINS服务器解析名字时，例如当使用网络上其它计算机的共享文件时，为了得到共享文件，用户需要指定两件事：系统名和共享名，系统名需要转换成IP地址。名字解析过程是这样的：

●当客户端计算机想要转换一个名字时，它首先检查本地NetBIOS名字缓存器。

●如果名字不在本地NetBIOS名字缓存器中，便发送一个名字查询到首选WINS服务器（每隔15秒发送一次，共发三次），如果请求失败，则向次选WINS发送同样的请求。

●如果都失败了，那么名字解析可以通过其它途径来转换（例如本地广播、lmhosts文件和hosts文件、或者DNS）来进行名字解析。

五.SMB/CIFS简介

SMB（服务信息块协议）是由IBM、Microsoft和其它几家公司一起开发的公用标准，主要用于计算机间共享文件系统、打印机和其它资源。随着Internet的流行，Microsoft希望将这个协议扩展到Internet上去，成为Internet上计算机之间相互共享数据的一种标准。因此它将原有的几乎没有多少技术文档的SMB协议进行整理，重新命名为CIFS（通用互联网文件系统），并打算将它与NetBIOS相脱离，试图使它成为Internet上的一个标准协议。

实际上，SMB是一个通过网络在共享文件、设备、命名管道和邮槽之间操作数据的协议。CIFS是SMB的一个公共版本。

与其它标准的TCP/IP协议不同，SMB协议是一种复杂的协议，因为随着Windows计算机的开发，越来越多的功能被加入到协议中去了，很难区分哪些概念和功能应该属于Windows操作系统本身，哪些概念应该属于SMB协议。其它网络协议由于是先有协议，然后实现相关的软件，因此结构上就清晰简洁一些，而SMB协议一直是与Microsoft的操作系统混在一起进行开发的，因此协议中就包含了大量的Windows系统中的概念。

六.SMB协议数据交互过程

假设有这样一个环境：一个客户端希望访问一台服务器上的特定资源。在这个环境中，SMB协议数据交互的过程如下：

●开始于客户端向服务器请求一个NetBIOS会话。客户端发送它的已编码的NetBIOS名字到SMB服务器（服务器在139端口监听连接请求）。服务器接收到NetBIOS名字后回复一个NetBIOS会话数据报给有效的会话连接。客户端在建立了连接之后才能进入访问。

●客户端发送一个SMBnegprot（磋商协议）请求数据报。客户端列出了它所支持的所有SMB协议版本。

●通过磋商之后，客户端进程向服务器发起一个用户或共享的认证。这个过程是通过发送SesssetupX（会话建立）请求数据报实现的。客户端发送一对登录名/密码或一个简单密码到服务器，然后服务器通过发送一个SesssetupX应答数据报来允许或拒绝本次连接。

●在客户端完成了磋商和认证之后，它会发送一个TconX数据报并列出它想访问的特定网络资源的名称，之后服务器会发送一个TconX应答数据报以表示此次连接是否接受或拒绝。

七.SMB的安全等级

SMB的安全模式有两个等级：

第一种是“共享”级的安全模式。这种安全模式需要一个访问网络上共享资源的密码。用户通过这个正确的密码来访问网络资源（IPC，磁盘，打印机）。

第二种是“用户”级的安全模式。这是在第一种模式上的增强版。它坚持使用一对登录名/密码来访问共享资源。所以如果一个用户想访问这种类型的共享资源，就必须同时提供登录名和密码。这种模式对了解用户如何使用共享资源是很有帮助的。

【实验步骤】

练习1分析WINS的名字解析

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。

1.禁用IPv6协议。点击“开始→控制面板→查看网络状态和任务→更改适配器设置”，然后在当前使用的网络连接上点击鼠标右键，选择“属性”。在属性框中取消勾选“Internet 协议版本6（TCP/IPv6）”选项，然后点击确定。

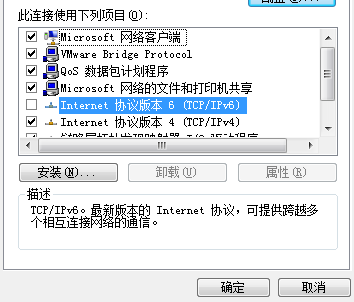


图16-2禁用IPv6协议

2.获取主机B的NetBIOS名字（主机B在命令行方式下，输入“nbtstat -n”）。

3.主机B启动协议分析器进行数据捕获，并设置过滤条件（提取主机A的IP<—>Any和NetBIOS\_Name协议）。

注：NetBIOS\_Name在UDP协议下。

4.在主机A上使用命令“nbtstat–R”，清除NetBIOS名字缓存。

5.在主机A上使用命令“ping主机B的NetBIOS名字”。

6.察看主机B捕获的数据填写下表。

表16-2实验结果

|  |  |  |
| --- | --- | --- |
| 字段名称 | 请求报文 | 应答报文 |
| 源端口、目的端口字段值 |  |  |
| 事务ID字段值 |  |  |
| 名字字段值 |  |  |
| 数据字段值(若有) |  |  |

●在请求报文中，目的IP地址字段的值是什么？它的作用是什么？

●结合上表填写数据，理解WINS名字解析的过程并绘制交互图。

思考问题：

1.WINS协议的主要功能是什么？

练习2分析SMB协议数据交互过程

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。

1.主机B新建一个用户。

(1) 主机B鼠标右键“计算机”，选择“管理”，打开计算机管理对话框（或通过“控制面板->管理工具->计算机管理”进入）。

(2)在左侧的树形结构中，右键点击系统工具->本地用户和组->用户，选择新用户。

(3)用户名设置为：SMBUser，密码设置为：1234，取消“用户下次登录时必须更改密码”选项。

2.主机B新建一个文件夹，在该文件夹中放置几个文件，然后右键点击该文件夹，选择属性打开文件夹属性对话框。进入“共享”页签，点击“共享”，在“文件共享”对话框中，选择并添加SMBUser用户，如下图所示：

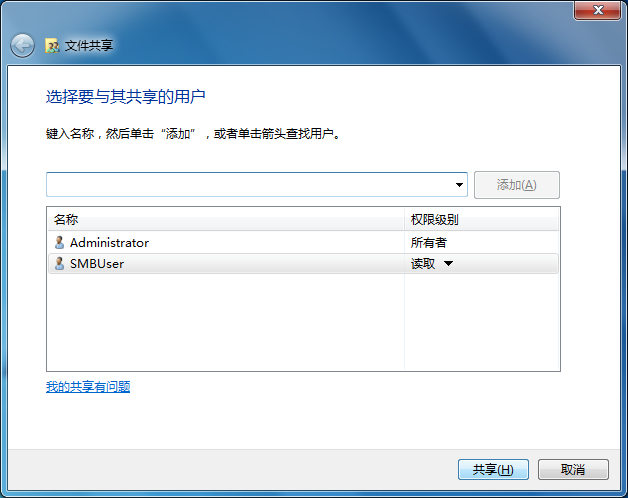


图16-3共享属性设置

点击“共享”，直至完成。

3.主机B启动协议分析器开始捕获数据，并设置过滤条件（提取NetBIOS\_SS、NetBIOS\_DGM协议）。

4.主机A访问主机B（用户名：SMBUser，密码：1234）：

方法一：在IE浏览器的地址栏中输入：\\主机B的IP地址。

方法二：在“开始”|“搜索程序和文件”栏中输入 ：\\主机B的IP地址。

「注」：若出现用户账户限制的提示，在主机B开始|搜索程序和文件栏输入gpedit.msc，打开组策略编辑器，在其左侧树形结构中依次单击Windows设置/安全设置/本地策略/安全选项，双击“账户：使用空白密码的本地账户允许进行控制台登录”打开对话框中选择本地安全设置，单击“已禁用”单选按钮。

5.找到该主机B后，打开其共享文件夹，从中选择一个文件并拷贝下来，粘贴到主机A，关闭访问窗口。

6.察看主机B捕获的数据。

●SMB会话分析中，主机A与B都使用了哪些端口？说明每种端口在会话中的作用。

●找出不同类型的SMB命令报文，理解它们的作用。

●结合上面的分析结果，绘制出大体的SMB会话交互图。

思考问题：

1.SMB协议的主要功能是什么？

练习3搜索网络上的一台计算机

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。

1.主机B启动协议分析器进行数据捕获，并设置过滤条件（提取本机IP<->Any）。

2.在主机A上使用命令“nbtstat–R”，清除NetBIOS名字缓存。

3.主机A单击桌面“计算机”，在左边列表中选择“网络”，在“搜索 网络”栏中，输入主机B的NetBIOS名字，开始在网络上搜索该计算机。

4.待搜索完成后，察看主机B捕获的数据。

5.分析该过程中涉及的协议。分析计算机“搜索”功能的实现。

思考问题：

1.通过实验说明windows网络协议的架构情况。

## 实验17路由信息协议(RIP)

【实验目的】

1.掌握路由协议的分类，理解静态路由和动态路由

2.掌握动态路由协议RIP的报文格式、工作原理及工作过程

3.掌握RIP计时器的作用

4.理解RIP的稳定性

【学时分配】

4学时

【实验环境】

练习1采用网络结构三（IPv4）

练习2 3 4采用网络结构四（IPv4）

【实验原理】

### 一.静态路由与动态路由

巨大的互联网是由许多小网络组成的，这些小网络使用路由器连接起来。在从源点到终点的通信过程中，数据包可能经过多个路由器，直到到达连接目的网络路由器为止。

路由器从一个网络接收数据包，并把数据包转发到另一个网络。一个路由器通常和多个网络相连。当路由器收到数据包时，它应当将数据包转发到哪一个网络取决于路由表的信息。

路由表可以是静态的也可以是动态的，静态路由拥有静态的路由表，动态路由拥有动态的路由表。静态路由表的路由信息是管理员设置的，并由管理员手动进行更新。动态路由表的路由信息是随着互联网的变化而自动更新的。现在只要互联网中有一些变化，路由器就应该尽快的更新路由表，所以现在互联网中的路由器大多使用动态路由表。例如，某条链路不能正常工作了，路由器就应该找到另一条路由，并把路由表进行更新。

由于互联网需要动态路由表的支持，因此就产生了多种路由选择协议。路由选择协议是一些规则和过程的组合。规则使得路由器之间能够共享它们所知道的互联网情况和邻站信息，而过程用来合并从其它路由器收到的信息。

### 二.内部和外部路由选择

今天的互联网非常庞大，仅仅使用一种路由选择协议是无法处理所有路由器的路由表更新任务的。为此，互联网划分为多个自治系统（AS）。自治系统是在单一的管理机构管辖下的一组网络和路由器。在自治系统内部的路由选择叫做域内路由选择。在自治系统之间的路由选择叫做域间路由选择。每一个自治系统使用一种域内路由选择协议（例如RIP或OSPF）处理本自治系统内部的路由选择。而对于自治系统之间的路由选择一般只能使用“域间路由选择协议（BGP）”来进行路由选择。如下图所示：

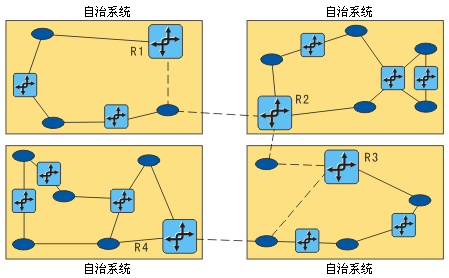


图17-1内部和外部路由选择

### 三.距离向量路由

距离向量路由选择协议得到的路由是任何两个节点之间代价最小的路由。在协议中，每一个节点维护一个到其它节点的最小距离向量表。在这个表中还指出路径的下一跳地址，以便把数据包发送到目的节点。

下图给出了一个具有5个节点的网络，以及这些节点维护的最小距离向量表。

节点A的路由表指出怎样能够从这个节点到达其它节点。例如，到节点E的最小代价是6，这条路由经过节点C。

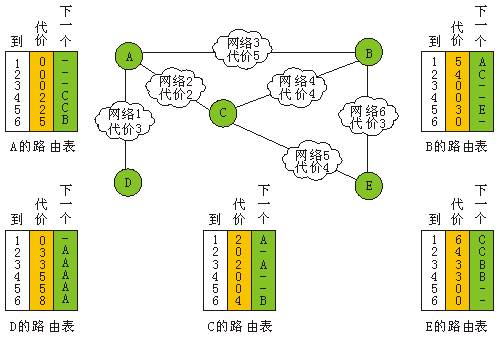


图17-2距离向量路由表

#### 1.初始化

在图17-2中的表是稳定的，每一个节点都知道如何到达其它节点，以及到达其它节点的代价。但是最初的情况并不是这样的，在初始状态下，每一个节点只知道到与它直接相连的节点的代价（假定每一个节点能够向其相邻节点发送一个报文，并且找出从它到这些邻节点的代价）。下图给出了每一个节点的最初的表。对于到不相邻节点的代价则标记为无穷大（不可达）。

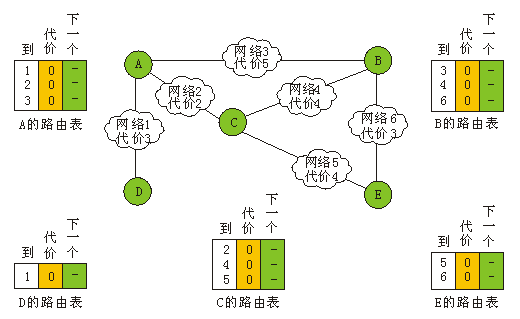


图17-3在距离向量路由选择中的表的初始化

#### 2.共享

距离向量路由选择总体的思想就是在相邻节点之间共享路由信息。节点A不知道有关节点E的路由信息，但是节点C知道。如果节点C把它的路由信息与节点A共享，那么节点A也就会知道怎样到达节点E。另一方面，节点C不知道怎样到达节点D，但是节点A知道。如果节点A把它的路由信息与节点C共享，那么节点C也就会知道怎样到达节点D。这就是路由信息的共享，作为相邻节点的节点A和节点C，如果它们彼此互助，就可以改进其路由表。

#### 3.更新

当一个节点从邻站收到路由表信息时，它就需要更新其路由表。更新的方法如下图所示：

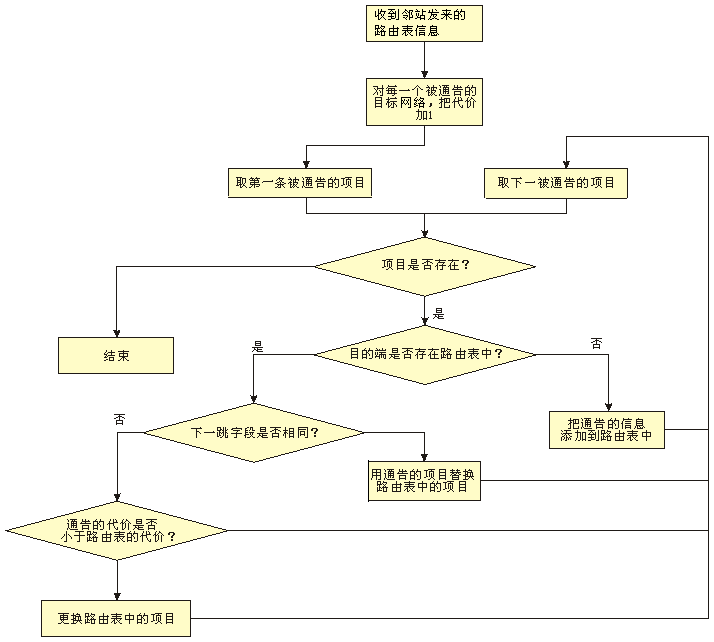


图17-4路由表更新

在这里，假设收到的路由信息包含两列内容：目的节点和到达目的节点的代价。如下图所示：

|  |  |
| --- | --- |
| 目的节点 | 代价 |
| A | 2 |
| B | 4 |
| C | 0 |
| D | ∞ |
| E | 4 |

图17-5只包含两列的路由表

下图表示节点A在收到从节点C传来的部分路由表后怎样更新它的路由表。

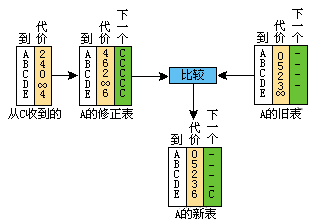


图17-6在距离向量路由选择中的更新

这里应该注意几点。第一，把一个数与无穷大相加，结果还是无穷大。第二，修正的表指出怎样从节点A经过节点C到达节点A。如果节点A需要经过节点C到达它自己，它必须先到节点C再返回，因此代价是4。第三，节点A经过路由表更新后，路由表的最后一项改变了。以前，节点A不知道如何到达节点E（代价是无穷大）；现在它知道到达节点E的代价是6，要经过节点C。

每一个节点都从其相邻节点接收路由表信息，并用这个信息来更新自己的路由表。在短时间内，如果网络本身没有变化（如链路出了故障），所有节点的路由表就会达到稳定状态，表中的内容保持不变。

#### 4.什么时候共享

一个节点周期性地向其所有相邻节点发送它的路由表信息。这个周期通常为30秒，具体的时间取决于使用的距离向量路由选择协议。

### 四.RIP协议简介

RIP（路由信息协议）是应用较早、使用较普遍的内部网关协议，适用于小型同类网络，是典型的距离向量路由协议。

RIP通过广播UDP协议520端口封装成的报文来交换路由信息，默认每30秒发送一次路由信息更新报文。RIP提供跳跃计数（hopcount）作为尺度来衡量路由距离，跳跃计数是一个数据报到达目标设备所必须经过的路由器数目。RIP最多支持的跳数为15，即在源和目的网络之间所要经过的最多路由器的数目为15，跳数16表示不可达。

RIP协议的特点：

●仅和相邻路由器交换信息。

●交换的信息是当前本路由器所知道的全部信息，即自己的路由表。

●按固定的时间间隔交换路由信息，例如：每隔30秒交换路由信息。

### 五.RIP报文格式

RIP报文的格式如下图所示：

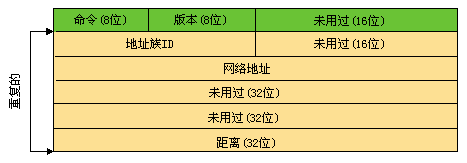


图17-7RIP报文的格式

●命令：这个8位字段指明报文的类型,1表示请求报文，2表示响应报文。

●版本：这个8位字段定义版本，1表示RIPv1，2表示RIPv2。

●地址族ID：这个16位字段定义所使用的协议系列。值为2时表示TCP/IP协议族。

●网络地址：这个字段定义目的网络的地址。RIP给这个字段已分配了14字节，可用于任何协议。但是，现在IP只使用4字节。地址的其余部分应填入0。

●距离：这个32位字段定义从发送报文的路由器到目的网络的跳数。

### 六.RIP运行过程

初始化：在启动一个路由守护程序时，它先判断启动了哪些接口，并在每个接口上发送一个请求报文，要求其它路由器发送完整路由表。在点对点链路中，该请求是发送给其它终点的。如果网络支持广播的话，这种请求是以广播形式发送的。目的UDP端口号是520。这种请求报文的命令字段为1，但网络地址字段设置为0，而度量字段设置为16。这是一种要求另一端完整路由表的特殊请求报文。

接收到请求：如果这个请求是刚才提到的特殊请求，那么路由器就将完整的路由表发送给请求者。否则，就处理请求中的每一个表项。如果有连接到指明地址的路由，则将距离设置成值，否则将距离置为16（一种称为“无穷大”的特殊值），然后发回响应。

接收到响应：为使响应生效，可能会更新路由表。更新路由表包括增加新表项、对已有的表项进行修改以及将已有表项删除。

定期选路更新：每过30秒，所有或部分路由器会将其完整路由表发送给相邻路由器。发送路由表可以是广播形式的（如在以太网上），或是发送给点对点链路的其它终点的。

在RIP运行过程中，有三个非常重要的计时器：定期计时器、截止期计时器和无用信息收集计时器。

定期计时器：定期计时器用于更新报文的定期通告。虽然协议指明了这个计时器应该设置为30秒，但是在实际应用中，这个计时器的值是25秒到35秒之间的一个随机数。这是为了防止在互联网上由于路由器都同时更新而引起的过载。

每一个路由器有定期计时器，设置为25秒到35秒之间的一个随机数。它向下计数，当到达零时就发送更新报文，然后把计时器再随机的进行设置一次。

定期计时器不受其它计时器的影响，当定期计时器到期时，更新报文就会发送出去，而不管从其它计时系统发出的其它更新报文。

截止期计时器：截止期计时器用于路由的有效性。当路由器收到路由的更新信息时，截止期计时器就对这个路由设置到180秒。每当收到这个路由新的更新信息时，截止期计时器就要复位。在正常情况下，每隔30秒发生一次复位。但是，若在互联网中出现了问题，并且在180秒内没有收到更新报文，那么就认为这个路由过期了，路由的跳数将被设置为16，这表示目的端不可达。每一条路由有它自己的截止期计时器。

无用信息收集计时器：当一条路由变为无效时，路由器将这条路由的跳数设置为16（而不是立即在路由表中清除这条路由），并启动无用信息收集计时器，设置计时器为120秒。当计时器到期时，路由器就从路由表中清除掉。这个计时器使得邻站知道这条路由是无效的。

