计算机网络课程设计

实验报告

专业班级 210710

姓名 高立扬

学号 21071003

组长姓名学号 高立扬 21071003

组长联系方式 18801336823

同组人姓名学号 李博文 21071015

高立扬 21071003

李 昂 21071014

实验日期 2024/7/16

2024年 7月 22日

|  |
| --- |
| **一、实验内容和要求**  1、实验一  两台路由器通过Serial口串联起来；配置每个路由器Ethernet口和Serial口的IP地址  两台路由器的Serial口之间采用PPP帧格式封装  每台路由器可配置静态路由，使得各子网能相互通信  每台PC通过交换机连接到路由器Ethernet口，配置每台PC的IP地址，使得各PC能相互通信  测试网络连通性  2、实验二  本实验目的是通过使用协议分析软件，对TCP/IP协议族中协议通信过程进行监控和分析，以了解TCP/IP各种协议的工作过程。  本实验内容是利用协议分析软件跟踪局域网数据包，将安装协议分析软件的PC接⼊以太网中，跟踪PC之间的数据包，并存⼊文件以备重新查；设置过滤器过滤网络数据包以检测特定数据流。利用协议分析软件的统计工具显示数据包的各种统计信息。  3、实验三  本实验目的是使学生掌握网络层协议的原理及实现方法。  本实验为ICMP实验。实验内容：Ping命令实现的扩充，在给定的Ping程序的基础上做如下功能扩充：  -h 显示帮助信息  -b 允许ping一个广播地址，只用于IPv4  -t 设置ttl值，只用于IPv4  -q 安静模式。不显示每个收到的包的分析结果，只在结束时，显示汇总结果 |
| 二、实验环境  1、实验一  操作系统：Windows XP  软件：超级终端  2、实验二  操作系统：Windows 10  软件：WireShark  3、实验三  操作系统：Ubuntu 18.04.6 LTS |