### 七.RIP的缺点及改进

#### 1.RIP协议的缺点

(1)缓慢收敛

缓慢收敛是RIP的缺点之一，它是指在互联网上某处发生的变化要传播到互联网的其它部分是很慢的。例如，在下图中，网络1有了变化，路由器R1立即更新了自己的路由表。但是，由于每一个路由器每隔30秒才发送其定期更新，因此这表示这个变化要到达R2平均需要经过15秒（0到30秒之间）。R3要收到这个变化又要经过另一个平均15秒，等等。当这个信息最后到达路由器Rn时，已经经过了15×n秒。若n是20，则要经过300秒。在300秒内，ATM网络能够发送超过10亿位。若这样的变化影响这些位，则10亿位就丢失了。

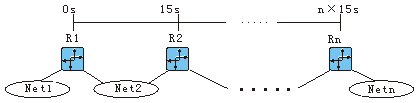


图17-8缓慢收敛

解决RIP缓慢收敛的方法是限制跳数为15。这样可防止数据包无休止地在网络中兜圈子而阻塞了互联网。因此，数值16被认为是无穷大，并表示不可达的网络。如下图所示：

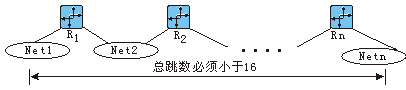


图17-9跳数

(2)不稳定性

RIP的另一个缺点是不稳定性，不稳定性表示运行RIP的互联网中数据包可能在一个回路中从一个路由器到另一个路由器兜圈子。把跳数限制为15能够改进稳定性，但不能解决所有的问题。

下图是一个不稳定性的例子，路由器A在它的路由表中对Net1写上的代价是0。路由器B只能通过路由器A才能接入到Net1，因此它的代价是1。现在假定Net1的连接断了，无法接入到Net1，这时路由器A立即响应，并把到Net1这列的代价改变为16（无穷大）。但是，它要等到30秒以后才能把这个新信息放在它的更新报文中发送出。在这时，路由器B可能把更新报文发送给A。路由器A现在有两列到Net1的项目：从它自己的路由表中得到的是代价为16，而从路由器B得到的代价是1。于是A被愚弄了，它想会有另一个路径可经过B再到Net1。路由器A就把他到Net1这列的代价改变为2（1＋1），并把这个更新发送给B。路由器B到Net1的代价现在是2（从A）和1（从它自己的路由表）。路由器B知道只有通过路由器A才能到达Net1，因此，它不考虑它自己的较低的代价而是把它代价改变为3（2＋1）。这样来回的更新不断继续下去，直到两个路由器都到达代价为16。到这个时候，这两个路由器才知道没有法子接入到网络Net1。

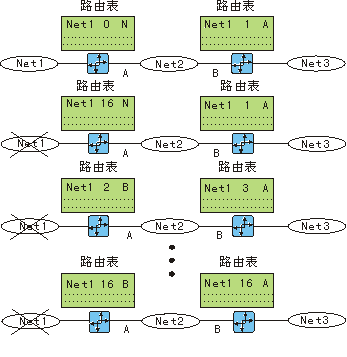


图17-10不稳定性

#### 2.RIP协议的改进

(1)触发更新

触发更新可提高稳定性。若网络中没有变化，路由器按30秒的间隔发送更新信息。但若网络有变化，路由器就立即发送它的更新信息。这个过程叫做触发更新。

每一个路由器在收到有变化的更新信息时就立即发出新的信息，这比平均的15秒快了很多。虽然触发更新可大大地改进路由选择，但它不能解决所有的路由选择问题。例如，用这种方法不能处理路由器出故障的问题。

(2)水平分割

水平分割也可以提高稳定性，在发送路由选择报文时增加了选择性，路由器必须区分不同的接口。如果路由器从某个接口已经收到了路由更新信息，那么这个同样的更新信息就不能再通过这个接口回送过去。如果某个接口通过了给某个路由器更新的信息，那么这个更新信息就不能再发送回去，这是已经知道了的信息，因而是不需要的。

(3)毒性反转

路由中毒是指路由信息在路由表中失效时，先将度量值变为无穷大，而不是马上从路由表中删掉这条路由信息。

毒性反转与路由中毒概念是不一样的，它是指收到路由中毒消息的路由器，不遵守水平分割原则，而是将中毒消息转发给所有的相邻路由器，也包括发送中毒信息的源路由器，也就是通告相邻路由器这条路由信息已失效了。毒性反转的主要目的是加快收敛。

### 八.RIP的限制

虽然RIP有很长的历史，但它还是有自身的限制。它非常适合于为早期的网络互联计算路由。然而，现代技术进步已极大地改变了互联网络建造和使用的方式。因此，RIP越来越不适应今天互联网的需求。

RIP的一些限制是：

●不能支持大于15跳的路径：RIP设计用于相对较小的自治系统。这样一来，它强制规定了一个严格的跳数限制为15跳。当报文由路由设备转发时，它们的跳数计数器会加上其要被转发的链路的耗费。如果跳数计值到15之后，报文仍没到达它寻址的目的地，那个目的地就被认为是不可达的，并且报文被丢弃。

●依赖于固定的度量来计算路由：对跳数的讨论为考察RIP的下一个基本限制作了很好的铺垫，这个限制就是固定耗费度量。虽然耗费度量能由管理员配置，但它们本质上是静态的。RIP不能实时地更新它们以适应网络中遇到的变化。由管理员定义的耗费度量保持不变，直到手动更新。这意味着RIP尤其不适合于高度动态的网络，在这种环境中，路由必须实时计算以反映网络条件的变化。

●路由更新耗费的资源过多：RIP节点会每隔30秒钟广播其路由表。在具有许多节点的大型网络中，这会消耗掉相当数量的带宽。

●相对缓慢的收敛：从人的角度来看，等待30秒进行一次更新不会感到不方便。然而，路由器和计算机以比人快得多的速度运行。不得不等上30秒进行一次更新会有很明显的不利结果。比仅仅等上30秒进行一次更新更具破坏性的是不得不等上180秒来作废一条路由。而这只是一台路由器开始进行收敛所需的时间。依赖于互联的路由器个数及它们的拓扑结构，可能需要重复更新才能完全收敛于新拓扑。RIP路由器收敛速度慢会创造许多机会使得无效路由仍被错误地作为有效路由进行广播。显然，这样会降低网络性能。

●缺乏动态负载均衡支持：由于RIP本身的特点，它缺乏对动态负载均衡的支持，如下图所示：

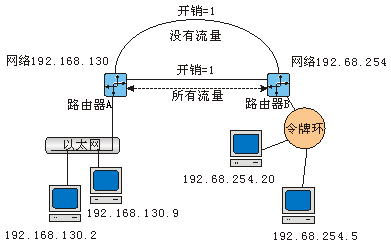


图17-11RIP缺乏动态负载均衡支持

【实验步骤】

练习1静态路由与路由表

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.主机A、B、C、D、E、F在命令行下运行“routeprint”命令，察看路由表，并回答以下问题：

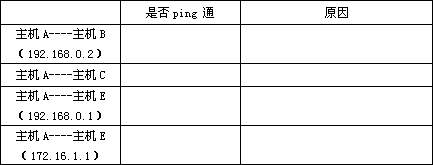
●路由表由哪几项组成？

2.从主机A依次ping主机B（192.168.0.2）、主机C、主机E（192.168.0.1）、主机E（172.16.1.1），观察现象，记录结果。通过在命令行下运行routeprint命令，察看主机B和主机E路由表，结合路由信息回答问题：

●主机A的默认网关在本次练习中起到什么作用？

●记录并分析实验结果，简述为什么会产生这样的结果？

表17-1实验结果



3.主机B和主机E启动静态路由。

(1)主机B与主机E在命令行下使用“staticroute\_config”命令来启动静态路由。

(2)在主机B上，通过在命令行下运行routeadd命令手工添加静态路由（“routeadd172.16.1.0mask255.255.255.0192.168.0.1metric2”）。

(3)在主机E上，也添加一条静态路由（“routeadd172.16.0.0mask255.255.255.0192.168.0.2metric2”）。

(4)从主机A依次ping主机B（192.168.0.2）、主机E（192.168.0.1）、主机E（172.16.1.1），观察现象，记录结果。

(5)通过在命令行下运行routeprint命令，察看主机B和主机E路由表，结合路由信息回答问题：

●记录并分析实验结果，简述手工添加静态路由在此次通信中所起的作用。

表17-2实验结果

|  |  |  |
| --- | --- | --- |
|  | 是否ping通 | 原因 |
| 主机A----主机B  （192.168.0.2） |  |  |
| 主机A----主机E  （192.168.0.1） |  |  |
| 主机A----主机E  （172.16.1.1） |  |  |

4.在主机B上，通过在命令行下运行routedelete命令（“routedelete172.16.1.0”）；在主机E上，运行routedelete命令（“routedelete172.16.0.0”）删除手工添加的静态路由条目。

●简述静态路由的特点以及路由表在路由期间所起到的作用。

练习2领略动态路由协议RIPv2

本练习将主机A、B、C、D、E、F作为一组进行实验。

注：如果想使用地址本扫描到各个虚拟路由器，那么首先要进入到虚拟路由系统中，执行“zebra -d”和“addrscan”命令。然后使用地址本扫描就可以扫描到虚拟路由器了。

1.在主机A、B、C、D、E、F上启动协议分析器，设置过滤条件（提取RIP和IGMP），开始捕获数据。

2.主机B和主机E启动RIP协议:

(1)主机B和主机E分别开启Ipv4虚拟路由器。

(2)主机B和主机E在Ipv4虚拟路由器内使用“zebra -d”和“ripd -d”命令开启RIP协议。

3.所有主机人员通过协议分析器观察报文交互，直到两台主机的路由表达到稳定态。

●如何判定路由表达到稳定态？

●在主机B、E察看路由表，记录稳定状态下主机B和主机E的路由表条目。

（1）使用“telnet localhost 2602”命令登陆虚拟路由器终端，密码123456。

（2）使用“enable”命令进入用户视图。

（3）使用“show ip rip”命令查看路由表。

4.观察协议分析器报文交互，并回答问题：

●IGMP报文在RIP交互中所起的作用是什么？

●通过以上4步，绘制主机B和主机E的RIP交互图（包括IGMP报文）。

练习3RIP的计时器

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.在主机A、B、C、D、E、F上重新启动协议分析器，设置过滤条件（提取RIP），开始捕获数据。

注：如果想使用地址本扫描到各个虚拟路由器，那么首先要进入到虚拟路由系统中，执行“zebra -d”和“addrscan”命令。然后使用地址本扫描就可以扫描到虚拟路由器了。

2.主机B和主机E重启RIP协议并查看路由表是否达到稳定状态（同练习二的步骤2、步骤3），使用“configure terminal”命令进入配置视图，同时设置“周期公告间隔”为20秒。

（1）在进入用户视图的基础上，使用“configure terminal”命令进入配置视图。

（2）使用“router rip”命令进入rip协议配置视图。

（3）使用“timer basic 20 180 120”命令设置“周期公告间隔”为20秒。

（4）所有主机人员用协议分析器察看报文序列，并回答问题：

●将“周期公告间隔”设置为0秒可以吗？为什么操作系统对“周期公告间隔”有时间上限和时间下限？上限和下限的作用是什么？

●通过协议分析器，比较两个相邻通告报文之间的时间差，是20秒吗？如果不全是，为什么？

3.将“路由过期前的时间”设置为30秒。

(1) 主机B 在rip协议配置下，输入“timer basic 20 30 120”。

(2) 主机E 在rip协议配置下，输入“timer basic 20 30 120”。

(3)禁用主机E的192.168.0.1的网络连接。在30秒内观察主机B的路由条目变化，并回答问题：

●简述“路由过期计时器”的作用是什么？

4.恢复主机e2的的网络连接。

练习4RIP的稳定性

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.在主机A、B、C、D、E、F上重新启动协议分析器捕获数据，并设置过滤条件（提取RIP）。

注：如果想使用地址本扫描到各个虚拟路由器，那么首先要进入到虚拟路由系统中，执行“zebra -d”和“addrscan”命令。然后使用地址本扫描就可以扫描到虚拟路由器了。

2.主机B和主机E重启RIP协议（同练习二的步骤2），同时去掉“启用水平分割处理”和“启用毒性反转”选项。

（1）在进入用户视图的基础上，使用“configure terminal”命令进入配置视图。

（2）主机B和主机E输入“interface eth0”，“ip rip split-horizon”。

（3）主机B和主机E输入“interface eth1”，“ip rip split-horizon”。

（4）等待一段时间，直到主机B和主机E的路由表达到稳定态。

3.主机B和主机E查看路由表（同练习二的步骤3），结合协议分析器上捕获的RIP报文内容，回答问题：

●记录此时主机B和主机E的路由表条目。

●同练习二中记录的路由表条目作比较，简述发生变化的原因。

4.查看未启用毒性反转的效果。

(1)拔掉主机E与主机F相连的网线

(2)主机A，主机C查看协议分析器捕获的数据

●主机A收到度量为16的RIP报文了吗？

●主机C收到度量为16的RIP报文了吗？

思考问题：

1.RIP使用UDP，这样做有何优点？

2.跳数限制如何缓解RIP的问题？

3.试列举RIP的缺点及其相应的补救办法。

## 实验18开放式最短路径优先协议(OSPF)Ⅰ

【实验目的】

1.掌握OSPF的报文格式

2.掌握OSPF的工作过程

3.了解常见的LSA的结构及LSDB的结构

【学时分配】

4学时

【实验环境】

该实验采用网络结构四（IPv4）

【实验原理】

### 一.链路状态路由

链路状态路由选择的原理与距离向量路由选择的原理不同。在链路状态路由选择中，节点需要知道区域的整个拓扑——节点和链路列表，以及它们是怎样连接起来的，包括类型、代价（度量）和链路的状态（正常工作或故障），然后这个节点使用Dijkstra算法构造出路由表。

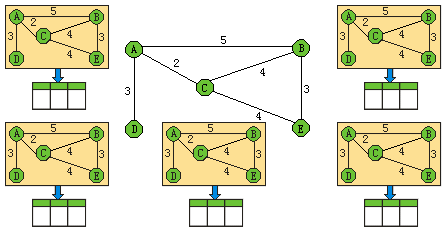


图18-1链路状态路由选择的概念

上图给出了具有5个节点的区域。每一个节点使用同样的拓扑创建路由表，但是不同节点的路由表却是不同的，这是因为路由表是基于对拓扑的不同解释计算的。这和不同的人看同一个城市的地图相似，不同的人可以有同样的地图，但是每一个人需要使用不同的路径到达他的目的地。

拓扑是动态的，表示每一个节点和每一条链路的最新情况。如果在网络中的某处出现了变化（例如，一条链路出故障），则拓扑必须相应的改变，每一个节点的路由表都需要进行更新。

一个公共的拓扑怎样能够动态的存储在每一个节点中呢？在初始时，没有节点知道区域的整体拓扑。虽然没有这个区域的整体拓扑，但是每一个节点有部分的拓扑信息，它知道它的链路的状态（类型、条件及代价）。整个区域的拓扑可以从每一个节点的部分拓扑信息组合而成。下图给出了与图18-1同样的区域，指出每一个节点的部分拓扑信息。

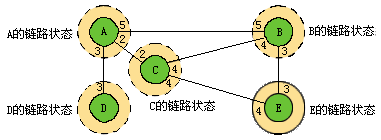


图18-2链路状态的部分拓扑

节点A知道它到节点B的代价为5，到节点C的代价为2，到节点D的代价为3。节点C知道它到节点A的代价为2，到节点B的代价为4，到节点E的代价为4。节点D知道它到节点A的代价为3，等等。虽然这些拓扑信息有些重叠，但是这能保证产生一个公共的关于区域完整的拓扑，并且给每一个节点提供这个拓扑信息。

#### 1.构造路由表

在链路状态路由选择中，每一个节点都根据区域的拓扑生成路由表，得出到其它节点的最小代价。每个节点使用4个步骤来完成这项工作：

第一步，每一个节点产生链路状态，这叫做LSP（链路状态数据包）。

第二步，向其它节点进行LSP的传播，这叫做洪泛。

第三步，每一个节点形成最短路径树。

第四步，基于最短路径树计算路由表。

#### 2.链路状态数据包的创建

LSP可以携带大量的信息。现在假定它携带最小数量的数据：节点标识、链路列表、一个序号以及生存期。节点标识和链路列表用来构造拓扑。序号有利于洪泛和把新的LSP与旧的LSP区分开。生存期用来防止旧的LSP长时间留在区域中。LSP是在以下两种情况下产生的：

(1)当这个区域的拓扑发生变化时。这时LSP的传播是快速通知区域中的所有节点更新拓扑的主要方法。

(2)定期更新。周期性的传播LSP。对于定期传播的周期通常设置为60分钟或120分钟。较长的周期可以确保洪泛不会在网络上产生很大的通信量。

#### 3.LSP的洪泛

每一个节点将LSP向所有其它节点传播，而不是仅仅向其邻站传播。这个过程叫做洪泛，它基于以下几点：

(1)产生LSP的节点把LSP的副本从它的每一个接口发送出去。

(2)收到LSP的节点，把收到的LSP和已经有的LSP副本进行比较。如果新到达的LSP比原有的LSP还旧（检查序号即可知道），就丢弃这个LSP。如果它比较新，这个节点就进行以下的操作：

●丢弃旧的LSP，保留新的。

●从所有的接口（除了这个数据包到达的那个接口）发送这个LSP的副本。这就保证洪泛会在这个区域的某处（这里的节点只有一个接口）停止。

### 二.OSPF协议简介

OSPF（开放式最短路径优先）是一种典型的链路状态路由协议。采用OSPF的路由器彼此交换并保存整个网络的链路信息，从而掌握全网的拓扑结构，独立计算路由。该路由协议基于链路状态算法，具有收敛速度快，平稳，杜绝环路等优点，十分适合大型的计算机网络使用。目前广泛使用的是OSPF第二版。

OSPF作为一种内部网关协议，用于在同一个自治系统中的路由器之间发布路由信息。在目前应用的路由协议中占有相当重要的地位。

### 三.区域与度量

#### 1.区域

为了使路由选择高效，OSPF把一个自治系统划分为若干区域。一个区域包含一些网络、主机和路由器。一个自治系统可划分为若干个不同的区域。在一个区域里的所有网络必须是互相连接的。

在一个区域内的路由器使用洪泛法传送路由选择信息。区域之间使用区域边界路由器连接，区域边界路由器把有关本区域的信息汇总起来发送到其它区域。在自治系统中有一个特殊区域叫做主干；在自治系统中的所有区域必须连接到主干上。换言之，主干相当于一级区域，而其它的区域相当于二级区域，二级区域必须与一级区域相连。但这并不表示在各区域内的路由器不能互相连接。在主干中的路由器叫做主干路由器。主干路由器也可以同时是区域边界路由器。

如果在主干和区域之间的连通性被破坏了，则管理员必须创建路由器间的虚链路，以使得作为一级区域的主干的各种功能的连续性。

每一个区域有区域标识。主干的区域标识是零。下图给出了自治系统及其划分的区域。

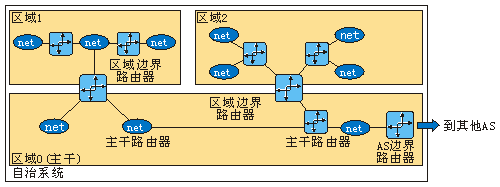


图18-3自治系统中的区域

#### 2.度量

OSPF协议允许管理员给每一条路由指派度量。度量是基于区分服务的。一个路由器可以有多个路由表，而每一个路由表基于不同的区分服务。

### 四.链路的类型

#### 1.链路的类型

在OSPF中已定义了4种类型的链路：点对点链路、转接链路、残桩链路和虚拟链路，如下图所示：



图18-4链路的类型

#### 2.点对点链路

点对点链路连接两个路由器，而中间没有任何其它的主机或路由器。这种链路的目的仅仅是为了连接这两个路由器。比如两个路由器用一条电话线或一条双绞线连接起来。不需要给这种类型的链路指派一个网络地址。用图表示时，路由器用节点表示，而链路用一条连接两个节点的双向边来表示。度量则表示在两端，各表示每一个方向的度量，通常它们是一样的。每一个路由器在链路的另一端只有一个相邻主机，如下图所示：

说明: 1317191979563_pic

图18-5点对点链路

#### 3.转接链路

转接链路是一种网络，在它上面连接有若干个路由器。数据可以从任何一个路由器进入网络，并从任何一个路由器离开网络。所有具有两个或更多的路由器的局域网和广域网都属于这种类型。每一个路由器有几个相邻主机。如下图（A）所示的以太网，路由器B的相邻主机是A、C、D和E。可以用下图（B）的表示方法表示这种相邻主机关系。

这种关系既低效又不现实。说它低效是因为每一个路由器需要通知相邻的其它4个路由器，总共有20个通知。说它不现实是因为在每一对路由器之间没有一个单独的网络，只有一个网络用作所有这5个路由器之间的交叉路口。

为了表示每一个路由器是通过单个网络连接到其它的路由器的，将这个网络用一个节点来表示。网络并不是一台计算机，它不能像路由器那样工作，所以在网络中的一个路由器就担负起这个责任。这个路由器有双重作用：即是一个真路由器，又是一个指定路由器。如下图（C）的拓扑表示转接网络的连接：

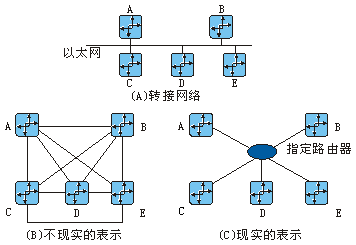


图18-6转接链路

现在每一个路由器只有一个相邻主机，即指定路由器（网络）。另一方面，这个指定路由器有5个相邻主机。向相邻主机的通知数从20减少到10。另外，链路用连接节点的双向边来表示。虽然从每一个节点到指定路由器都有一个度量，但从指定路由器到任何其它节点都没有度量。原因就是指定路由器代表网络，不能为它计算两次。当数据包进入网络时，就指派一个代价，当数据包离开网络到其它的路由器时就没有代价。

#### 4.残桩链路

残桩链路是指只连接到某个路由器的网络。数据包通过这个单一路由器进入网络，而离开网络也是通过这个路由器。这是转接网络的一个特例。这里将路由器表示为一个节点而用指定路由器表示这个网络。这条链路是单向的，即从路由器到网络，如下图所示：

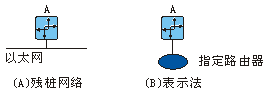


图18-7残桩链路

#### 5.虚拟链路

当两个路由器之间的链路断开时，管理员就在它们之间创建一条使用更长路径的虚拟链路，这可能要经过好几个路由器。

### 五.OSPF报文格式

OSPF报文格式如下图所示：

|  |  |  |
| --- | --- | --- |
| 版本（8位） | 类型（8位） | OSPF包长度（16位） |
| 路由器ID（32位） | | |
| 区域ID（32位） | | |
| 校验和（16位） | | 鉴别类型（16位） |
| 鉴别（64位） | | |
| 鉴别（64位） | | |
| OSPF报文（有五种类型） | | |

图18-8OSPF报文格式

在OSPF报文格式中，各个字段的意义如下：

●版本：定义所采用的OSPF路由协议的版本，通常为版本2。

●类型：定义OSPF数据包类型。OSPF数据包共有五种：

表18-1OSPF数据包类型

|  |  |
| --- | --- |
| Hello报文 | 用于建立和维护相邻的两个OSPF路由器的关系，该数据包是周期性地发送的。 |
| DatabaseDescription报文 | 用于描述整个数据库，该数据包仅在OSPF初始化时发送。 |
| LinkStateRequest报文 | 用于向相邻的OSPF路由器请求部分或全部的数据，这种数据包是在当路由器发现其数据已经过期时才发送的。 |
| LinkStateUpdate报文 | 这是对linkstaterequest数据包的响应，即通常所说的LSA数据包。 |
| LinkStateAcknowledgment报文 | 是对LSA数据包的响应。 |

●OSPF包长度：定义整个数据包的长度。

●路由器ID：用于描述数据包的源地址，以IP地址来表示。

●区域ID：用于区分OSPF数据包属于的区域号，所有的OSPF数据包都属于一个特定的OSPF区域。

●校验和：用于标记数据包在传递时有无误码，检查范围覆盖整个OSPF报文。

●鉴别类型：定义OSPF验证类型。

●鉴别：包含OSPF验证信息。

OSPF的报文共分为以下五种类型：

●问候（Hello）报文：发现及维持邻居关系，选举DR（指定路由器）、BDR（次级指定路由器）。报文格式如下图所示：

|  |
| --- |
| OSPF报文头（192位） |
| 网络掩码（32位） |
| 发送hello报文的间隔（16位） |
| 选项（8位） |
| 路由器优先级（8位） |
| 认为对方断开时间（32位） |
| 选举/备份选举路由器（192位） |
| 所有邻居路由器ID（n\*32位） |

图18-9问候报文

●数据库描述（DatabaseDescription）报文：描述本地LSDB的情况。报文格式如下图所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OSPF头标（192位） | | | | |
| 保留（24位） | 可选（5位） | I（1位） | M（1位） | MS（1位） |
| DD序号（n\*32位） | | | | |
| 数据段（1ton位） | | | | |

图18-10数据库描述报文

●链路状态请求（LinkStateRequest）报文：向对端请求本端没有或对端更新的LSA。报文格式如下图所示：

|  |
| --- |
| OSPF报文头（192位） |
| 链接状态类型1（32位） |
| 链接状态ID1（32位） |
| 发送报文的路由器1（32位） |
| …… |
| 链接状态类型n（32位） |
| 链接状态IDn（32位） |
| 发送报文的路由器n（32位） |

图18-11链路状态请求报文

●链路状态更新（LinkStateUpdate）报文：向对方更新LSA。报文格式如下图所示：

|  |  |  |  |
| --- | --- | --- | --- |
| OSPF报文头（192位） | | | |
| 链接状态报文头（160位） | | | |
| 全0（6位） | E（1位） | B（1位） | 全0（8位） |
| 路由器链接的数量（16位） | | | |
| 路由器链接状态报文 | | | |

图18-12链路状态更新报文

●链路状态确认（LinkStateAcknowledgment）报文：收到LSU报文后，进行确认。报文格式如下图所示：

|  |
| --- |
| OSPF报文头（192位） |
| 链接状态类型（32位） |
| 链接ID（32位） |
| 发送报文的路由器（32位） |
| 链接状态序号（32位） |
| 链接状态校验和（16位） |
| 发送状态年龄（16位） |

图18-13链路状态确认报文

### 六.OSPF运行过程

#### 1.建立路由器的邻接关系

所谓“邻接关系”是指OSPF路由器以交换路由信息为目的，在所选择的相邻路由器之间建立的一种关系。路由器首先发送拥有自身ID信息的Hello报文。与之相邻的路由器如果收到这个Hello报文，就将这个报文内的ID信息加入到自己的Hello报文内。如果路由器的某端口收到从其它路由器发送的含有自身ID信息的Hello报文，则它根据该端口所在网络类型确定是否可以建立邻接关系。

在点对点网络中，路由器将直接和对端路由器建立起邻接关系，并且该路由器将直接进入到第三步操作：发现其它路由器。若为MultiAccess网络，该路由器将进入选举步骤。

#### 2.选举DR/BDR

不同类型的网络选举DR（指定路由器）和BDR（次级指定路由器）的方式不同。MultiAccess网络支持多个路由器，在这种状况下，OSPF需要建立起作为链路状态和LSA更新的中心节点。选举利用Hello报文内的ID和优先权（Priority）字段值来确定。优先权字段的取值范围是0到255，优先权值最高的路由器成为DR。如果优先权值大小一样，则ID值最高的路由器选举为DR，优先权值次高的路由器选举为BDR。优先权值和ID值都可以直接设置。

#### 3.发现路由器

在这个步骤中，路由器与路由器之间首先利用Hello报文的ID信息确认主从关系，然后主从路由器相互交换部分链路状态信息。每个路由器对信息进行分析比较，如果收到的信息有新的内容，路由器将要求对方发送完整的链路状态信息。这个状态完成后，路由器之间建立完全相邻（FullAdjacency）关系，同时邻接路由器拥有自己独立的、完整的链路状态数据库。

#### 4.选择适当的路由器

当一个路由器拥有完整独立的链路状态数据库后，它将采用SPF算法计算并创建路由表。OSPF路由器依据链路状态数据库的内容，独立地用SPF算法计算出到每一个目的网络的路径，并将路径存入路由表中。

OSPF利用度量计算目的路径，度量最小者即为最短路径。在配置OSPF路由器时可根据实际情况，如链路带宽、时延或经济上的费用设置链路度量大小。度量越小，则该链路被选为路由的可能性越大。

#### 5.维护路由信息

当链路状态发生变化时，OSPF通过洪泛过程通告网络上其它路由器。OSPF路由器接收到包含有新信息的链路状态更新报文，将更新自己的链路状态数据库，然后用SPF算法重新计算路由表。在重新计算过程中，路由器继续使用旧路由表，直到SPF完成新的路由表计算。新的链路状态信息将发送给其它路由器。

### 七.OSPF链路状态公告类型

OSPF路由器之间交换链路状态公告（LSA）信息。OSPF的LSA中包含连接的接口、使用的Metric及其它变量信息。OSPF路由器收集链接状态信息并使用SPF算法来计算到各节点的最短路径。LSA也有几种不同功能的报文，在这里简单地介绍一下：

LSATYPE1：由每台路由器为所属的区域产生的LSA，描述本区域路由器链路到该区域的状态和代价。一个边界路由器可能产生多个LSATYPE1。

LSATYPE2：由DR产生，含有连接某个区域路由器的所有链路状态和代价信息。只有DR可以监测该信息。

LSATYPE3：由ABR（区域边界路由器）产生，含有ABR与本地内部路由器连接信息，可以描述本区域到主干区域的链路信息。它通常汇总缺省路由而不是传送汇总的OSPF信息给其它网络。

LSATYPE4：由ABR产生，由主干区域发送到其它ABR，含有ASBR（自治系统边界路由器）的链路信息，与LSATYPE3的区别在于TYPE4描述到OSPF网络的外部路由，而TYPE3则描述区域内路由。

LSATYPE5：由ASBR产生，含有关于自治域外的链路信息。除了存根区域和完全存根区域，LSATYPE5在整个网络中发送。

LSATYPE6：MOSF（多播OSPF）可以让路由器利用链路状态数据库的信息构造用于多播报文的多播发布树。

LSATYPE7：由ASBR产生的关于NSSA的信息。LSATYPE7可以转换为LSATYPE5。

【实验步骤】

练习1分析OSPF报文，理解OSPF工作过程

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A、B、C、D、E、F作为一组进行实验。

注：如果想使用地址本扫描到各个虚拟路由器，那么首先要进入到虚拟路由系统中，执行“zebra -d”和“addrscan”命令。然后使用地址本扫描就可以扫描到虚拟路由器了。

1.主机B、E启动协议分析器，开始分别捕获两块网卡数据，并设置过滤条件（提取OSPF协议）。

2.主机B和主机E启动OSPF协议：

(1)主机B和主机E分别开启Ipv4虚拟路由器。

(2)主机B和主机E在Ipv4虚拟路由器内使用“zebra -d”、“addrscan”和“ospfd -d”命令开启OSPF协议。

(3)主机B和主机E进入配置视图：

① 使用“telnet localhost 2604”命令登陆虚拟路由器终端，密码zebra。

② 使用“enable”，“ configure terminal”，“router ospf”命令进入配置视图。

(4)主机B设置router-id 为1.1.1.1：“ospf router-id 1.1.1.1”。

(5)主机E设置router-id 为2.2.2.2：“ospf router-id 2.2.2.2”。

(6)主机B发布接口网段：

①发布网段172.16.0.0：在命令行方式下输入“network 172.16.0.0/24 area 0.0.0.0”。

②发布网段192.168.0.0：在命令行方式下输入“network 192.168.0.0/24 area 0.0.0.0”。

(7)主机E发布接口网段：

①发布网段172.16.1.0：在命令行方式下输入“network 172.16.1.0/24 area 0.0.0.0”。

②发布网段192.168.0.0：在命令行方式下输入“network 192.168.0.0/24 area 0.0.0.0”。

路由器ID：一个32位的无符号整数，是一台路由器在整个自治系统内惟一的标识。路由器ID或者由管理员手工配置，或者由当前的IP地址中选择一个作为路由器ID。

例如，在命令ospf router-id 1.1.1.1中1.1.1.1为路由器ID值。

区域ID：一个32位的无符号整数，是一个区域在整个自治系统内惟一的标识。

例如，在命令network 192.168.0.0/24 area 0.0.0.0中0.0.0.0为区域ID值。

3.观察主机B、E的OSPF的相关信息，宏观了解该路由器的基本信息：

(1)在用户视图下，通过输入“show ip ospf”察看区域信息。

(2)在用户视图下，通过输入“show ip ospf database”察看链路状态数据库信息。

(3)在用户视图下，通过输入“show ip ospf neighbor”察看邻居信息。

区域：OSPF协议规定需要将一个较大的自治系统划分为若干个区域，这样可以避免在计算最短路径时占用太多的时间，提高OSPF的效率。

链路状态数据库：由整个区域中所有路由器的LSA组成的数据库称为链路状态数据库。

neighbor关系：当选举出DR之后，每台路由器（非DR）就只与DR交换路由信息，而它们之间只发送Hello报文，用来维持它们的邻接关系，这种关系称为neighbor关系。也就是说，当两台路由器是neighbor关系时，它们之间只发送Hello报文，而不交换路由信息，neighbor关系只发生在两台非DR路由器之间。

4.观察路由表，如果出现了OSPF路由，则路由表达到稳定态，表明两台路由器成功建立邻居关系并交换路由信息。

●在用户视图下输入“show ip ospf route”命令，分析主机B和主机E的路由表条目。

5.察看主机B、E捕获的数据，分析OSPF的5种协议报文，理解OSPF的工作过程：

(1)Hello报文

●在会话分析中找到“192.168.0.2—224.0.0.5”会话，观察该会话的第一个报文B\_PKT1，填写表格；

●找出第一个含有字段“邻站IP地址”的报文B\_PKT2，填写表格；

●找出第一个字段“指定路由IP地址”的值不为0.0.0.0报文B\_PKT3，填写表格；

●在会话分析中找到“192.168.0.1—224.0.0.5”会话，观察该会话的第一个报文E\_PKT1，填写表格；

●找出第一个含有字段“邻站IP地址”的报文E\_PKT2，填写表格；

●找出第一个字段“指定路由器IP地址”的值不为0.0.0.0报文E\_PKT3，填写表格。

表18-2实验结果

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 192.168.0.2—224.0.0.5会话 | | | | | | | |
|  | 类型 | 路由器ID | 区域ID | 路由器  优先级 | 指定路由器 | 备份指定路由器 | 邻站IP地址  (若有) |
| B\_PKT1 |  |  |  |  |  |  |  |
| B\_PKT2 |  |  |  |  |  |  |  |
| B\_PKT3 |  |  |  |  |  |  |  |
| 192.168.0.1—224.0.0.5会话 | | | | | | | |
|  | 类型 | 路由器ID | 区域ID | 路由器优先级 | 指定路由器 | 备份指定路由器 | 邻站IP地址ID(若有) |
| E\_PKT1 |  |  |  |  |  |  |  |
| E\_PKT2 |  |  |  |  |  |  |  |
| E\_PKT3 |  |  |  |  |  |  |  |

依据基础理论和对比上面填写的数据，回答下面的问题：

●Hello报文的作用是什么？

●路由器间的邻接关系是怎样建立的？

●指定路由器（DR）、备份指定路由器（BDR）是怎样选举出来的？

(2)DatabaseDescription报文

逐个观察DatabaseDescription报文，注意字段“初始化标识”、“更多标识”、“主/从位”、“报文序号”的变化情况。

●DatabaseDescription报文的作用是什么？

●路由器间的主从关系是怎样确定的？

●OSPF是通过什么方式确保数据的正确传输？

(3)LinkStateRequest报文

观察字段“链路状态类型”、“链路状态ID”、“发送公告的路由器”的值。

●LinkStateRequest报文的作用是什么？

(4)LinkStateUpdate报文

观察该报文各字段的值及LSA信息。

●LinkStateUpdate报文的作用是什么？

●该报文是怎样描述其它路由器信息的？

(5)LinkStateAcks报文

观察该报文各字段的值及LSA信息。

●LinkStateAcks报文的作用是什么？

6.结合上面对报文的分析结果，绘制OSPF工作过程示意图。

思考问题：

1.OSPF使用IP，这样做有何优点？在DatabaseDescription报文中，OSPF是通过什么方式确保数据的正确传输？

练习2分析LSA、LSDB，理解LSA的作用

本练习将主机A、B、C、D、E、F作为一组进行实验。

注：如果想使用地址本扫描到各个虚拟路由器，那么首先要进入到虚拟路由系统中，执行“zebra -d”和“addrscan”命令。然后使用地址本扫描就可以扫描到虚拟路由器了。

1.主机B、E启动协议分析器进行数据捕获并设置过滤条件(提取OSPF协议)。

2.主机B和主机E启动OSPF协议：

(1)主机B和主机E分别开启Ipv4虚拟路由器。

(2)主机B和主机E在Ipv4虚拟路由器内使用“zebra -d”、“addrscan”和“ospfd -d”命令开启OSPF协议。

(3)主机B和主机E进入配置视图：

① 使用“telnet localhost 2604”命令登陆虚拟路由器终端，密码123456。

② 使用“enable”，“ configure terminal”，“router ospf”命令进入配置视图。

(4)主机B设置router-id 为2.2.2.2：“ospf router-id 2.2.2.2”。

(5)主机E设置router-id 为3.3.3.3：“ospf router-id 3.3.3.3”。

(6)进行区域划分：

①主机B在配置视图中，输入“area 1.1.1.1 range 172.16.0.0/24”与“area 0.0.0.0 range 192.168.0.0/24”。

②主机E在配置视图中，输入“area 0.0.0.0 range 192.168.0.0/24”与“area 2.2.2.2 range 172.16.1.0/24”。

(7)主机B发布接口网段：

①发布网段172.16.0.0：在命令行方式下输入“network 172.16.0.0/24 area 1.1.1.1”。

②发布网段 192.168.0.0：在命令行方式下输入“network 192.168.0.0/24 area 0.0.0.0”。

(8)主机E发布接口网段：

①发布网段 192.168.0.0：在命令行方式下输入“network 192.168.0.0/24 area 0.0.0.0”。

②发布网段172.16.1.0： 在命令行方式下输入“network 172.16.1.0/24 area 2.2.2.2”。

3.察看捕获的数据，在链路状态（LSA）类型为1、2、3的报文中任取一个，分析这些链路状态的结构及作用，填写下表：

表18-3实验结果

|  |  |  |  |
| --- | --- | --- | --- |
|  | 发送者 | 所描述的路由 | 传递范围 |
| 类型1(路由器) |  |  |  |
| 类型2(网络) |  |  |  |
| 类型3(网络摘要) |  |  |  |

思考问题：

1.为什么OSPF报文比RIP报文传播得更快？

## 实验19开放式最短路径优先协议(OSPF)Ⅱ

【实验目的】

1.掌握OSPF协议在区域内部SPF（最短路径树）的具体计算过程

【学时分配】

2学时

【实验环境】

该实验采用网络结构五

【实验原理】

### 一.Dijkstra算法在OSPF中的应用

#### 1.形成最短路径树：Dijkstra算法

经过LSP洪泛之后，每个节点都会有整个区域拓扑的一个副本。然而，这个拓扑对找出到其它节点的最短路径是不充分的，还需要使用最短路径树。

一个树是由许多节点和边构成的，其中有一个节点叫做根节点，可以从这个根节点经过一条路径到达其它节点。所谓最短路径树就是指从根节点到所有其它节点之间的路径都是最短的。在链路状态路由选择中，每一个节点都以自身作为根节点生成最短路径树。

Dijkstra算法可以从一个图产生最短路径树。这个算法把图中的节点划分为两个集合：临时的和永久的。它找出与当前永久的节点相邻的所有节点，并使它们是临时的，然后检查它们，如果它们通过某个准则，就变为永久的。下图给出的流程图描述了Dijkstra算法。

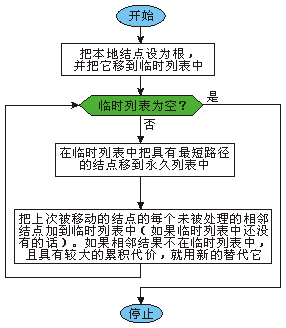


图19-1Dijkstra算法

例如，在拓扑如下图所示的网络中，使用Dijkstra算法计算节点A到其它节点的最短路径的步骤如下（在步骤中，使用灰色的圆圈代表永久节点，使用无色的圆圈代表临时节点，所有永久的节点组成永久列表，所有临时的节点组成临时列表，括号内给出了累积代价）：