|  |
| --- |
| 三、程序的需求分析与逻辑框图  1、实验一  2.1需求分析：  网络拓扑结构：两台路由器通过Serial口串联。每台路由器的Ethernet口连接到交换机。交换机连接多台PC。  IP地址配置：为每台路由器的Ethernet口和Serial口配置IP地址。为每台PC配置IP地址。  协议与封装：两台路由器之间的Serial口使用PPP帧格式封装。  路由配置：每台路由器配置静态路由，确保各子网间能相互通信。  网络连通性测试：通过ping命令测试各PC之间的网络连通性。    2、实验二  在PC中安装Wireshark。具体安装过程详见Wireshark用户指南。  启动Wireshark协议分析软件，选择抓包菜单，启动实时监视器，开始实时跟踪显示网络数据包。可根据系统提示修改显示方式，详见Wireshark用户指南。  调出跟踪存储的历史报文，选择有代表性的以太网、IP、ICMP、ARP、TCP、UDP、DNS等数据包，对照有关协议逐个分析报文各字段的含义及内容。  3、实验三  2.1需求分析：需要完成一个Ping程序，实现Ping命令实现的扩充，在给定的Ping程序的基础上做功能扩充。在完成实验之前，首先需要熟悉Ping命令的基本描述：  Ping的操作是向某些IP地址发送一个ICMP Echo消息，接着该节点返回一个ICMP Echo replay消息。  ICMP消息使用IP头作为基本控制。IP头的格式如下  0 1 2 3  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |Version| IHL |Type of Service| Total Length |  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  | Identification |Flags| Fragment Offset |  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  | Time to Live | Protocol | Header Checksum |  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  | Source Address |  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  | Destination Address |  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+    Version=4  IHL Internet头长  Type of Service = 0  Total Length IP包的总长度  Identification, Flags, Fragment Offset 用于IP包分段  Time to Live IP包的存活时长  Protocol ICMP = 1  Addresses 发送Echo消息的源地址是发送Echo reply消息的目的地址,相反,发送Echo 消息的目的地址是发送Echo reply消息的源地址。  Ping实际上是使用ICMP中的ECHO报文来实现的。Echo 或 Echo Reply 消息格式如下:  0 1 2 3  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  | Type | Code | Checksum |  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  | Identifier | Sequence Number |  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  | Data ...  +-+-+-+-+-  Type  echo消息的类型为8  echo reply 的消息类型为0。  Code=0  Checksum  为从TYPE开始到IP包结束的校验和  Identifier  如果 code = 0, identifier用来匹配echo和echo reply消息  Sequence Number  如果 code = 0, identifier用来匹配echo和echo reply消息  功能描述:  收到echo 消息必须回应 echo reply 消息。  identifier 和 sequence number 可能被发送echo的主机用来匹配返回的  echo reply消息。例如: identifier 可能用于类似于TCP或UDP的 port  用来标示一个会话, 而sequence number 会在每次发送echo请求后递增。  收到echo的主机或路由器返回同一个值与之匹配  1、数据结构的描述  1) IP包格式  struct ip {  BYTE Ver\_ihl; //版本号与包头长度  BYTE TOS; //服务类型  WORD Leng; //IP包长度  WORD Id; //IP包标示,用于辅助IP包的拆装,本实验不用,置零  WORD Flg\_offset; //偏移量,也是用于IP包的拆装,本实验不用,置零  BYTE TTL; //IP包的存活时间  BYTE Protocol; //上一层协议,本实验置ICMP  WORD Checksum; //包头校验和,最初置零,等所有包头都填写正确后,计算并替换。  BYTE Saddr[4]; //源端IP地址  BYTE Daddr[4]; //目的端IP地址  BYTE Data[1]; //IP包数据  };  2）ICMP包格式  struct icmp {  BYTE Type; //ICMP类型,本实验用 8: ECHO 0:ECHO REPLY  BYTE Code; //本实验置零  WORD Checksum; //ICMP包校验和,从TYPE开始,直到最后一位用户数据,如果为  字节数为奇数则补充一位  WORD ID; //用于匹配ECHO和ECHO REPLY包  WORD Seq; //用于标记ECHO报文顺序  BYTE Data[1]; //用户数据  };  2.2逻辑框图：  ·总体设计  程序分为两大部分：一部分读取收到的所有消息，并输出ICMP Echo replay消息，另一部分每个一秒钟发送一个Echo消息。另一部分由SIGALARM信号每秒驱动一次。  ·详细设计  main函数：①设置发送的ICMP回射请求报文的数据长度datalen②处理命令行参数从而确定Ping命令的选项③根据选项调整相关参数④为了持续发送请求报文，需要SIGALARM信号每秒定时发送报文⑤获取目的地址需要域名解析⑥调用readloop接收ICMP响应报文并输出分析结果。    readloop函数：①创建一个套接口②设置接收数据缓存区大小③设置与套接口关联的选项④发送定时信号发送报文并调用proc\_v4函数读取响应报文。    proc函数：①获取ICMP头部指针，检查报文长度并计算往返时间，并输出。    send：①构造ICMP回射请求报文，计算校验和，发送回射请求报文。 |