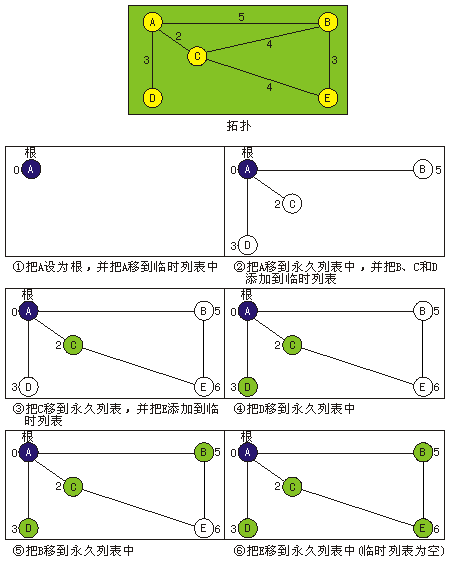


图19-2Dijkstra算法

●让节点A作为树根，把它移到临时列表。这时两个列表是：

|  |  |
| --- | --- |
| 永久列表：空 | 临时列表：A（0） |

●节点A在临时列表中到所有节点有最小累积代价。因此把A移到永久列表，并把A的所有相邻节点加到临时列表。这时两个列表是：

|  |  |
| --- | --- |
| 永久列表：A（0） | 临时列表：B(5)，C(2)，D(3) |

●节点C在临时列表中到所有节点有最小累积代价。因此把C移到永久列表。节点C有3个相邻节点，但节点A已被处理了，因此未被处理的相邻节点仅有B和E。但B已经在临时列表中，且累积代价是5。节点A可以通过C到达节点B，累积代价是6。由于5小于6，因此就保留具有累积代价为5的节点B在临时列表中而不替换它，添加节点E到临时列表。这时两个列表是：

|  |  |
| --- | --- |
| 永久列表：A（0），C(2) | 临时列表：B(5)，D(3)，E(6) |

●节点D在临时列表中到所有节点有最小累积代价。因此把D移到永久列表。节点D没有未被处理的相邻节点要加到临时列表中。这时两个列表是：

|  |  |
| --- | --- |
| 永久列表：A（0），C（0），D(3) | 临时列表：B(5)，E(6) |

●节点B在临时列表中到所有节点有最小累积代价。因此把B移到永久列表。然后把B的所有未被处理的相邻节点加到临时列表中（现在仅有节点E）。但E(6)已经在临时列表中，且具有更小的累积代价。到节点E（它是B的相邻节点）的累积代价是8。把节点E(6)保留在临时列表中。这时两个列表是：

|  |  |
| --- | --- |
| 永久列表：A（0），B(5)，C(2)，D(3) | 临时列表：E(6) |

●节点E在临时列表中到所有节点有最小累积代价。把E移到永久列表。节点E没有相邻节点。现在临时列表已空。最后的列表是：

|  |  |
| --- | --- |
| 永久列表：A（0），B(5)，C(2)，D(3)，E(6) | 临时列表：空 |

#### 2.从最短路径树计算路由表

每一个节点使用最短路径树构造其路由表。路由表给出从根到每一个节点的代价。下表为节点A的路由表。

表19-1节点A的路由表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 节点 | 代价 | 下一路由器 | 节点 | 代价 | 下一路由器 |
| A | 0 | — | D | 3 | — |
| B | 5 | — | E | 6 | C |
| C | 2 | — |  |  |  |

【实验步骤】

练习1分析OSPF的计算过程

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A、B、C、D、E、F作为一组进行实验。

注：如果想使用地址本扫描到各个虚拟路由器，那么首先要进入到虚拟路由系统中，执行“zebra -d”和“addrscan”命令。然后使用地址本扫描就可以扫描到虚拟路由器了。

1.主机B、D、E启动协议分析器，开始分别捕获两块网卡数据，并设置过滤条件（提取OSPF协议）。

2.主机B、D、E配置虚拟路由器IP地址。

(1)主机B、主机D和主机E分别开启虚拟路由器。

(2)使用账号“root”、密码“12345678”进入系统。

(3)主机B、主机D和主机E分别执行“ospfconfig”命令，还原OSPF设置。

3.主机B和主机D和主机E启动OSPF协议：

(1)主机B和主机E分别开启Ipv4虚拟路由器。

(2)主机B和主机E在Ipv4虚拟路由器内使用“zebra -d”、“addrscan ”和“ospfd -d”命令开启OSPF协议。

(3)主机B和主机D和主机E进入配置视图：

① 使用“telnet localhost 2604”命令登陆虚拟路由器终端，密码zebra。

② 使用“enable”，“ configure terminal”，“router ospf”命令进入配置视图。

(4)主机B设置router-id 为1.1.1.1：“ospf router-id 1.1.1.1”。

(5)主机D设置router-id 为2.2.2.2：“ospf router-id 2.2.2.2”。

(6)主机E设置router-id 为3.3.3.3：“ospf router-id 3.3.3.3”。

(7)主机B发布接口网段：

①发布网段172.16.0.0：在命令行方式下输入“network 172.16.0.0/24 area 0.0.0.0”。

②发布网段192.168.0.0：在命令行方式下输入“network 192.168.0.0/24 area 0.0.0.0”。

(8)添加主机D的接口：

①发布网段172.16.1.0：在命令行方式下输入“network 172.16.1.0/24 area 0.0.0.0”。

②发布网段192.168.0.0：在命令行方式下输入“network 192.168.0.0/24 area 0.0.0.0”

(9)添加主机E的接口：

①发布网段172.16.1.0：在命令行方式下输入“network 172.16.1.0/24 area 0.0.0.0”。

②发布网段172.16.0.0：在命令行方式下输入“network 172.16.0.0/24 area 0.0.0.0”。

注：在虚拟路由器路由器配置模式下，在命令行输入exit可以退出到配置模式，然后在配置模式输入interface eth0，就可以进入到接口配置模式。

（10）设置主机B接口的花销，首先进入接口配置模式，在命令行输入interface eth0,设置IP为172.16.0.1接口的花销为100：在命令行方式下输入“ip ospf cost 100”。

（11）设置主机B接口的花销，首先进入接口配置模式，在命令行输入interface eth1,设置IP为192.168.0.1接口的花销为500：在命令行方式下输入“ip ospf cost 500”。

(12)设置主机D接口的花销，首先进入接口配置模式，在命令行输入interface eth0,

设置IP为172.16.1.2接口的花销为200：在命令行方式下输入“ip ospf cost 200”。

(13)设置主机D接口的花销，首先进入接口配置模式，在命令行输入interface eth1,

设置IP为192.168.0.2为接口的花销为500：在命令行方式下输入“ip ospf cost 500”。

（14）设置主机E接口的花销，首先进入接口配置模式，在命令行输入interface eth0,

设置IP为172.16.0.3为接口的花销为100：在命令行方式下输入“ip ospf cost 100”。

（15）设置主机E接口的花销，首先进入接口配置模式，在命令行输入interface eth1,

设置IP为172.16.1.3为接口的花销为200：在命令行方式下输入“ip ospf cost 200”。

4.用ping命令检测各路由器的连通性，如果能够达到全网互通，说明OSPF已经完成了SPF最短路径树的计算。

5.SPF的计算过程分析-带权有向图的生成：

这里以虚拟路由器BV为例利用邻居关系列表和接口权值来生成一张带权有向图。

首先在虚拟路由器BV上察看LSDB（在命令行方式下，输入“Telnet localhost 2604”然后输入密码“zebra”，输入“enable”进入特权模式，然后输入“show ip ospf database”），如下图所示：

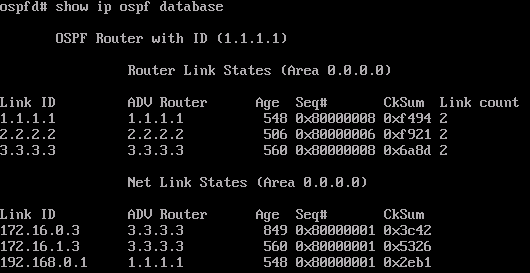


图19-3LSDB

由上表可知区域0.0.0.0内包含三个路由器（1.1.1.1，2.2.2.2，3.3.3.3），并划分三个子网（172.16.0.0/24，172.16.1.0/24，192.168.0.0/24）。

分别在主机B、D、E上察看邻居关系列表（在命令行方式下，输入“show ip ospf neighbor”）。以虚拟路由器BV为例:

BV的邻居关系列表的内容如下：

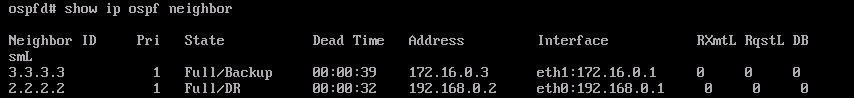


图19-4  邻居关系列表

由上面的信息可绘制出三个路由器间的关系图，如下所示：

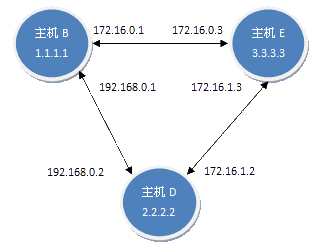


图19-5三个路由器间的关系图

分别在虚拟路由器BV、DV、EV上察看接口信息中的COST值,下面以虚拟路由器BV为例（在命令行方式下，输入“telnet localhost 2604”，输入密码“zebra”，输入“enable”进入特权模式，然后使用“show ip ospf interface”来查看cost）。

主机B的接口信息如下：

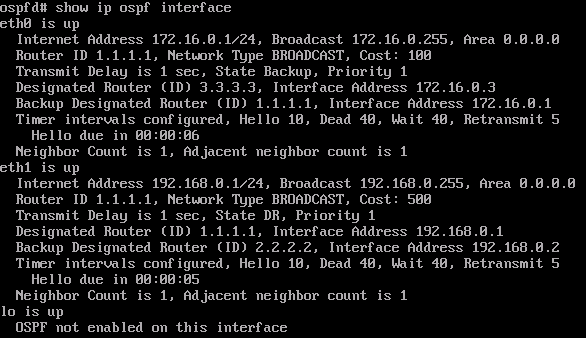


图19-6接口信息

由上面的信息可知三个路由器间的COST，下图为三个路由器带权有向图：

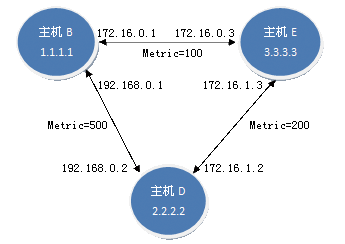


图19-7三个路由器带权有向图

6.SPF的计算过程分析-最短路径树计算的过程：

这里以主机B为根来分析最短路径树的生成过程，由于是最短路径树，所以要求从主机B到网络中任意一点的距离都是最短的，具体计算可以通过对metric值的累加来完成。

我们把主机B到网络中各点的下一跳和metric值列在下表中，以方便进行最短路径树的计算：

表19-2跳与metric

| 目的 | 下一跳（路径） | OSPFmetric |
| --- | --- | --- |
| 主机D（路由器2） | 192.168.0.2（2.2.2.2） | 500 |
| 172.16.0.3（3.3.3.3，2.2.2.2） | 100+200=300 |
| 网络192.168.0.0/24 | 直连 | 500 |
| 172.16.0.3（3.3.3.3，2.2.2.2） | 100+200+500=800 |
| 主机E（路由器3） | 172.16.0.3（3.3.3.3） | 100 |
| 192.168.0.2（2.2.2.2，3.3.3.3） | 500+200=700 |
| 网络172.16.0.0/24 | 直连 | 100 |
| 192.168.0.2（2.2.2.2，3.3.3.3） | 500+200+100=800 |
| 网络172.16.1.0/24 | 192.168.0.2（2.2.2.2） | 500+200=700 |
| 172.16.0.3（3.3.3.3） | 100+200=300 |

参照上表中到同一目的地的不同下一跳和OSPFmetric值，可以得出从主机B到网络中各点的最短路径，下图所示的是以主机B为根的最短路径树。

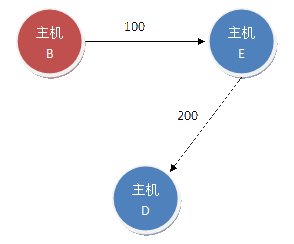


图19-8根的最短路径树

在虚拟路由器BV上，用traceroute命令来验证最短路径树的形状。

在命令行方式下，输入命令“traceroute 172.16.1.2”。

●由上述命令的显示结果说明主机B到主机D的路径。

7.参照步骤4的过程，分别绘制以主机D和主机E为根的最短路径树。

8.改变主机B和主机D间、主机D和主机E间接口的OSPF路由的COST值，重新计算SPF，加深对SPF算法的理解。

(1)参照实验步骤1中设置接口权值的方法，将主机B和主机D间接口的OSPF路由的COST值设置为200，将主机D和主机E间接口的OSPF路由的COST值设置为500。

(2)参照步骤4的过程，分别绘制以主机B、主机D和主机E为根的最短路径树。使用traceroute命令来验证各路由器（主机B、D、E）最短路径树的形状。

9.结合步骤4—步骤8的实验结果，说明OSPF协议在区域内部SPF（最短路径树）的具体计算过程。

完成本实验后，需要还原虚拟路由器快照后才可以进行其他的虚拟路由器实验。

思考问题：

1.OSPF在大型网络中可能遇到哪些问题？

## 实验20代理

【实验目的】

1.了解什么是代理

2.了解代理状态下的通信方式

3.了解SOCKS5协议

【学时分配】

2学时

【实验环境】

该实验采用网络结构一

【实验原理】

### 一.代理服务器和代理

代理服务器是介于代理客户端和代理客户访问目的端服务器之间的一台服务器。如使用代理方式浏览网页时，浏览器不是直接到网站的服务器去取回网页，而是向代理服务器发出请求，由代理服务器到要访问的网站服务器取回你所需要的信息，并传送给本地的浏览器。

代理就是指那些通过代理服务器进行互联网访问的一种网络访问方式，常用的代理方式有HTTP代理和SOCKS（v4/v5）代理。

### 二.使用代理的优点

代理服务器（ProxyServer）是Internet链路级网关所提供的一种重要的安全功能，它的工作主要在开放系统互联（OSI）型的对话层，主要的功能是突破自身IP访问限制，通过代理可以做到：

(1)突破网络提供商的IP封锁

大多网络提供商基于安全方面的考虑对一些外部网站做了相应的限制，用来阻止用户的访问。这种限制是人为的，不同提供商对地址的封锁是不同的。所以不能访问时可以选择一个能够访问限制网站的代理服务器，通过代理方式来进行访问。

(2)提高访问速度

通常代理服务器都设置一个较大的硬盘缓冲区，当有外界的信息通过时，同时也将其保存到缓冲区中，当其他用户再访问相同的信息时，可以直接由缓冲区中取出信息，传给用户，以提高访问速度（这也会带来一定的弊端，可能造成代理用户访问到过期信息）。

(3)访问一些私有机构的内部资源

用户可以通过代理服务器访问一些私有机构的内部资源（前提是这个代理服务器拥有这些内部资源的访问权限）。

(4)隐藏真实IP

上网者也可以通过这种方法隐藏自己的IP，免受攻击，因为被访问者永远只能看到代理服务器的地址。

### 三.SOCKS和HTTP的区别

SOCKS是一组由Internal工程工作小组（IETF）所开发出来的开放软件开放标准，用来处理网络安全的事宜。SOCKS象一堵墙被夹在Internal服务器和客户端之间，对于出入企业网络的资讯提供流量和安全的管理。SOCKS这个名词并不是一组英文字头的缩写，而是一个和TCP/IP的Socket端口有关的安全标准，一般防火墙系统通常是象网关（Gateway）一样是作用在OSI模型的第七层也就是应用层上，对TCP/IP的高级协议，如Telnet、FTP、HTTP和SMTP加以管制，而SOCKS作用在OSI模型的第四层也就是会话层上，象一个代理一样对客户端到服务器端或服务器和服务器之间的数据联系，提供安全上的服务。由于SOCKS作用在会话层上，因此它是一个提供会话层到会话层间安全服务的方案，不受高层应用程序变更的影响。Socks代理只是简单地传递数据包，而不必关心是何种应用协议(比如FTP、HTTP和NNTP请求)，所以Socks代理服务器比应用层代理服务器要快。

被代理端与代理服务器通过“SOCKS4/5代理协议”进行通迅(具体协议内容可查看RFC文档)。SOCKS4代理协议可以说是对HTTP代理协议的加强，它不仅是对HTTP协议进行代理，而是对所有向外的连接进行代理，是没有协议限制的。也就是说，只要你向外连接，它就给你代理，并不管你用的是什么协议，极大的弥补了HTTP代理协议的不足，使得很多在HTTP代理情况下无法使用的网络软件都可以使用了。(例如：QQ、MSN、ICQ等软件)SOCKS5代理协议又对前一版进行了修改，增加了支持UDP代理及身份验证的功能。它不是“协议代理”，所以它会对所有的连接进行代理，而不管用的是什么协议。

SOCKS5是一种透明的代理协议,而且很好的解决了认证,保密,以及proxy接力等问题netscape,ie等可以直接使用这种代理,而且可以使用SOCKS5的client软件使那些不直接支持SOCKS5proxy的internet软件的通过代理访问internet.更重要的是,SOCKS5支持UDP协议的代理,从而成为很多firewall后面的人使用ICQ不可缺少的东西。

### 四.SOCKS5通信协议简介

很多组织的内部通过防火墙隔离与外界的通信，一般都可以控制HTTP、FTP和SMTP的代理服务，然而随着日益广泛的应用，就需要一种通用的协议来完成代理的功能，SOCKS代理规范就此诞生。先是比较早的SOCKS4，到现在的SOCKS5。SOCKS5比SOCKS4增加了UDP的代理功能。SOCKS5协议规定了代理的通信逻辑，它支持权限认证管理，还可以进行端口限制，通常其服务端口为1080。此外，为保证数据不会在第一级就丢失，该协议必须通过TCP方式进行通信。

SOCKS5客户端与服务器的通信过程如下：

●客户建立与服务器的连接，发送协议协商命令，命令的格式如下图所示：

|  |  |  |
| --- | --- | --- |
| VER | NMETHODS | METHODS |
| 1 | 1 | 1~255 |

图20-1命令的格式

字段名及解释如下：

VER：协议版本对于SOCKS5其值为0x05

NMETHODS：客户端支持的认证方式的数量（即METHODS中含有的字节数（octets））

METHODS：客户端支持的认证方式列表

下表给出了SOCKS5定义的所有有效认证方法，但不同的服务器可以有自己的定义：

表20-1SOCKS5有效认证方法

|  |  |
| --- | --- |
| METHOD | 含义 |
| 0x00 | 不需要身份确认 |
| 0x01 | 通过GSSAPI协议 |
| 0x02 | 通过账号密码认证 |
| 0x03 | 到0x7F用IANA组织分配 |
| 0x80 | 到0xFE保留给私人用 |
| 0xFF | 没有可接受的方法 |

●当服务器接受到客户的以上消息后就会回复如下图所示的格式的数据：

|  |  |
| --- | --- |
| VER | METHOD |
| 1 | 1 |

图20-2数据格式

其中，版本号要和客户请求的一样，如果不一样则表示服务器支持的版本号不对。而第二个参数与客户一样为表20-1所列情况的一种。

●一旦服务器和客户经过上面的认证后，如果上面的方法是0x02即通过账号密码认证，便开始进行身份认证。账号和密码的认证按下面的步骤进行，首先客户向服务器发送明码的账号和密码，格式如下图所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| VER | ULEN | UNAME | PLEN | PASSWD |
| 1 | 1 | 1~255 | 1 | 1~255 |

图20-3账号和密码的格式

其中，版本号为0x01而不是前面的0x05，这一点要特别注意，后面的一个字段表示名字的长度，然后就是账号，接着密码长度，然后是密码。这里所有的字符串不需要在末尾肩上NULL字符。服务器收到上面协商数据后给客户端返回：

|  |  |
| --- | --- |
| VER | STATUS |
| 1 | 1 |

图20-4数据格式

如果Status为0，表示验证成功，否则返回用各个服务器自己定义的错误编码。这种方法的缺点是安全行很差，尤其是在可以SNIFF的内部网，用户的账号密码很容易被别人监控。

●一旦通过了账号和密码验证，就可以进行数据通信了，通信的格式如下图所示：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| VER | CMD | RSV | ATYP | DST.ADDR | DST.PORT |
| 1 | 1 | 0x00 | 1 | 可变长 | 2 |

图20-5通信的格式

各个字段的解释如下：

其中VER和前面一样是0x05（对于SOCKS5）。

CMD表示客户请求的命令类型，有下面几种：

CONNECT 0x01

BIND 0x02

UDPASSOCIATE0x03

ATYP表示地址类型，也有三种可能性：

IPv4addrss 0x01

DOMAINNAME 0x03

IPv6address 0x04

如果ATYP为IPv4则DST.ADDR字段长度为四个字节，如果为IPv6则为16个字节；如果ATYP为DOMAINNAME则表示长度是可变的，但一般不会超过256字节，它的第一个字节就表示该域的长度。

DST.PORT表示端口号，长度为固定的2个字节。

下面来简单介绍一下三个命令的不同之处：

CONNECT模式：表示客户请求的是去连接防火墙外面的套接字。

BIND模式：表示客户建立套接字并等待外面的程序的连入。该命令的典型应用是在FTP的主动模式，就是需要自己建立套接字，等待服务器的连接。

UDPASSOCIATE模式：UDP是刚刚增加的功能，可以发送和接收UDP数据包。

### 五.SOCKS5对UDP协议的支持

●服务器收到UDP数据包后做出的回应格式如下图所示：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| VER | REP | RSV | ATYP | BND.ADDR | BND.PORT |
| 1 | 1 | 0x00 | 1 | 可变长 | 2 |

图20-6回应格式

上面的数据格式发送的基本一致，唯一的不同是CMD字段换成了REP字段，该字段表示服务器对客户请求的回应，其可能的值及含义如下表所示。

表20-2REP字段可能值

| 取值 | 含义 |
| --- | --- |
| 0x00 | 成功 |
| 0x01 | 一般失败 |
| 0x02 | 该功能被禁止 |
| 0x03 | 目的网络不可达 |
| 0x04 | 目的主机不可达 |
| 0x05 | 远端拒绝连接 |
| 0x06 | TTL过期 |
| 0x07 | 不支持的命令类型 |
| 0x08 | 不支持的地址类型 |
| 0x09到0xFF | 暂时未分配 |

对于CONNECT命令可以在接收到此确认后，把数据发到服务器就行了，服务器会自动发送到远端，然而对于BIND命令在收到上面的确认信息后，还必须等待服务器端发回来的所有客户连接信息。不过从服务器发回的消息里，包含有可能是服务器用于连接的套接字的地址（域名）和端口。

基于UDP的数据每次发送都必须加上一定的头信息。该头的结构如下：

该信息头是以两个00开始的，这样的好处是比较容易区分。这里的Data字段不需要知道长度，以为服务器端会有结构长度的，这里还有一个Frag字段表示，你可以把几个帧拼成一个，再以UDP的格式发送。这样做可以把小帧变大帧，有利于效率提高。但一般不太支持，所以尽量不要使用。从服务器收到的消息也是这样的，必须先解开再传给下一级。服务器并不接收所有UDP包，而只接收客户登记过的端口的数据包。

【实验步骤】

练习1页面访问

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。

1.主机A启动并默认安装ccproxy，完成后启动协议分析器开始捕获数据。

2.主机B清空IE缓存，依次选择IE的菜单栏的“工具/Internet选项/连接选项卡/局域网设置”选中“为LAN使用代理服务器”前的复选框，并在下面的地址栏中输入主机A的IP地址，端口号输入808，点击确定后退出。启动协议分析器，开始捕获数据。

3.主机B在“地址”框中输入http://*服务器的ip*/complexpage.htm，并连接，服务器IP默认为172.16.0.253。

4.主机A和主机B停止捕获数据，并保存会话命令。

5.结合主机A捕获的会话分类树中的“TCP其它会话”和“HTTP会话”，会看到主机B连接到主机A的808端口后，主机A又去连接172.16.0.253的80端口，而对比主机A到主机B，主机B到172.16.0.253服务器的数据包会发现，最后携有HTTP协议数据的TCP报文是从172.16.0.253的80端口返回到主机A，又从主机A的808端口将TCP数据传给主机B。

6.查看主机B捕获的会话分类树中的“TCP其它会话”会看到主机B连接主机A的808端口，主机A把TCP数据又通过808端口返回给主机B，主机B并没有直接连接到172.16.0.253获取页面。

7.通过分析会话数据，画出网页访问过程的流程图。

8.恢复实验环境

卸载软件ccproxy，取消“工具/Internet选项/连接选项卡/局域网设置”中的“为LAN使用代理服务器”设置，恢复原始设置。

思考问题：

1.简述使用代理的优点和缺点。

2.代理与网络地址转换有何同异。

## 实验21IPv4综合实验

【实验目的】

1.提高拓扑结构设计能力

2.提高网络应用程序设计的能力

3.加深对某些基础协议的理解

【学时分配】

4学时

【实验环境】

网络结构一环境下自主设计，每组可以使用只能我那个裸设备和组空设备。

【实验需求】

每组成员利用组控设备自行设计网络结构，使该网络满足如下需求：

●该网络能够明显的区分出内网和外网。

●内网主机之间可以相互通信；内网主机可以主动访问外网资源，但内网主机不能够被外网主机直接访问。

●该网络对外要提供一个服务器，服务器有一个外网IP地址，可以被内外网上的所有主机访问。

●每个组设计的网络都要含有一个外网接口，将各个组的外网接口连接起来，可以形成一个更大的网络。

●通过编写网络应用程序，实现点对点的通信。这里所说的点对点通信是指同一内网的任意两台主机之间的通信，以及一个组的一台内网主机和另一个组的一台内网主机之间的通信。

【实验步骤】

练习1IPv4综合实验

1.按照实验需求设计网络结构，绘制网络拓扑结构图并搭建网络环境。

2.自行设计网络拓扑结构的验证方法并验证网络连接的正确性。

3.所有实验组搭建完自身网络后，将各个网络的外网接口连接到中心设备上，形成更大的网络。

4.使用内网主机访问外网资源（本组及其它组所设置的服务器上的资源）。

5.在本组内网主机之间自行验证点对点通信。

6.任意两个组的内网主机之间验证点对点通信。

**拓扑结构参考:**

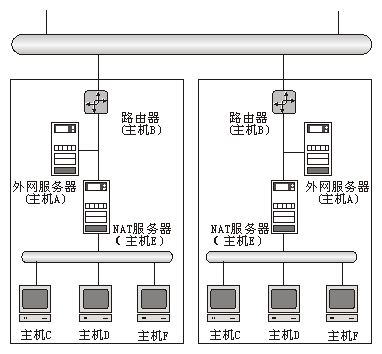


图21-1拓扑结构参考图

第2部分网络攻防与故障

## 实验22ARP地址欺骗

【实验目的】

1.加深对ARP高速缓存的理解

2.了解ARP协议的缺陷

3.增强网络安全意识

【学时分配】

4学时

【实验环境】

该实验采用网络结构二

【实验原理】

### 一.ARP地址欺骗

在以太网中，ARP缓存表是IP地址和MAC地址的映射关系表，任何实现了IP协议栈的设备，一般情况下都通过该表维护IP地址和MAC地址的对应关系，这是为了避免ARP解析而造成的广播数据报文占用过多的网络带宽。在一般情况下，ARP表是通过两个途径建立的：

●主动解析，如果一台计算机想与另外一台不知道MAC地址的计算机通信，则该计算机主动发ARP请求；

●被动请求，如果一台计算机接收到了另一台计算机的ARP请求，则在本地建立请求计算机的IP地址和MAC地址的对应表。

因此，针对ARP表项，有一种攻击方式就是误导计算机建立错误的ARP表。假设有三台计算机A，B，C，其中B已经正确建立了A和C的ARP表项。假设A是攻击者，此时，A发出一个ARP请求报文，该ARP请求报文这样构造：

●源IP地址是C的IP地址，源MAC地址是A的MAC地址；

●请求的目标IP地址是B的IP地址。

这样计算机B在收到这个ARP请求报文后（ARP请求是广播报文，网络上所有设备都能收到），发现C的ARP表项已经在自己的缓存中，但MAC地址与收到的请求的源MAC地址不符，于是根据ARP协议的实现原理，使用ARP请求的源MAC地址（即A的MAC地址）更新自己的ARP表。

这样B的ARP缓存中就存在这样的错误ARP表项：C的IP地址跟A的MAC地址对应。这样的结果是，B发给C的数据都被计算机A接收到。

【背景描述】

流经主机A和主机C的数据包被主机D使用ARP欺骗进行截获和转发。

流经主机E（172.16.0.2接口）和主机F的数据包被主机B（172.16.0.1接口）使用ARP欺骗进行截获和转发。

【实验步骤】

练习1ARP地址欺骗

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A、C和D作为一组，主机B、E和F作为一组。现仅以主机A、C、D所在组为例，其它组的操作参考主机A、C、D所在组的操作。

1.主机A和主机C使用“arp-a”命令察看并记录ARP高速缓存。

2.主机A、C启动协议分析器开始捕获数据并设置过滤条件（提取ARP协议和ICMP协议）。

3.主机Aping主机C。观察主机A、C上是捕获到的ICMP报文，记录MAC地址。

4.主机D启动协议编辑器向主机A编辑ARP请求报文（暂时不发送）。其中：

MAC层：

源MAC地址：主机D的MAC地址。

目的MAC地址：主机A的MAC地址。

ARP层：

发送端MAC地址：主机D的MAC地址。

发送端IP地址：主机C的IP地址。

目的端MAC地址：000000-000000。

目的端IP地址：主机A的IP地址。

5.主机D向主机C编辑ARP请求报文（暂时不发送）。其中：

MAC层：

源MAC地址：主机D的MAC地址。

目的MAC地址：主机C的MAC地址。

ARP层：

发送端MAC地址：主机D的MAC地址。

发送端IP地址：主机A的IP地址。

目的端MAC地址：000000-000000。

目的端IP地址：主机C的IP地址。

6.同时发送第4步和第5步所编辑的数据包。

「注意」为防止主机A和主机C的ARP高速缓存表被其它未知报文更新，可以定时发送数据包（例如：每隔500ms发送一次）。

7.观察并记录主机A和主机C的ARP高速缓存表。

8.在主机D上启动静态路由服务（方法：在命令行方式下输入“staticroute\_config”），目的是实现数据转发。

9.主机D禁用ICMP协议。

(1)在命令行下输入“mmc”，启动微软管理控制台。

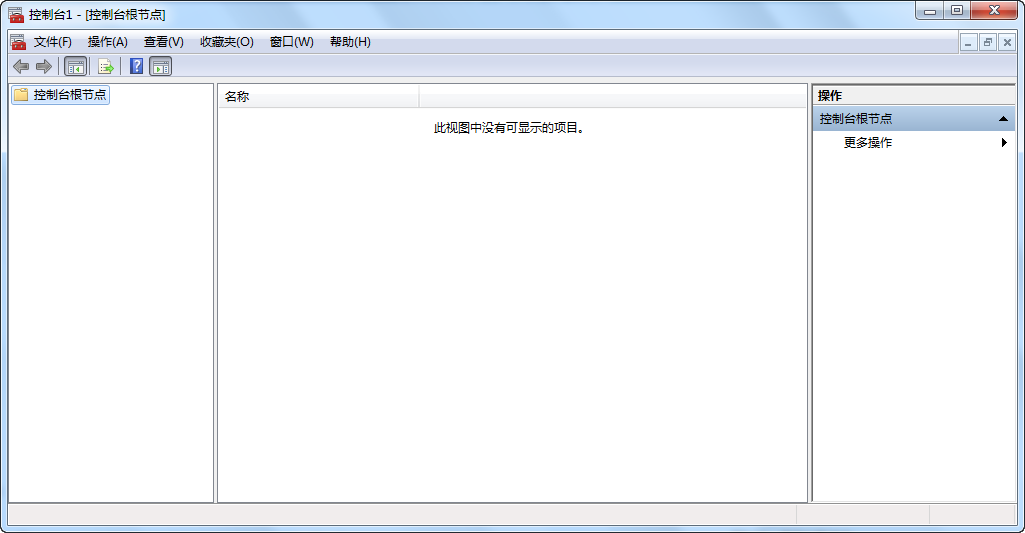


图22-1禁用ICMP协议

(2)导入控制台文件。

单击“文件(F)\打开(O)...”菜单项来打开“c:\WINDOWS\sysWOW64\IPSecPolicy\stopicmp.msc”。

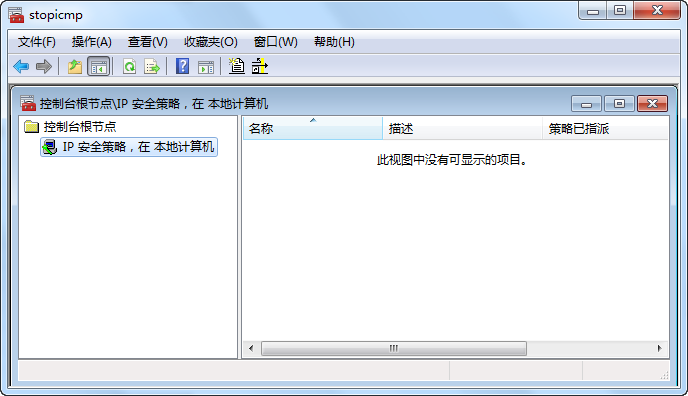


图22-2导入控制台文件

(3)导入策略文件。

单击“操作(A)\所有任务(K)\导入策略(I)...”菜单项来打开“c:\WINDOWS\ sysWOW64\IPSecPolicy\stopicmp.ipsec”。此命令执行成功后，在策略名称列表中会出现“禁用ICMP”项。

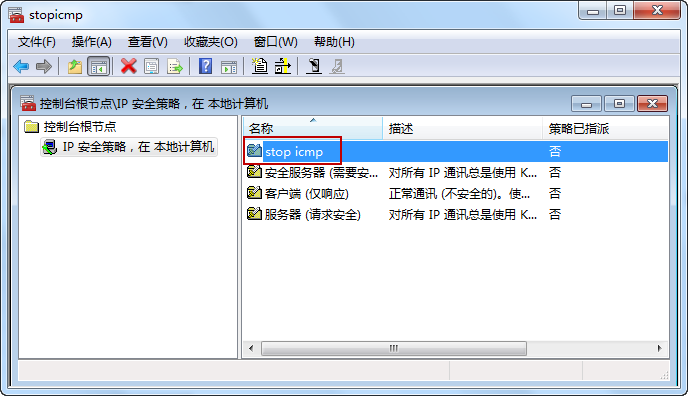


图22-3导入策略文件

(4)启动策略。

用鼠标选中“禁用ICMP”项，单击右键，选择“分配(A)”菜单项。

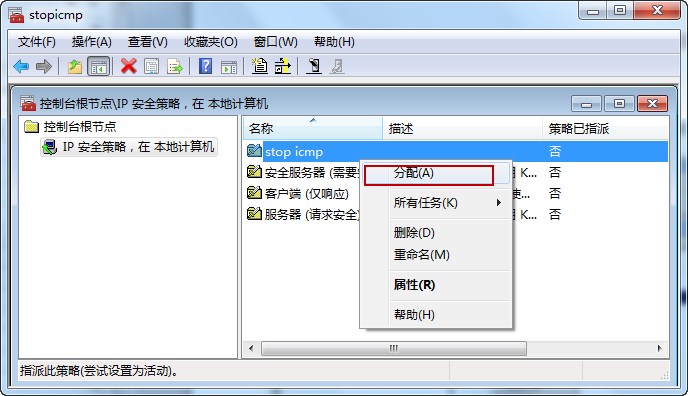


图22-4启动策略

10.主机A上ping主机C（“ping主机C的IP地址–n1”）。

11.主机A、C停止捕获数据，分析捕获到的数据，并回答以下问题：

●主机A、C捕获到的ICMP数据包的源MAC地址和目的MAC地址是什么？

●结合主机A和主机C捕获到的数据包，绘制出第10步发送的ICMP数据包在网络中的传输路径图。

12.主机A首先使用arp –d清除缓存，主机A使用“arp –s C的IP地址C的MAC地址”添加一条静态ARP缓存，重复步骤10的操作。

●主机A、C捕获到的ICMP数据包的源MAC地址和目的MAC地址是什么？

●结合主机A和主机C捕获到的数据包，绘制出第10步发送的ICMP数据包在网络中的传输路径图。

13.主机D取消对ICMP的禁用。

在微软管理控制台（mmc）上，用鼠标选中“禁用ICMP”项，单击右键，选择“未分配(U)”菜单项。

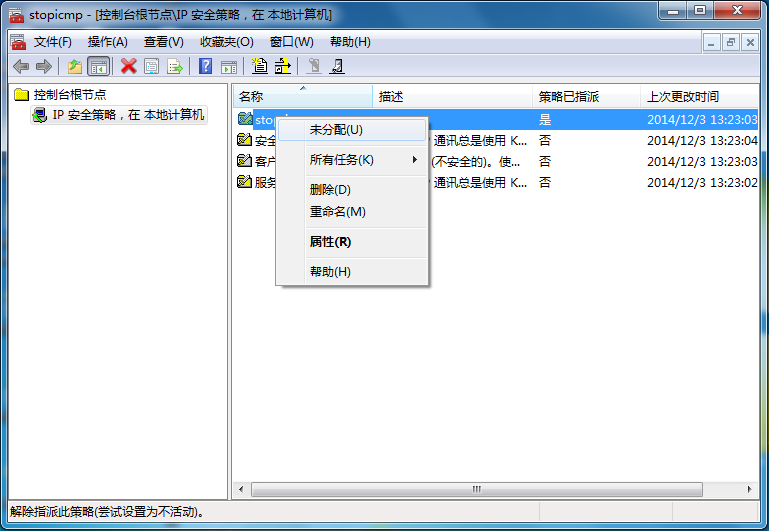


图22-5取消对ICMP的禁用

14.主机D在命令行方式下输入recover\_config命令，停止静态路由服务。

思考问题：

在主机A上使用“arp–s主机C的IP主机C的MAC”命令，在主机C上使用“arp–s主机A的IP主机A的MAC”命令，分别为主机A和主机C添加一条静态ARP高速缓存条目，ARP欺骗是否还能成功？你认为添加静态ARP高速缓存条目能从根本上解决ARP欺骗吗？

## 实验23ICMP重定向

【实验目的】

1.加深对ICMP协议的理解

2.了解简单的信息窃取技术

3.增强网络安全意识

【学时分配】

2学时

【实验环境】

该实验采用网络结构二

【实验原理】

### 一.ICMP重定向

在Internet上，主机数量要比路由器多出许多，为了提高效率，主机都不参与路由选择过程。主机通常使用静态路由选择。当主机开始联网时，其路由表中的项目数很有限，通常只知道默认路由的IP地址。因此主机可能会把某数据报发送到一个错误的路由，其实该数据报本应该是发送给另一个路由器的。在这种情况下，收到该数据的路由器会把数据报转发给正确的路由器，同时，它会向主机发送ICMP重定向报文，来改变主机的路由表。

路由器发送ICMP重定向报文给主机来指出存在一个更好的路由。ICMP重定向报文的报文格式如下图所示。

|  |  |  |
| --- | --- | --- |
| 类型=5 | 代码=0-3 | 检验和 |
| 应该使用的路由器IP地址 | | |
| IP首部（包括选项）+原始IP数据报中数据的前8字节 | | |

图23-1ICMP重定向报文格式

当IP数据报应该被发送到另一个路由器时，收到数据报的路由器就要发送ICMP重定向差错报文给IP数据报的发送端。ICMP重定向报文的接收者必须查看三个IP地址：

导致重定向的IP地址（即ICMP重定向报文的数据位于IP数据报的首部）；

发送重定向报文的路由器的IP地址（包含重定向信息的IP数据报中的源地址）；

应该采用的路由器的IP地址。

在ICMP重定向报文中，代码字段可以为以下值：

代码为0：路由器发送这个ICMP重定向报文来指出有一个到达目标网络的更好方法。

代码为1：路由器发送这个ICMP重定向报文来指出有一个到达目标主机的更好方法。

代码为2：路由器发送这个ICMP重定向报文来指出有一个到达使用希望的区分服务目的网络更好的方法。

代码为3：路由器发送这个ICMP重定向报文来指出有一个到达使用所要求的区分服务的目标主机更好的方法。

例如，在如下图所示的拓扑结构中，在主机A与主机D正常的通信过程中，数据经过路由器R进行转发。

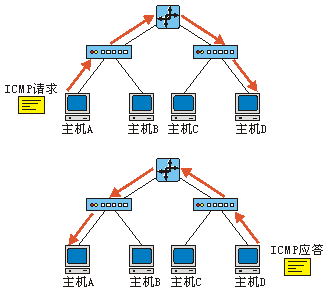


图23-2正常通信过程

现在假设主机B为攻击者，想要窃取主机A与主机D之间的通信数据。主机B模拟路由器R的身份向主机A发送一个ICMP重定向报文，此时，主机A被欺骗了，它会认为与主机D通信有一条更好的路由，这条路由经过主机B，因此它会将数据包发送给主机B。如下图所示：

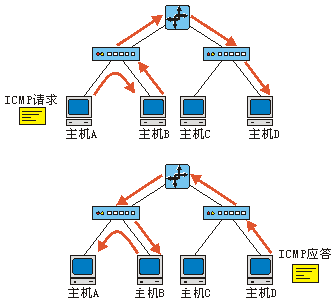


图23-3重定向后的通信过程

【背景描述】

主机A、C、D属172.16.1网段，主机E、F属172.16.0网段，主机B作为路由器连接172.16.1与172.16.0网段，正常情况下主机D和主机E可以通过路由器B通信，现在主机C的使用者要窃取主机D和主机E之间的通信数据。

【实验步骤】

练习1利用ICMP重定向进行信息窃取

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.主机D启动ICMP重定向功能，在命令行方式下输入“icmpredirect\_configenable”。

2.主机B启动静态路由服务，在命令行方式下输入“staticroute\_config”。

3.主机A、B、D、E、F启动协议分析器开始捕获数据并设置过滤条件（提取ICMP协议）。

4.主机Dping主机E（172.16.0.2），察看主机A、B、D、E、F捕获到的数据。

●通过此ICMP及其应答报文的MAC地址，绘制其在网络中的传输路径图。

5.主机C启动协议编辑器，模拟主机B身份（172.16.1.1）编辑向主机D发送的ICMP重定向报文，其中：

MAC层：

源MAC地址：主机C的MAC地址。

目的MAC地址：主机D的MAC地址。

IP层：

源IP地址：主机B的IP地址（172.16.1.1）。

目的IP地址：主机D的IP地址（172.16.1.4）。

ICMP层：

类型：5。

代码：1。

网关地址：主机C的IP地址（172.16.1.3）。

ICMP数据：

伪造的主机D向主机E发送的ping请求报文的一部分（包括整个IP首部和数据的前8个字节）。

「注」为防止主机D的路由表被其它未知数据包更新，可以定时发送此报文（例如：每隔500ms发送一次）。

6.在主机C上启动静态路由服务（方法：在命令行方式下，输入“staticroute\_config”），并添加一条静态路由条目（方法：在命令行方式下，输入“route add 172.16.0.0 mask 255.255.255.0 172.16.1.1 metric2”），目的是实现数据转发。