|  |
| --- |
| 四、程序核心功能的实现机制  1、实验一  物理连接：使用一根串行线将两台路由器的Serial端口连接起来。同时用Console线连接两台PC与对应的路由器。同时，将网线连接到交换机上。  IP地址配置：为两台路由器的Ethernet端口配置合适的IP地址，这些地址应该位于不同的子网中，以便区分不同网络段。为每台PC配置与连接到的路由器Ethernet端口相同子网的IP地址。  静态路由配置：利用路由器命令在每台路由器上配置静态路由，指明到达其他子网的路径。这通常涉及到指定目的网络地址和下一跳地址  用户模式  Router> // 用户模式  Router>sh interface // 查看接口信息  Router>sh ip route // 查看路由表信息  特权模式  Router>enable // 进入特权模式  Router# // 特权模式  Router#sh ip route // 显示路由表信息  Router#sh interface // 显示接口信息  Router#disable // 退出特权模式  全局配置模式  Rouer#config // 进入全局配置模式  Router\_config# // 全局配置模式  Router\_config#ip route destination subnet-mask next-hop // 配置静态路由  接口配置模式  Router\_config#interface f0/0 // 进入接口配置模式  Router\_config\_f0/0# // 接口配置子模式  Router\_config\_f0/0#ip address address subnet-mask  // 陪置IP和子网掩码  Router\_config\_f0/0#no shutdown // 激活接口  Router\_config\_f0/0#exit // 退回上一级  路由协议配置模式  Router\_config#router rip // 进入RIP动态路由  Router\_config\_router# // 路由协议配置模式  2、实验二  Wireshark使用WinPCAP作为[接口](https://www.hqchip.com/app/1787" \t "https://www.elecfans.com/d/_blank)，直接与网卡进行数据报文交换。  Wireshark使用的环境大致分为两种，一种是[电脑](https://m.elecfans.com/v/tag/1247/" \t "https://www.elecfans.com/d/_blank)直连网络的单机环境，另外一种就是应用比较多的网络环境，即连接[交换机](https://m.elecfans.com/v/tag/1392/" \t "https://www.elecfans.com/d/_blank)的情况。  「单机情况」下，Wireshark直接抓取本机网卡的网络流量；  「交换机情况」下，Wireshark通过[端口](https://www.hqchip.com/app/1808" \t "https://www.elecfans.com/d/_blank)镜像、ARP欺骗等方式获取局域网中的网络流量。  端口镜像：利用交换机的接口，将局域网的网络流量转发到指定电脑的网卡上。  ARP欺骗：交换机根据MAC地址转发数据，伪装其他终端的MAC地址，从而获取局域网的网络流量。  3、实验三  ping ，中文名称因特网包探索器，是一种计算机网络管理应用，该程序通常被用于确认因特网上的一台主机是否可达 。Ping是工作在 TCP/IP网络体系结构中应用层的一个服务命令， 其原理是向特定的目的主机发送 ICMP（Internet Control Message Protocol 因特网报文控制协议）Echo回显 请求（Echo Request）报文，并根据回复的ICMP回显应答消息（Echo Reply），测试目的站是否可达并获取连接的丢包率和平均往返时间等有关状态信息。  当网络管理员在命令提示符下输入 ping 命令时，主机会向目标设备或指定的 IP 地址发送ICMP回显请求消息（ICMP Echo Request Message），这些请求消息是最多 64 字节的小数据包（在ping命令中一般为32字节），用于判断数据包是否成功到达对端。然后，发送请求的本地计算机将等待远程主机的响应。如果远程主机正在运行且网络可达，目标计算机收到回显请求消息后，会将ICMP回显响应消息（ICMP Echo Reply Message）送回原始计算机。  ping程序会在传出的回显请求（Echo Request）的可选数据区（Payload）中包含一份本地时间拷贝。这个时间和数据区域中的剩余内容均包含在返回的回显应答（Echo Reply）报文中。当应答被发出ping的收到时，ping程序注意到当前的时间，并用它减去应答中的时间，便得到了一个达到被ping主机的往返时间（RTT）估计值。由于只用到了原始发送者记录的时间，因此这个特征不会涉及到发送者和接收者之间的时钟同步。 |

|  |
| --- |
| **五、程序源代码（核心部分）**  1、实验一  无代码。  2、实验二  无代码。  3、实验三  main函数：仅展示与样例不同的部分。首先是对int信号的处理，对应函数见下文   1. if (signal(SIGINT, sigint\_handler) == SIG\_ERR) { 2. perror("ERROR"); 3. return 1; 4. } 6. while ( (c = getopt(argc, argv, "vhbc:i:qs:t:l:")) != -1) { 7. switch (c) { 9. case 'c': 10. count=atoi(optarg); 11. break; 12. case 'i': 13. time\_interval=atoi(optarg); 14. break; 15. case 's': 16. datalen=atoi(optarg); 17. if(datalen<=0){ 18. err\_quit("unvalid numer",datalen); 19. } 20. break; 21. case 'q': 22. quiet=1; 23. break; 24. case 't': 25. ttl\_value=atoi(optarg); 26. if(ttl\_value<=0){ 27. err\_quit("unvalid number",ttl\_value); 28. } 29. break; 30. case 'h': 31. print\_help(); 32. exit(EXIT\_SUCCESS); 33. case 'v': 34. verbose++; 35. break; 36. case 'b': 37. broadcast=1; 38. break; 39. case '?': 40. err\_quit("unrecognized option: %c", c); 41. }   获取目的地址主机名host，设置SIGALRM和处理函数sig\_alarm，通过调用host\_serv将host转换成addrinfo的链表，并将表头赋值为ai，调用Sock\_ntop\_host函数将其中socket地址转换为十进制字符串。将pr结构体中发送地址指向目的地址，接收地址使用calloc函数分配1个长度为ai\_addrlen的地址空间，之后调用readloop函数进一步处理。   1. pid = getpid(); 2. signal(SIGALRM, sig\_alrm); 3. ai = host\_serv(host, NULL, 0, 0); 4. name = ai->ai\_canonname; 5. printf("ping %s (%s): %d data bytes\n", ai->ai\_canonname, 6. Sock\_ntop\_host(ai->ai\_addr, ai->ai\_addrlen), datalen); 7. pr->sasend = ai->ai\_addr; 8. pr->sarecv = calloc(1, ai->ai\_addrlen); 9. pr->salen = ai->ai\_addrlen; 10. readloop();   对应的函数如下   1. void sigint\_handler(int signum) { 2. if(1){ 3. printf("--- %s ping statistics---\n", name); 4. printf("%d packets transmitted, %d received, %.1f%% loss, rtt %.3lfms\n", 5. already\_sent, packets\_received, ((already\_sent - packets\_received) / already\_sent) \* 100.0, total\_time); 6. } 7. exit(signum); 8. }   readloop函数调用socket函数创建套接口描述字，之后使用setsockopt函数设置接收数据缓冲区大小。   1. sockfd = socket(pr->sasend->sa\_family, SOCK\_RAW, pr->icmpproto); 2. setuid(getuid()); /\* don't need special permissions any more \*/ 3. size = 60 \* 1024; /\* OK if setsockopt fails \*/ 4. setsockopt(sockfd, SOL\_SOCKET, SO\_RCVBUF, &size, sizeof(size));   激活定时信号SIGALRM，sig\_alarm函数每秒调用一次send\_v4函数来发送ICMP回射请求报文，之后readloop函数调用recvfrom函数接收响应报文数据，通过gettimeofday函数获取当前时间，然后调用proc函数输出报文信息。   