7.主机C禁用ICMP协议，避免主机C发送ICMP回显应答。

(1)在命令行下输入“mmc”，启动微软管理控制台。

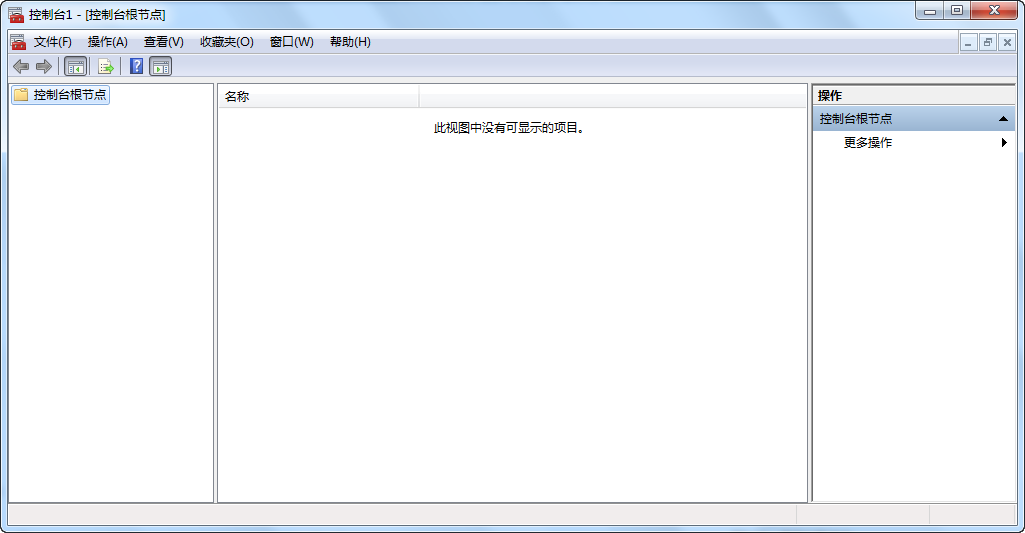


图23-4禁用ICMP协议

(2)导入控制台文件。

单击“文件(F)\打开(O)...”菜单项来打开“c:\WINDOWS\SysWOW64 \IPSecPolicy\stopicmp.msc”。

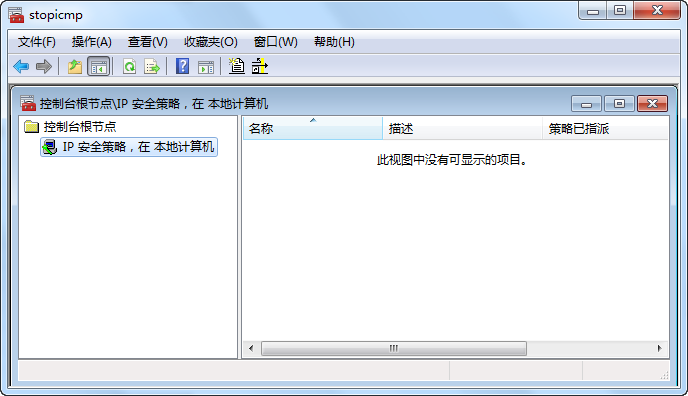


图23-5导入控制台文件

(3)导入策略文件。

单击“操作(A)\所有任务(K)\导入策略(I)...”菜单项来打开“c:\WINDOWS\SysWOW64\IPSecPolicy\stopicmp.ipsec”。此命令执行成功后，在策略名称列表中会出现“禁用ICMP”项。

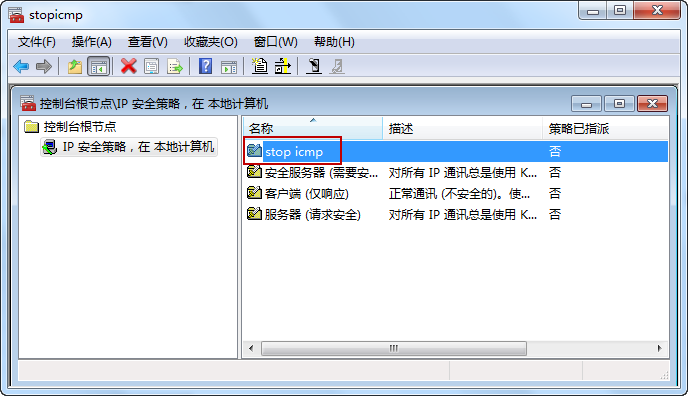


图23-6导入策略文件

(4)启动策略。

用鼠标选中“禁用ICMP”项，单击右键，选择“分配(A)”菜单项。

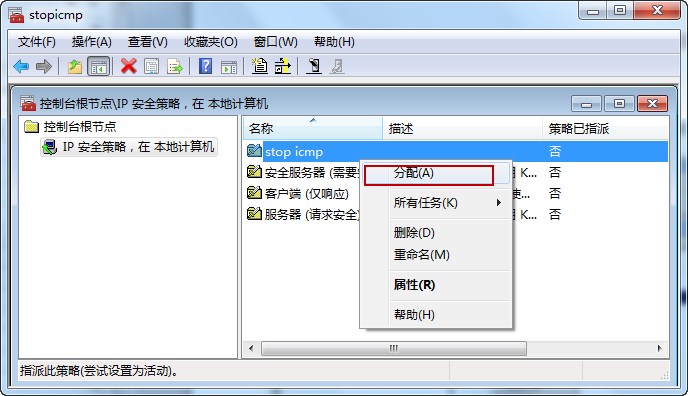


图23-7启动策略

8.主机Dping主机E（172.16.0.2），察看主机A、B、D、E、F捕获到的数据。

●通过此ICMP及其应答报文的MAC地址，绘制其在网络中的传输路径图。

●比较两次ICMP报文的传输路径，简述ICMP重定向报文的作用。

●简述第5步和第6步在信息窃取过程中所起到的作用。

9.主机C取消对ICMP的禁用。

在微软管理控制台（mmc）上，用鼠标选中“禁用ICMP”项，单击右键，选择“未分配(U)”菜单项。

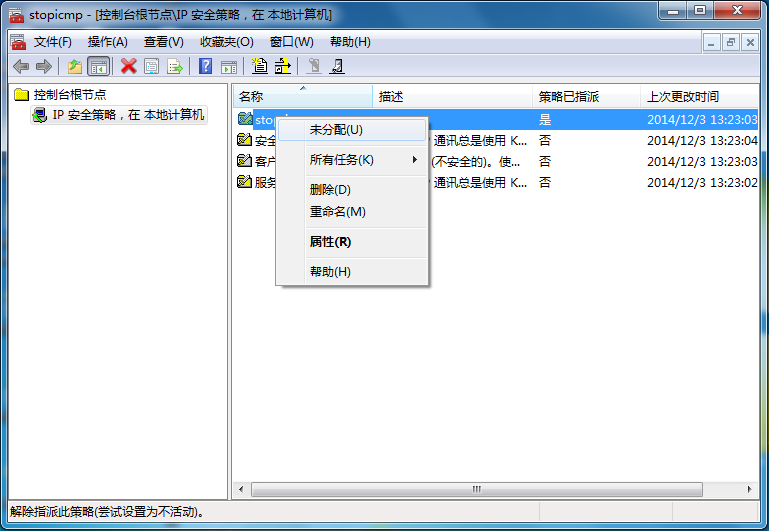


图23-8取消对ICMP的禁用

10.主机B和主机C在命令行下输入“recover\_config”，停止静态路由。

11.主机C输入“route delete 172.16.0.0”删除手工添加的静态路由条目。

思考问题：

通过实验，实现了将主机D发送到主机E的数据经过主机C转发，那么主机C如何操作才能使主机D到主机A数据也经过主机C转发？

## 实验24TCP与UDP端口扫描

【实验目的】

1.了解常用的TCP、UDP端口扫描的原理及其各种手段

2.增强网络安全意识

【学时分配】

4学时

【实验环境】

该实验采用网络结构一

【实验原理】

### 一.TCPconnect扫描

这种方法最简单，直接连到目标端口并完成一个完整的三次握手过程（SYN，SYN/ACK，和ACK）。操作系统提供的“connect()”函数完成系统调用，用来与每一个感兴趣的目标计算机的端口进行连接。如果端口处于侦听状态，那么“connect()”函数就能成功。否则，这个端口是不能用的，即没有提供服务。这个技术的一个最大的优点是不需要任何权限，系统中的任何用户都有权利使用这个调用。另一个好处是速度快。如果对每个目标端口以线性的方式，使用单独的“connect()”函数调用，那么将会花费相当长的时间，你可以通过同时打开多个套接字，从而加速扫描。使用非阻塞I/O允许你设置一个低的时间用尽周期，同时观察多个套接字。但这种方法的缺点是很容易被发觉，并且很容易被过滤掉。目标计算机的日志文件会显示一连串的连接和连接出错的服务消息，目标计算机用户发现后就能很快使它关闭。

### 二.TCPSYN扫描

这种技术也叫“半开放式扫描”（half-openscanning），因为它没有完成一个完整的TCP协议连接。这种方法向目标端口发送一个SYN数据包，如果目标端口返回SYN/ACK标志，那么可以肯定该端口处于侦听状态；否则，返回的是RST/ACK标志。这种方法比第一种更具隐蔽性，可能不会在目标系统中留下扫描痕迹。但这种方法的缺点是实现比较复杂。

### 三.TCPFIN扫描

这种方法向目标端口发送一个FIN数据包。对于关闭的端口，目标系统应该返回一个RST（复位）标志。这种方法通常用在基于Unix的TCP/IP协议堆栈。这种扫描方法的思想是关闭的端口会用RST来回复FIN数据包；相反，打开的端口会忽略对FIN数据包的回复。这种方法和系统的实现有一定的关系，有的系统不管端口是否打开，都回复RST，这样，这种扫描方法就不适用了。并且这种方法在区分Unix系统和WindowsNT系统时，是十分有用的。

### 四.UDP端口扫描

这种方法向目标端口发送一个UDP协议数据包。如果目标端口以“ICMP目的端不可达”消息响应，那么说明该端口是关闭的；反之，如果没有收到“ICMP目的端不可达”响应消息，则可以肯定该端口是打开的。由于UDP协议是面向无连接的协议，这种扫描技术的精确性高度依赖于网络性能和系统资源。另外，如果目标系统采用了大量数据包过滤技术，那么UDP协议扫描过程会变得非常慢。如果你想对Internet进行UDP协议扫描，那么你不能指望得到可靠的结果。

【实验步骤】

练习1TCPConnect扫描

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。

1.主机A、主机B在命令行下输入“netstat-an”命令查看本机打开了那些TCP协议端口。

2.在主机B上启动协议分析器开始捕获数据，并设置过滤条件(提取TCP协议)。

3.在主机A上使用TCP工具，连接主机B的某一端口：

(1)主机A上填入主机B的IP、主机B的某开放端口号，点击“连接”按钮进行连接。

(2)观察提示信息，是否连接上。

(3)主机A点击“断开”按钮断开连接。

(4)主机A使用主机B的某一未开放的端口重复以上实验步骤。

4.察看主机B捕获的数据，分析连接成功与失败的数据包差别。

●结合捕获数据的差别，说明TCPConnect扫描的实现原理。

练习2TCPSYN扫描

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。

1.主机B在命令行中使用“netstat –an –p tcp”命令查看本地开放了那些TCPv4协议端口。

2.在主机B上启动协议分析器开始捕获数据，并设置过滤条件(提取TCP协议)。

3.主机A启动协议编辑器编辑TCP数据包：

MAC层：

目的MAC地址：B的MAC地址。

源MAC地址：A的MAC地址。

IP层：

源IP地址：A的IP地址。

目的IP地址：B的IP地址。

TCP层：

源端口：A的未用端口（大于1024）。

目的端口：B的开放的端口（建议不要选择常用端口）。

标志SYN置为1，其它标志置为0。

计算“长度”和“校验和”字段并填充。

4.发送主机A编辑好的数据包。

5.修改主机A编辑的数据包(将目的端口置为主机B上未开放的TCP端口)，将其发送。

6.察看主机B捕获的数据，找到主机A发送的两个数据包对应的应答包。

●分析两个应答包的不同之处，说明TCPSYN扫描的实现原理。

练习3UDP端口扫描

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。

1.在主机B上使用命令“netstat-an”，显示本机可用的TCP、UDP端口。

注意：通过此命令，会得到一个UDP开放端口列表，同学可以在UDP开放端口列表中任选一个来完成此练习。下面以2425端口为例，说明练习步骤。

2.在主机B上启动协议分析器进行数据捕获并设置过滤条件(提取主机A的IP和主机B的IP)。

3.主机A启动协议编辑器编辑UDP数据包：

MAC层：

目的MAC地址：B的MAC地址。

源MAC地址：A的MAC地址。

IP层：

源IP地址：A的IP地址。

目的IP地址：B的IP地址。

UDP层：

源端口：A的可用端口。

目的端口：B开放的UDP端口2425。

计算“长度”和“校验和”字段并填充。

4.发送主机A编辑好的数据包。

5.修改主机A编辑的数据包(将目的端口置为主机B上未开放的UDP端口)，将其发送。

6.察看主机B捕获的数据。

●主机A发送的数据包，哪个收到目的端口不可达的ICMP数据报。

●说明这种端口扫描的原理。

●使用这种端口扫描得到的结果准确吗？说明理由。

思考问题：

1.根据各种端口扫描的原理，设计出不同端口扫描的预防措施。

## 实验25路由欺骗

【实验目的】

1.了解针对RIP协议的攻击方式及原理

2.理解RIP2的安全属性

3.增强网络安全意识

【学时分配】

2学时

【实验环境】

该实验采用网络结构四（IPv4）

【实验原理】

### 一.路由欺骗

RIP协议是通过周期性（一般情况下为30S）的路由更新报文来维护路由表的，一台运行RIP路由协议的路由器，如果从一个接口上接收到了一个路由更新报文，它就会分析其中包含的路由信息，并与自己的路由表进行比较，如果该路由器认为这些路由信息比自己所掌握的要有效，它便把这些路由信息引入自己的路由表中。

这样如果一个攻击者向一台运行RIP协议的路由器发送了人为构造的带破坏性的路由更新报文，就很容易的把路由器的路由表搞紊乱，从而导致网络中断。

如果运行RIP路由协议的路由器启用了路由更新信息的HMAC验证，则可从很大程度上避免这种攻击，另外RIP第二版增加了在安全方面的功能。

【实验步骤】

练习1利用RIP协议修改路由表

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.在主机A、B、D、E、F上启动协议分析器开始捕获数据，并设置过滤条件（提取RIP和ICMP）。

2.主机B和主机E启动RIP协议:

(1)主机B和主机E分别开启Ipv4虚拟路由器。

(2)主机B和主机E在Ipv4虚拟路由器内使用“zebra -d”和“ripd -d”命令开启RIP协议。

3.等待1分种左右，待主机B和主机E的路由表达到稳定态。使用“telnet localhost 2601”命令连接虚拟路由，密码是“123456”，使用“enable”，输入密码“123456”。“show ip route”察看主机B和主机E的路由表。

4.通过主机Aping主机F（172.16.1.2）。

●通过主机A、B、D、E、F上协议分析器采集到的数据包，记录ping报文的路径。

5.在主机C上启动静态路由。在命令行方式下，输入“staticroute\_config”。

为主机C添加两个静态路由条目在命令行方式下，输入：

“route add 172.16.1.0 mask 255.255.255.0 192.168.0.1 metric 2”；

“route add 172.16.0.0 mask 255.255.255.0 192.168.0.2 metric 2”；

目的是实现数据转发。

6.在主机C上启动协议编辑器，编辑RIP-Request报文。

MAC层：

源MAC地址：主机C的MAC地址

目的MAC地址：主机B的MAC地址（192.168.0.2接口对应的MAC）

IP层：

源IP地址：主机C的IP地址

目的IP地址：广播地址（192.168.0.255）

UDP层：

源端口：520

目的端口：520

RIP层：

命令：1（RIP-Request）

版本：2

路由选择信息选项号：选中该项后点击右侧按钮【B】来追加块。

计算“长度”和“校验和”字段，填充后发送。

7.在主机C上，编辑RIP-Response报文。

MAC层：

源MAC地址：主机C的MAC地址。

目的MAC地址：主机B的MAC地址（192.168.0.2接口对应的MAC）。

IP层：

源IP地址：主机C的IP地址。

目的IP地址：广播地址（224.0.0.9）。

UDP层：

源端口：520。

目的端口：520。

RIP层：

命令：2（RIP-Response）。

版本：2。

路由选择信息选项号：选中该项后点击右侧按钮【B】来追加块。

地址族ID：2。

网络地址：172.16.1.0。

网络掩码：255.255.255.0

下一跳路由器：主机C地址（192.168.0.3）。

度量：1。

计算并填充“长度”和“校验和”，以时间间隔为1秒发送此报文6000次。

8.察看主机B的路由表中路由条目变化。

9.通过主机A来ping主机F（172.16.1.2）。

●通过主机A、B、D、E、F上协议分析器，记录ping报文的路径。

●比较两次ping报文的路径。简述发生欺骗的原理（DV算法）。

10.主机C输入“recover\_config”，停止静态路由服务。输入“route delete 172.16.1.0”和“route delete 172.16.0.0”删除手工添加的静态路由条目。

思考问题：

1.使用RIP协议的“身份验证”功能是否能防止RIP欺骗？你还了解其它的验证机制吗？

## 实验26冲突与网络广播风暴

【实验目的】

1.掌握局域网检测IP地址和主机名冲突的原理

2.了解网络广播风暴的成因及现象

【学时分配】

2学时

【实验环境】

该实验采用网络结构一

【实验原理】

### 一.冲突

IP地址是给每个连接在Internet上的主机分配一个全世界惟一的标识符。当主机IP地址发生冲突时，主机不能进行正常的数据通信；Windows系统在主机启动以及修改IP地址时，通过发送ARP数据包来检测IP地址冲突。

一个网络中的计算机名也是惟一的。Windows系统使用WINS数据包来检测计算机名冲突。

### 二.网络广播风暴

网络广播风暴是指某一时刻网络内充斥广播数据包，从而导致网络性能急剧下降，最终导致网络瘫痪。网络广播风暴一般是由于交换设备或集线器连接成环导致，网卡故障也可能导致网络广播风暴。

【实验步骤】

练习1IP地址冲突

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。

1.主机B启动协议分析器开始捕获数据，并设置过滤条件（提取ARP协议）。

2.主机A将自己的IP地址修改为主机B的IP地址，观察本机现象。

3.主机B停止捕获数据。

4.主机A在命令行下运行“ipconfig/all”命令，察看目前的IP地址。

5.分析捕获到的数据，并回答以下问题：

●结合本练习，简述主机A检测IP地址冲突的过程。

6.将主机A的IP地址恢复。

练习2计算机名冲突

本练习将主机A和B作为一组，主机C和D作为一组，主机E和F作为一组。现仅以主机A、B所在组为例，其它组的操作参考主机A、B所在组的操作。

1.主机B启动协议分析器进行数据捕获。

2.主机A将自己的计算机名修改为主机B的计算机名，观察本机现象。

3.主机B停止捕获数据。分析捕获到的数据，注意查看wins会话，并回答以下问题：

●结合本练习，简述主机A检测计算机名冲突的过程。

4.将主机A的计算机名恢复。

思考问题：

1.发生IP地址冲突，会对正在传输的数据造成影响吗？

练习3网络广播风暴

本练习将主机A、B、C、D、E、F作为一组进行实验。

1.全体同学启动协议分析器并开始捕获数据。

2.选一位同学甲在其所操作的主机的命令行下，使用“arp-d”命令清空arp高速缓存。

3.准备好后，请教师协助将组控设备的两个空闲口用同一根直连线串联起来。

4.同学甲使用ping命令（目的主机是本网段内的主机）来产生一个arp广播报文。

5.全体同学停止捕获数据，分析捕获到的数据，并回答以下问题：

●结合本练习的网络环境和协议分析器捕获的数据，简述网络广播风暴的形成过程。

思考问题：

1.你还能想到其它导致网络广播风暴的环境吗？

## 实验27路由环与网络回路

【实验目的】

1.观察网络中形成路由环时的现象

2.观察网络中形成回路的现象

【学时分配】

2学时

【实验环境】

该实验采用网络结构四(IPv4)

【实验原理】

### 一.路由环

路由器的错误配置及网络连接不合理会导致多个路由器的路由表项连接成环。产生路由环的一种现象是某数据包从一个路由器转发出去后又一次回到该路由器。路由环产生后会增加路由器的负担，严重时会导致网络瘫痪。

### 二.网络回路

网络回路是指连接在网络上的设备形成一个闭合回路的现象。有些回路不会对网络产生影响，但是有些回路会使网络性能下降，严重的会导致网络瘫痪。网络回路通常是由复杂网络接线错误导致。产生网络回路的一种现象是网络上出现大量重复数据包，且其TTL值呈某种规律变化。