1. sig\_alrm(SIGALRM); /\* send first packet \*/ 2. for ( ; ; ) { 3. len = pr->salen; 4. n = recvfrom 5. (sockfd, recvbuf, sizeof(recvbuf), 0, pr->sarecv, &len); 6. if (n < 0) { 7. if (errno == EINTR) 8. continue; 9. else 10. err\_sys("recvfrom error"); 11. } 12. gettimeofday(&tval, NULL); 13. (\*pr->fproc)(recvbuf, n, &tval); 14. }   send\_v4函数部分代码如下。构造ICMP数据包格式后，调用in\_cksum函数计算校验和，然后调用sendto函数将构造好的报文发送至目的地址。   1. icmp = (struct icmp \*) sendbuf; 2. icmp->icmp\_type = ICMP\_ECHO; 3. icmp->icmp\_code = 0; 4. icmp->icmp\_id = pid; 5. icmp->icmp\_seq = nsent++; 6. gettimeofday((struct timeval \*) icmp->icmp\_data, NULL); 7. len = 8 + datalen; /\* checksum ICMP header and data \*/ 8. icmp->icmp\_cksum = 0; 9. icmp->icmp\_cksum = in\_cksum((u\_short \*) icmp, len); 10. //set TTL 11. if(setsockopt(sockfd,IPPROTO\_IP,IP\_TTL,&ttl,sizeof(ttl))<0){ 12. err\_sys("set TTL FAIL"); 13. } 14. if (broadcast && setsockopt(sockfd, SOL\_SOCKET, SO\_BROADCAST, &broadcast, sizeof(broadcast)) < 0) { 15. err\_sys("setsockopt for broadcast failed"); 16. } 17. sendto(sockfd, sendbuf, len, 0, pr->sasend, pr->salen);   proc\_v4函数关键代码如下。截取ip包中icmp数据部分，然后得到响应时间，通过tv\_sub函数得到往返时间，然后将其转化为rtt值。   1. ip = (struct ip \*) ptr; /\* start of IP header \*/ 2. hlen1 = ip->ip\_hl << 2; /\* length of IP header \*/ 3. icmp = (struct icmp \*) (ptr + hlen1); /\* start of ICMP header \*/ 4. if ( (icmplen = len - hlen1) < 8) 5. err\_quit("icmplen (%d) < 8", icmplen); 6. if (icmp->icmp\_type == ICMP\_ECHOREPLY) { 7. if (icmp->icmp\_id != pid) 8. return; /\* not a response to our ECHO\_REQUEST \*/ 9. if (icmplen < 16) 10. err\_quit("icmplen (%d) < 16", icmplen); 11. tvsend = (struct timeval \*) icmp->icmp\_data; 12. tv\_sub(tvrecv, tvsend); 13. rtt = tvrecv->tv\_sec \* 1000.0 + tvrecv->tv\_usec / 1000.0; 14. //rttmax=rtt; 15. //rttmin=rtt; 16. if(rtt>rttmax)rttmax=rtt; 17. if(rtt<rttmin)rttmin=rtt; 18. rttsum+=rtt; 19. rttcnt++; 20. total\_time += rtt; 21. avg=rttsum/(1.0\*rttcnt); 22. if(!quiet) 23. printf("%d bytes from %s: seq=%u, ttl=%d, rtt=%.3f ms\n", 24. icmplen, Sock\_ntop\_host(pr->sarecv, pr->salen), 25. icmp->icmp\_seq, ip->ip\_ttl, rtt); 26. } |