【实验步骤】

练习1路由环

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机A、B、C、D、E、F作为一组进行实验。

1. 主机B和主机E开启rip协议。

(1)主机B和主机E分别开启Ipv4虚拟路由器。

(2)主机B和主机E在Ipv4虚拟路由器内使用“zebra -d”和“ripd -d”命令开启RIP协议。

2.在主机B的虚拟路由器中使用命令“gwconfig”将主机B的b2所在的网卡的默认网关设置为192.168.0.1；在主机E的虚拟路由器中使用命令“gwconfig”主机E的e2网卡的默认网关设置为192.168.0.2。

3.主机B、E启动协议分析器开始捕获数据，并设置过滤条件（提取ICMP协议）。

4.主机A发送ping报文（ping实验组内不存在的网段主机，例如：10.0.0.3）。

5.主机B、E停止捕获数据，分析捕获到的数据。

●简述路由环对网络的危害。

●绘制主机A发送的报文在网络中的传输路径。

6.主机B、E分别在虚拟路由器中使用“gwrecover”恢复实验初始环境。

思考问题：

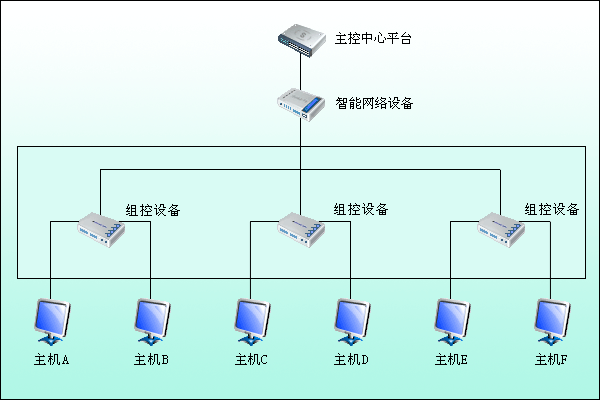
1.在真实的网络环境中，什么时候能形成路由环？

2.在真实的网络环境中，如何避免路由环？

## 附录

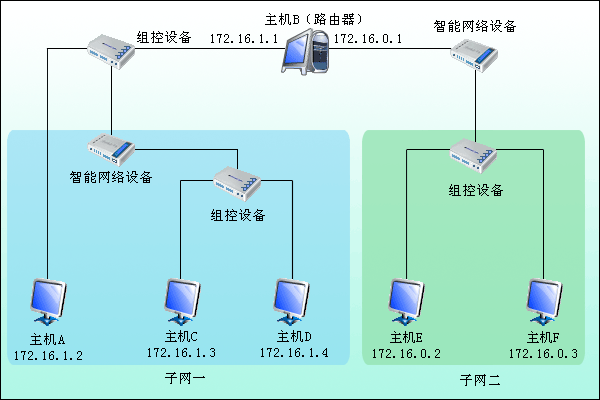
## 附录A网络结构

### 网络结构一



说明：IP地址分配规则为主机使用原有IP，保证所有主机在同一网段内。

### 网络结构二

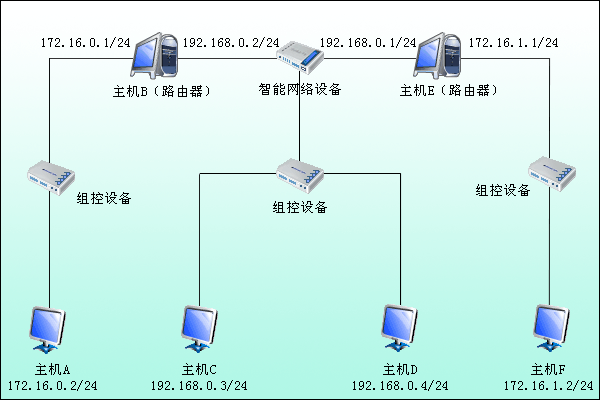


说明：

●主机A、C、D的默认网关是172.16.1.1；主机E、F的默认网关是172.16.0.1。

●主机B为双网卡主机，左端物理接口为b1，右端物理接口为b2。

### 网络结构三(IPv4)



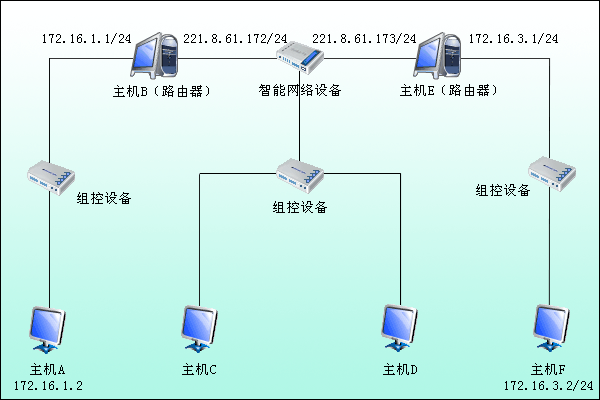
说明：

●主机A的默认网关为172.16.0.1；主机C的默认网关为192.168.0.2；主机D的默认网关为192.168.0.1；主机F的默认网关为172.16.1.1；主机B和主机E不设置默认网关。

●主机B为双网卡主机，左端物理接口为b1，右端物理接口为b2。

●主机E为双网卡主机，左端物理接口为e2，右端物理接口为e1。

### 网络结构三(IPv6)



说明：

●主机B、E做6to4路由器或ISATAP路由器使用。

●主机A的IPv4地址是172.16.1.2，子网掩码是255.255.255.0，默认网关是172.16.1.1；

●主机B的物理接口1（b1）的IPv4地址是172.16.1.1，子网掩码是255.255.255.0；

●主机B的物理接口2（b2）的IPv4地址是221.8.61.172，子网掩码是255.255.255.0；

●主机E的物理接口1（e2）的IPv4地址是221.8.61.173，子网掩码是255.255.255.0；

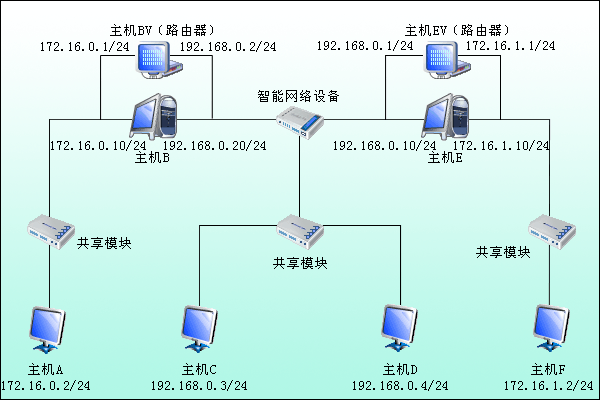
●主机E的物理接口2（e1）的IPv4地址是172.16.3.1，子网掩码是255.255.255.0；

●主机F的IPv4地址是172.16.3.2，子网掩码是255.255.255.0，默认网关是172.16.3.1；

●主机B为双网卡主机，左端物理接口为b1，右端物理接口为b2。

●主机E为双网卡主机，左端物理接口为e2，右端物理接口为e1。

### 网络结构四(IPv4)



1. 智能网络设备说明：

切换智能网络设备至“拓扑结构四”。

2．网络结构说明：

●主机B为双网卡主机，左端物理接口为b1，右端物理接口为b2。

●主机E为双网卡主机，左端物理接口为e2，右端物理接口为e1。

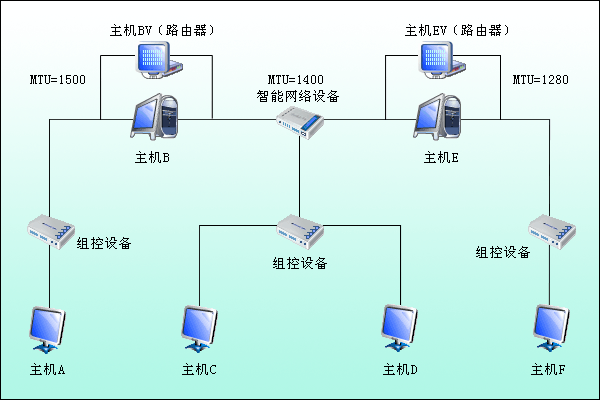
●网络结构划分为3个网段。

●主机BV、EV做路由器使用。它们分别是主机B、主机E虚拟机下的Linux操作系统。

● 主机BV左端为物理接口1，右端为物理接口2。

● 主机EV左端为物理接口1，右端为物理接口2。

### 网络结构四(IPv6)



1. 智能网络设备说明：

切换智能网络设备至“拓扑结构四”。

2．网络结构说明：

●主机B为双网卡主机，左端物理接口为b1，右端物理接口为b2。

●主机E为双网卡主机，左端物理接口为e2，右端物理接口为e1。

●网络结构划分为3个网段：

网段1：站点本地子网ID为FEC0:0:0:1::/64

网段2：站点本地子网ID为FEC0:0:0:2::/64

网段3：站点本地子网ID为FEC0:0:0:3::/64

●主机BV、EV做路由器使用。它们分别是主机B、主机E虚拟机下的Linux操作系统。

● 主机BV左端为物理接口1，右端为物理接口2。

● 主机EV左端为物理接口1，右端为物理接口2。

●主机A的IPv6地址的默认网关是主机BV接口1的链路本地地址；主机C的IPv6地址的默认网关是主机BV接口2的链路本地地址；主机D的IPv6地址的默认网关是主机BV接口2的链路本地地址；主机F的IPv6地址的默认网关是主机EV接口2的链路本地地址。

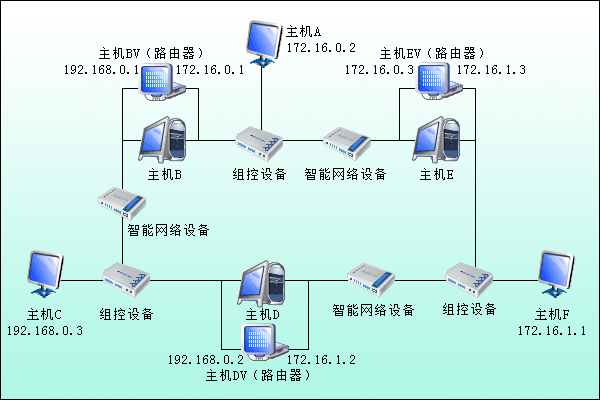
3．检测网络结构三连接的正确性：

●按照网络结构图连接网络，并使用拓扑验证检查连接的正确性。

●在主机A上，执行ping命令（pingfec0:0:0:1::1），若ping不通，则交换主机BV的两个网卡的网线连接。

●在主机F上，执行ping命令（pingfec0:0:0:3::1），若ping不通，则交换主机EV的两个网卡的网线连接。

### 网络结构五



说明：

●主机A的默认网关为172.16.0.1；主机C的默认网关为192.168.0.2；主机F的默认网关为172.16.1.3；主机B、D、E不设置默认网关。

●主机B为双网卡主机，左端物理接口为b1，右端物理接口为b2。

●主机D为双网卡主机，左端物理接口为d1，右端物理接口为d2。

●主机E为双网卡主机，左端物理接口为e2，右端物理接口为e1。

●主机BV、DV、EV做路由器使用。它们分别是主机B、主机D、主机E虚拟机下的Linux操作系统。

## 附录B OutlookExpress的使用方法

用户名和密码的分配规则：groupx\_y，其中x表示组别索引，y表示角色索引。邮件服务器的地址为：JServer.Netlab（172.16.0.253）。例如：实验组1的主机A使用group1\_1作为用户名和密码，他的邮箱是group1\_1@JServer.Netlab，实验组2的主机C使用group2\_3作为用户名和密码，其邮箱地址是group2\_3@JServer.Netlab，依此类推。

下面以用户名group1\_1和group1\_2为例来说明Outlook的配置以及电子邮件的发送与接收。

#### 一．OutlookExpress配置

首次打开OutlookExpress会弹出“Internet连接向导”，如果不是首次打开，还可以通过菜单中的命令--“工具”->“帐户”->“添加”->“邮件”打开“Internet连接向导”。按照向导来配置Outlook邮件帐户，具体步骤如下：

(1)打开OutlookExpress，在“显示名”中填写“group1\_1”，单击“下一步”按钮；

(2)在“电子邮件地址”中填写“group1\_1@JServer.NetLab”，单击“下一步”按钮；

(3)在“接收邮件服务器”与“发送邮件服务器”均填写“172.16.0.253”，单击“下一步”按钮。

(4)在“帐户名”与“密码”均填写“group1\_1”，单击“下一步”直到完成。

#### ****二．发送邮件****

(1)单击“创建邮件”工具栏按钮，弹出“新邮件”窗口；

(2)在“收件人”中填写“group1\_2@JServer.NetLab”,主题自选；

(3)如果要附带发送文件，单击“插入->文件附件”菜单项，选中目标文件夹下的目标文件，单击“发送”按扭，完成发送过程。

#### ****三．接收邮件****

单击“发送/接收”按钮接收即可接收发过来的邮件，右链点击附件，选择“另存为”即可将对方发送过来的文件保存到目标文件夹下。

## 附录C Windows Live Mail的使用方法

用户名和密码的分配规则：groupx\_y，其中x表示组别索引，y表示角色索引。邮件服务器的地址为：JServer.Netlab（172.16.0.253）。例如：实验组1的主机A使用group1\_1作为用户名和密码，他的邮箱是group1\_1@JServer.Netlab，实验组2的主机C使用group2\_3作为用户名和密码，其邮箱地址是group2\_3@JServer.Netlab，依此类推。

下面以用户名group1\_1和group1\_2为例来说明Outlook的配置以及电子邮件的发送与接收。

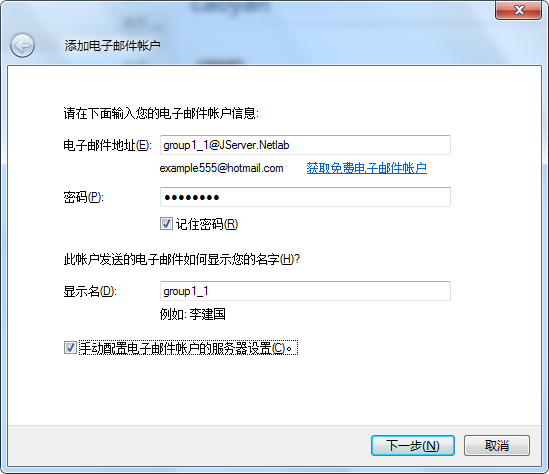
#### 一．Windows Live Mail配置

首次打开Windows Live Mail会弹出“添加电子邮件账户”向导（如果不是首次打开，可在发件箱中添加电子邮件账户），按照向导来配置邮件帐户，具体步骤如下：

（1）编辑电子邮件账户信息，具体如下：

* 电子邮件地址：group1\_1
* 密码：group1\_1
* 显示名：group1\_1

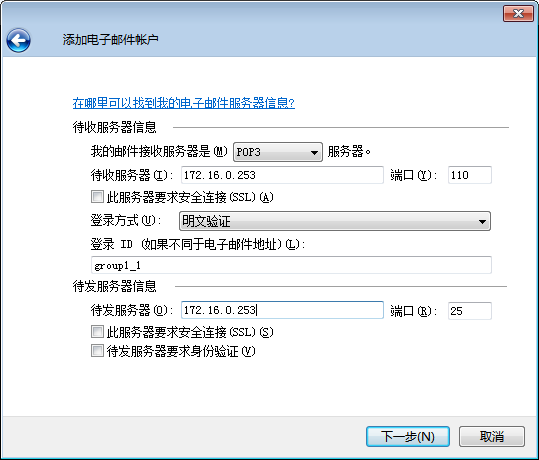
勾选“手动配置电子邮件账户的服务器设置”，下一步。



（2）编辑邮件服务器信息，具体如下：

* （待收服务器信息）待收服务器：172.16.0.253
* （待发服务器信息）待发服务器：172.16.0.253

其他信息保持不变，单击“下一步”直至完成。



#### 二．发送邮件

菜单栏“新建 | 电子邮件”（Ctrl + N），弹出“新邮件”窗口，编辑邮件信息，具体如下：

* 收件人：group1\_2@JServer.Netlab
* 主题：自定义
* 附件：如果要附带发送文件，单击工具栏“附件”，选中目标文件，

单击“发送”按扭，完成发送过程。

#### 三．接收邮件

（group1\_2）点击“同步”，查看“收件箱”。

## 附录D Win7下SNMP服务的安装

Win7下SNMP服务器的安装步骤如下：

按Windows窗口键+R调出运行命令对话框，在对话框中输入“appwiz.cpl”按确定调出“卸载或更改程序”对话框，选中“打开或关闭windows功能”将弹出Windows功能，如下图：

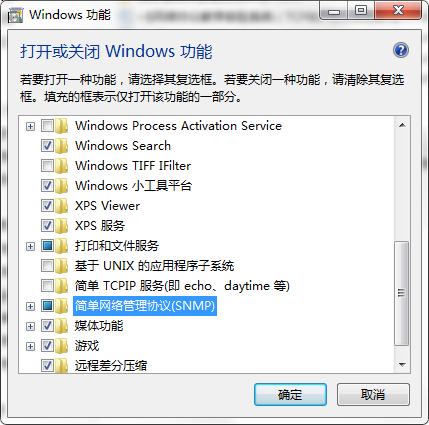


图3-1 Windows功能

选中“简单网络管理协议（SNMP）”条目前的复选框。按“确定”，等待几秒钟，系统会自动完成SNMP服务的安装。

## 附录E协议编辑器使用说明

协议编辑器简介

协议编辑器主界面如下：

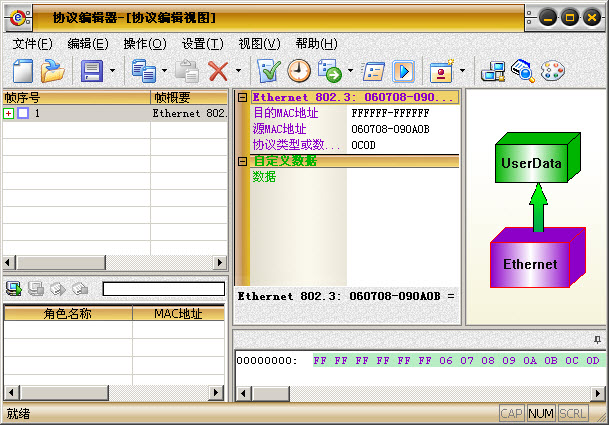


图4-1协议编辑器主界面

本系统的主界面显示分为五个部分：多帧编辑区、单帧编辑区、协议模型图、地址本和十六进制显示区。

1.多帧编辑区

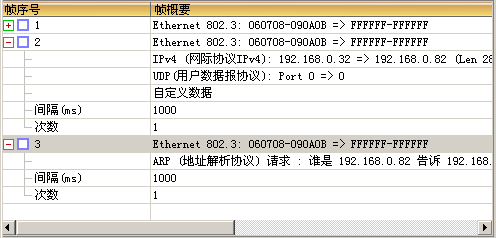


图4-2多帧编辑区

(1)第一列为帧序号，此序号无其它特殊含义，即为各帧顺序向下的计数号。点击“+”可展开为多行，相邻两帧发送的时间间隔默认值为1000ms，如果点击展开的为第一帧，则为发送延迟时间。发送次数默认为1次。

(2)第二列为帧的概要信息：

MAC源地址、目的地址、协议类型（在协议类型下拉框中进行选择）。

LLCLLCUnnumbered/LLCInformation/LLCSupervisor，DSAP，Ctrl。

ARP请求：标明请求的源IP和目的IP。

ARP应答：标明应答主机IP和MAC地址的对应情况。

IP源IP、目的IP、IP总长度、协议类型（在协议类型下拉框中进行选择），（如果分片或偏移量≠0）Flagment偏移量：偏移量+（总长度-首部长度-1）。

TCP源端口、目的端口、TCP类型（SYN，FIN，ACK，RST）、序号、ACK确认序号和窗口大小。

UDP源端口、目的端口。

2.单帧编辑区

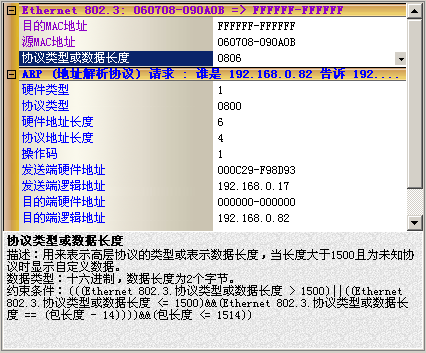


图4-3单帧编辑区

单帧编辑区分为帧编辑区和提示区两部分，其中帧编辑区可以对协议属性进行编辑，提示区可以根据选中的协议属性给出相应的提示，便于学生了解各属性的含义或范围。

3.协议模型图

在协议模型图中，以图形化模型显示该协议的封装层次，并与单帧编辑区对应的协议层相互联动，当前被选中的协议层用红色边框显示，如图所示：

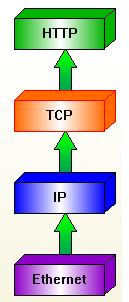


图4-4协议模型

4.地址本

地址本包括两项功能：主机扫描和端口扫描。点击[主机扫描]按钮，系统在当前网络内进行主机扫描，并在地址本中列出扫描到的所有主机。选中一台主机后，可以点击[端口扫描]按钮，对该主机进行TCP端口扫描。地址本的主机扫描结果分为三列：角色名称、MAC地址和IP地址。每台主机的端口扫描结果也分为三列：端口号、端口类型和服务名称。如图所示：



图4-5地址本

5.十六进制显示区

十六进制显示区与单帧编辑区是联动的，选中相应的十六进制数据，右键可进行计算校验和、复制字符串、复制16进制数据、计算长度操作。十六进制显示区也可伸缩显示，不需要观看时可单击[隐藏]按钮将其隐藏到下面。如图所示：

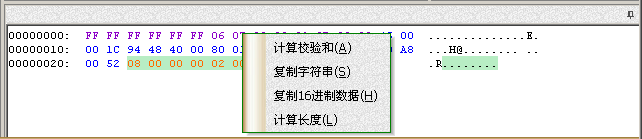


图4-6十六进制显示区

6.协议编辑器菜单栏

(1)文件菜单

说明: http://172.16.0.13/etmis/resource/images/1284882836893_pic.bmp

图4-7文件菜单

新建文件：新建立一个帧序列文件。

打开文件：打开原有的文件。

保存文件：保存当前正在编辑的帧序列文件。

保存选中帧：保存被选中的帧。

退出：退出协议编辑器。

(2)编辑菜单

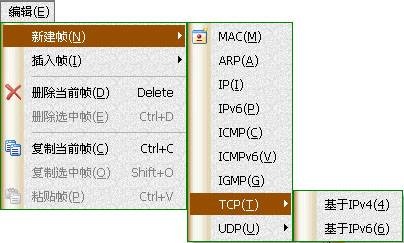


图4-8编辑菜单

新建帧：在帧序列的末尾新建一个帧。

插入帧：在当前帧之前插入一个帧。

删除当前帧：删除当前正在编辑的帧。

删除选中帧：删除所有选中的帧。

复制当前帧：将当前帧复制到剪贴板中。

复制选中帧：将所有被选中的帧复制到剪贴板中。

粘贴帧：将剪贴板中的帧粘贴到当前位置。

(3)操作菜单



图4-9操作菜单

全部选中：选中当前所有帧。

反向选择：对当前被选中的帧进行反向选择。

取消选择：取消当前选中的帧。

校验帧：可随时检验编辑的协议帧。

设置时间间隔：对全部帧或选中帧设置发送时间间隔。

设置触发条件：设置数据包触发条件约束。

启动触发：启动触发功能。

发送全部帧：对多帧编辑区中的所有帧进行发送。

发送选中帧：对多帧编辑区中被选中的帧进行发送。

定制发送：自定义发送。在确认是否修改IP标识号、是否修改源IP地址，并重新计算修改校验和后，设置发送时间间隔和发送次数，定制发送当前数据帧。

取消发送：当系统正在发送数据帧时，可以停止发送帧。

(4)设置菜单



图4-10设置菜单

选择适配器：当主机有两个以上适配器时，选择用来发送数据帧的适配器。

设置端口扫描：对要进行扫描的主机端口进行添加、删除、修改、复位等设置，并可以将自定义的端口信息进行导出和导入。

设置协议颜色：可以自定义不同协议的显示颜色。

加载协议脚本：可以加载自定义协议脚本。

恢复默认协议脚本：可以恢复默认的协议脚本。

系统语言：软件系统语言切换，其中分为中文显示和英文显示，选择一种后重启软件系统进行语言切换。

界面风格：软件界面风格切换，其中包括黄色系和蓝色系，选择一种后进行切换。

(5)视图菜单：

说明: http://172.16.0.13/etmis/resource/images/1313560453928_pic.jpg

图4-11视图菜单

协议编辑视图：进入数据帧协议编辑的主界面。

协议生成视图：进入自定义协议编辑界面。

(6)帮助



图4-12帮助菜单

帮助主题：显示所有帮助主题。

协议帮助：显示多种协议帮助。

关于：显示版本信息及版权信息。

2.应用实例

下面通过一个例子来进一步说明本系统的功能与使用方法。

编辑MAC层：

(1)源MAC地址和目的MAC地址可以在地址本中进行查找，双击后自动添加，这样可以保证填写的MAC地址在网络中真实存在，保证了模拟的真实性。

(2)在协议类型或数据长度字段中选择IP协议。学生可以根据下拉列表框中的项目选择此帧的上层协议，也可以直接填写，系统会根据填写的内容进行字符匹配判断，自动选择上层协议。

(3)编辑MAC层的界面如下图所示：

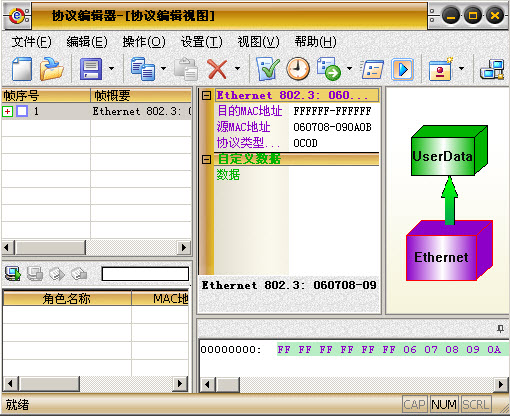


图4-13编辑MAC层的界面

编辑IP层：

(1)学生选择IP协议后，单帧编辑区和协议模型图会相应地显示出IP层，如下图所示：

(2)当前IP层为选中状态，协议模型图中以红色线条显示。

(3)单帧编辑区显示IP层需要填写的各种属性。

(4)十六进制显示区显示学生填写的属性的十六进制值。

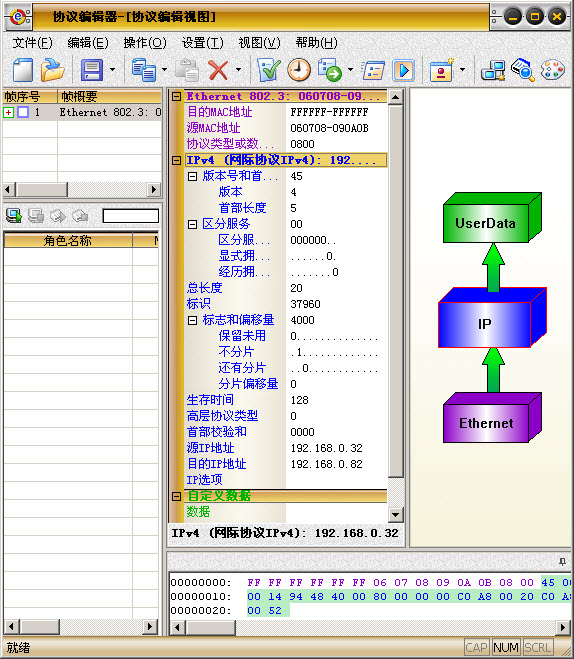


图4-14编辑IP层的界面

说明：IP层涉及到“校验和”的计算，系统提供了自动计算、手动计算和十六进制显示区右键计算三种方式：

●自动方式可以直接得出结果，自动填入。

●手动方式先将IP协议中首部校验和字段置0，根据点击［M］弹出的“手动计算校验和”对话框提示进行计算，并手动将结果填到首部校验和字段。

●十六进制显示区右键计算的具体计算方法如下：在计算“校验和”前将IP协议中首部校验和字段置0。将涉及到校验和计算的十六进制代码全部选中，点击鼠标右键，在弹出的对话框中选择“计算校验和”，并手动将结果填到首部校验和字段。

编辑TCP层：

(1)与前两层相同，单帧编辑区和协议模型图进行相应的刷新后，学生就可以填写其相应的属性值了。

(2)在每一层，学生均可以填写“自定义数据”这一选项，但是一旦填写了这一项，就意味着这个单帧已经编辑完毕，并且按照其填写的数据发送出去。

(3)TCP的上层协议是通过端口来判断的。填写不同的端口，上层协议类型会不同。

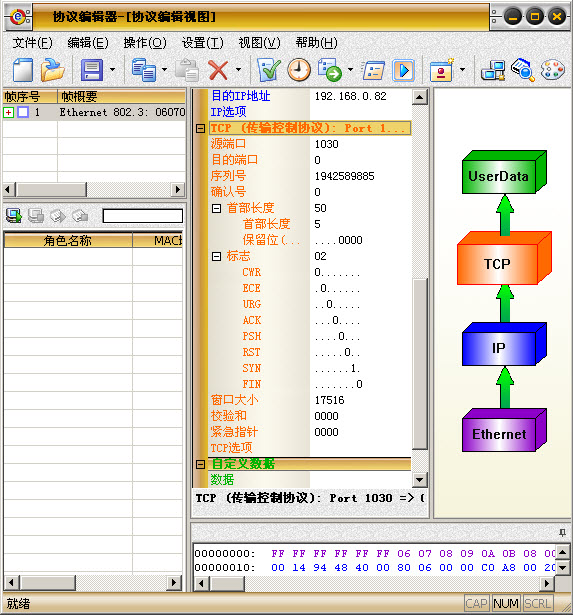


图4-15编辑TCP层

## 附录F协议分析器使用说明

协议分析器简介

协议分析器负责捕获网络上的各种数据帧，分析其中包含的各层协议，提供辅助教学功能。主要有两个功能模块：会话分析和协议解析。

进入协议分析器，单击[开始捕获]按钮进行数据捕获。刷新显示后，在会话分析和协议解析视图显示对数据的分析。

1.会话分析

会话分析功能将捕获到的常用协议的数据帧加入会话列表，并且有会话次序和数据传输方向的图示，使学生能够直观的看出一次完整的会话过程。

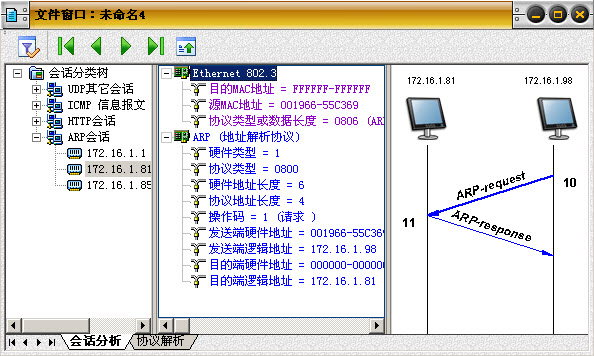


图5-1会话分析界面

会话分析功能主要用于有完整会话过程的协议，例如：ARP、ICMP、HTTP、DNS、SMTP、POP3等。会话列表包含有通信双方的IP地址、端口号、会话类别。通过一次完整会话的分析，使学生加深对协议原理的理解。

2.协议解析

协议解析界面显示如图所示：



图5-2协议解析界面

该窗口主要有三个显示区：概要解析列表、详细解析树和十六进制数据显示区，十六进制数据显示区包括十六进制数据显示和字符显示两种状态。

学生可以从详细解析树中获得非常详细的解释和说明，并且不同栏目的数据和解释可以按照学生的选择实现动态跟踪显示，方便学生对数据的分析。

定义过滤条件，可保存成过滤器配置。当应用程序重新启动时，不加载上一次保存的过滤器配置，只能使用默认的过滤器。过滤器可以实现网络地址、数据模式和协议过滤三种过滤方式。

●网络地址过滤

网络地址过滤中可以使用IP地址、MAC地址。

填写网络地址，中间的编辑框表示数据方向，在这里可以选择需要过滤地址数据的传输方向，“站点1”中填写传输的一端地址，“站点2”中填写传输的另一端地址，如果不填写“站点2”，系统则缺省设为“Any”，表示过滤站点1到任意地址的数据。

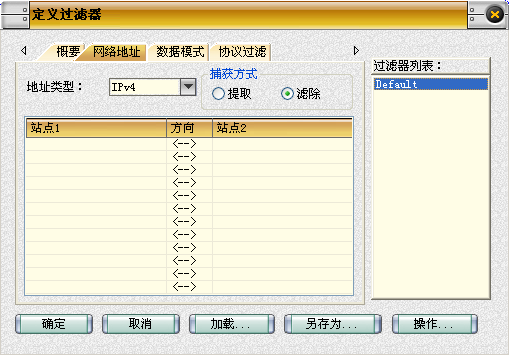


图5-3网络地址过滤设置

●数据模式过滤

数据模式过滤，是针对十六进制数据而言的。起始位置指的是十六进制数据中的第几个字节，数据长度是指从起始位置算起共有多少个字节。下面的文本框中要输入过滤的十六进制内容，长度要与填入的“数据长度”一致。点击［确定］按钮之后，加入的过滤条件会生效，在数据捕获时会将符合条件的内容过滤出来。

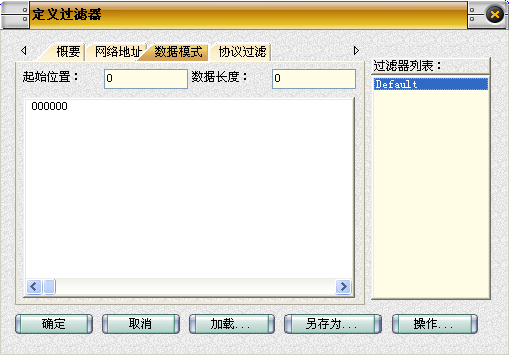


图5-4数据模式过滤

●协议过滤模式

协议过滤可以针对具体某个协议进行设置，如果针对该协议有封装类型复选框，则需要进行选择，如不做选择，过滤设置无效。

按钮主要功能：

操作：可以新建、删除、重命名过滤器。

加载：调入以前保存的扩展名为“.flt”的过滤器文件。

另存为：将新设置的过滤器存储为扩展名为“.flt”的文件。



图5-5协议过滤设置

在过滤器[操作]中点击[新建]按钮，在“过滤器名称”中输入过滤条件名称，然后进行过滤条件设置。也可以通过［删除］、［重命名］按钮对学生自定义的过滤条件进行删除、重命名等操作。但不能对“缺省过滤条件”（即“默认”）进行这样的操作。

在定义过滤器窗口中，单击［另存为］按钮可保存该过滤条件，通过［加载］按钮可打开保存过的过滤条件，方便学生的使用。

## 附录G工具使用说明

工具使用说明

(1)TCP工具：分为客户端和服务器两种类型。如图所示：



图6-1TCP工具-服务器端

选择“客户端”时，填写正确信息，客户端连接到服务器，可以发送、接收数据。

(2)UDP工具：分为客户端和服务器两种类型。如图所示：

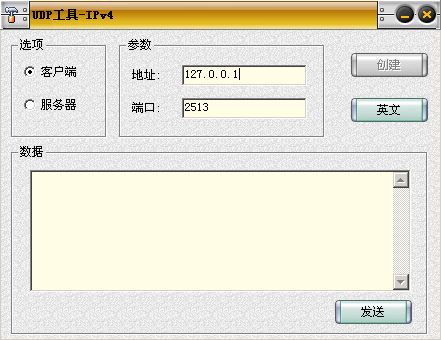


图6-2UDP工具-客户端

客户端和服务器端进行连接通讯，选择服务器端后，服务器可接收多个客户端发送的信息。

(3)组播工具：打开组播工具对话框。如图所示：

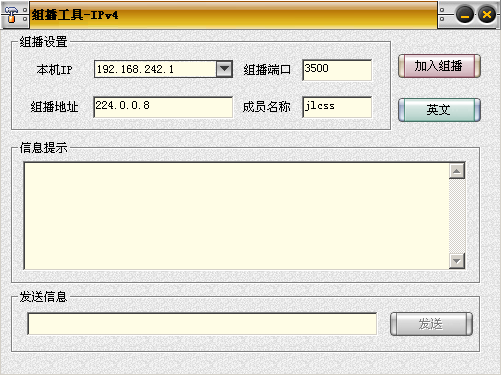


图6-3组播工具

选择本机要加入组的IP地址填写要连接的组播地址和端口号加入到组内进行发送和接收信息，可多台主机加入同一组播地址。

(4)地址本工具：打开地址本工具对话框。如图所示：



图6-4地址本工具界面

打开地址本对话框，单击[主机扫描]按钮，系统会自动进行扫描，并列出扫描到的所有主机。地址本分为三列：角色名称，MAC地址、IP地址。右击主机、地址信息可以复制角色名称、MAC地址、IP地址。刷新可以获取当前网络上的地址信息。

(5)SNMP连接工具：点击弹出对话框，对启动SNMP代理主机获取相应信息，左侧为SNMP扫描与MIB树切换区，右侧显示值区，扫描启动SNMP代理主机并选择填写某个主机的IP地址，点击按钮或双击获取MIB树上节点值，可对节点值进行设置。分为两个界面显示如图：



图6-5SNMP工具获取界面

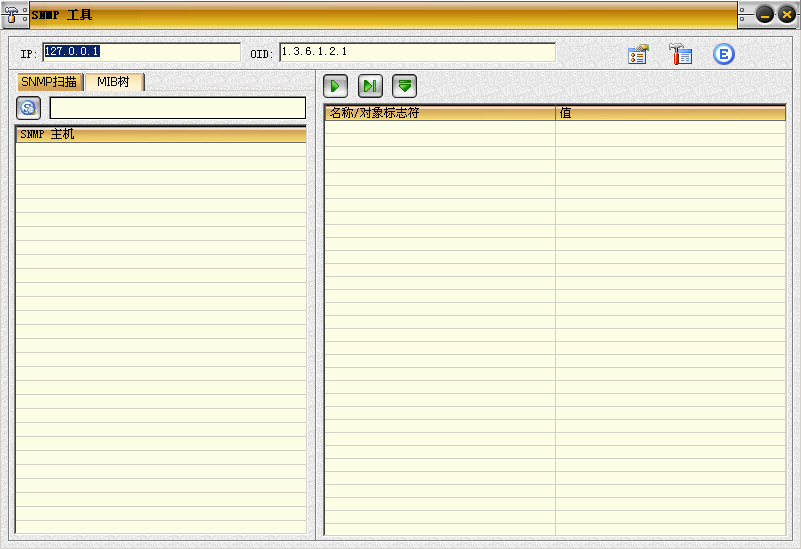


图6-6SNMP工具设置界面

IP地址：需要获取SNMP信息的机器地址

OID：类识别代码（OID，ObjectIdentifier）

获取：点击显示获取节点值界面，其中包括以下功能：

●获取：发送get-request请求，获取SNMP值

●获取下一个：发送get-next-request请求，获取SNMP信息

●获取子树：连续发送get-next-request请求，直到该子树遍历完成

获取信息显示区：视图右侧为获取管理信息的显示区域，包括NAME/OID、VALUE。

设置：点击显示设置节点值界面，选择节点确定绑定类型进行设置节点值。

中英文切换按钮：点击后整个工具界面切换为中文或英文。

右侧显示区：

SNMP主机扫描：显示扫到启动SNMP代理服务器的主机。

MIB树：包含代理服务器信息，可双击或点击工具条上按钮获取相应节点信息。下面有树节点对应中文解释。

【注意】SNMP主机扫描和MIB树以属性页形式切换。

(6)拓扑验证工具：打开拓扑验证工具对话框。如图所示：

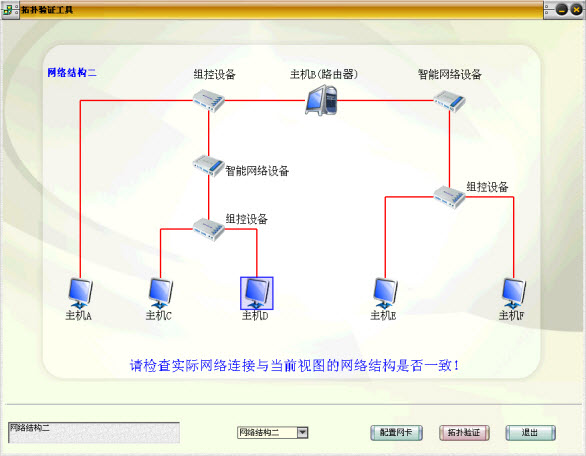


图6-8拓扑验证工具

在下拉列表框中选择实验所需要的拓扑结构，单击“配置网卡”在打开的配置网卡信息对话框中选择所需要的适配器，单击配置。配置成功后，单击“拓扑验证”完成网络结构的拓扑验证。

## 附录H 常见问题

问题1： 在做实验八的SNMP实验时，需要启动操作系统的SNMP服务，但主机的服务中却没有？

解决办法：SNMP服务不是系统的默认安装选项，需要手动添加。具体操作为：点击“开始菜单/设置/控制面板”中的“添加/删除程序”，选择 “添加/删除windows组件”，从滚动列表中选择“管理和监视工具”，点击 [详细信息] 按钮，从子组件列表中选择“简单网络管理协议”，点击 [确定]和[下一步]按钮进行安装。

问题2： 在通过HTTP连接管理信息系统的时候，页面无法正常显示。

解决办法：请检查网络结构及主控中心平台是否运行正常，同时正常访问管理信息系统要将浏览器升级到IE6.0或以上版本，升级IE后显示正常。

印刷信息：

版次：2015年01月第一版

时间：2015年01月第一次印刷

Version1.1