|  |
| --- |
| **六、程序扩展功能的需求分析与实现**  1、实验一  无拓展。  2、实验二  无拓展。  3、实验三  在实验基本要求上，进行了如下扩展：-c可以控制ping发包次数；-  i可以控制每一次发包的时间间隔，单位为秒；-s可以设置包的大小。   1. case 'c': 2. count=atoi(optarg); 3. break; 4. case 'i': 5. time\_interval=atoi(optarg); 6. break; 7. case 's': 8. datalen=atoi(optarg); 9. if(datalen<=0){ 10. err\_quit("unvalid numer",datalen); 11. } 12. break; |

|  |
| --- |
| **七、实验数据、结果分析**  1、实验一  路由器0的f0/0接口配置成功  路由器0的s0/2接口配置成功    路由器1的f0/0接口配置成功  d85b69cd52c8ca71d1146a1a4881a59  路由器1的网络地址正确  f9d528b3088804e53179eac36b9fb7e  PC1（192.168.13.33）可以与PC0（192.168.13.5）和PC2（192.168.113.2）通信  32f56f36ab346f8d9d98014c749c868  PC2（192.168.113.2）可以与PC0（192.168.13.5）和PC1（192.168.13.33）通信  2200e86b656e455e8df0a4e8ffce9c1  PC0（192.168.13.5）和PC1（192.168.13.33）通信  8e19408af37a4aa7a698f28e4b42d28  PC0（192.168.13.5）和PC2（192.168.113.2）通信  c1b8d52a806c3715bf7db373544472c  2、实验二   * 以太网   如图1.1所示，右侧标黑的即为以太网帧头，根据图1.2的以太网帧格式，分析可得以太网的目的地址为0x00:1f:7a:27:ae:ef，以太网的源地址为0xc0:3c:59:35:7f:0a，而类型为0x0800（IPv4）。    图1.1 以太网帧    图1.2 以太网帧格式   * IP   IP包格式如图1.4所示。通过Wireshark抓包，通过设置过滤器过滤网络数据，筛选IP数据包，如图1.3所示。具体而言，黑色背景部分为IP包，被封装在以太网帧中。其中，前四位为版本号：4（0100），即为ipv4；接着四位是IP头部长度：5，单位为4字节，即ip包头长度为5\*4=20字节；接着8位为服务类型（TOS）：00；接着16位为总长度：000x54，单位为字节，即IP包总长为84字节（包含头和数据）；接着16位为标识位：0x35f4，用于唯一标识该ip包；接着3位为标志：010，即该ip包不可分段（第二位DF为1），且该ip包没有后续分段（第三位MF为0）；接着13位为分段偏移：0 0000 0000 0000，单位为8字节，即表明当前为ip包中第一段也是唯一一段；接着8位为生存时间（TTL）：0x80（128），单位为秒，即该ip包生存时间为128秒，当经过128个路由器时该ip包的TTL减为0，会被丢弃；接着8位为协议：06，即表示高层协议为TCP类型，因此黑色背景后面的为被该ip包封装的TCP包；接着16位为头部校验和：0x0000；接着32位为源地址：0xac110654，即包的发送者的IP地址为172.17.6.84；接着32位为目的地址：0x7835f5aa，即包的接收者的IP地址为120.53.245.170。由于IP头部长度为20字节，是最短的固定长度，且上述字段已达到20字节，该ip包没有选项字段。信息如图1.5所示。    图1.3 IP头    图1.4 IP头格式    图1.5 IP头信息   * ICMP   设置过滤器目标为ICMP，选择其中的信息，如图1.6所示，其中右侧标黑的数据即为ICMP报文，ICMP可以看做为IP协议的伴随协议，ICMP报文被封装在IP数据报中发送。结合图1.7中ICMP头格式，和捕获的ICMP报文可以得知，此报文的类型为0x0b（11），代码为0x00，校验和为0xf4fe，此后的内容会根据不同类型和代码有所不同，根据翻阅资料，当代码为0，类型为11时说明超时了。ICMP头信息如图1.8所示。    图1.6 ICMP头    图1.7 ICMP头格式    图1.8 ICMP头信息   * ARP   如图1.9所示，右侧标黑的内容为ARP报文，根据图1.10中ARP的头格式分析报文，可知，硬件类型为0x01，协议类型为0x0800（IPv4），硬件地址长度为0x06，协议地址长度为0x04，操作为0x0001，发送方硬件地址为0x72:4a:2f:ba:ed:37，发送方协议地址为0xac11051b（172.17.5.27），接收方硬件地址0x00:00:00:00:00:00，接收方协议地址为0xac:11:07:71（172.17.7.113），分析结果如图中上方所示，通过Wireshark查看该ARP数据包的information一栏，发现有着“Who has 172.17.7.113？ Tell 172.17.5.27”字样，表明正在发送广播报文到以太网，查找拥有ip地址为172.17.7.113的mac地址，只有该ip地址的主机会用自己的mac地址作为相应，发给ip地址为172.17.5.27的寻找主机。    图1.9 ARP头格式    图1.10 ARP头格式   * TCP   TCP包格式如图1.11所示。通过Wireshark抓包，通过设置过滤器过滤网络数据，筛选TCP数据包，如图1.12所示。具体而言，黑色背景部分为TCP包，被封装在IPV4包中，IPV4包又被封装在以太网帧中。前16位为源端口：0x0050，即80；接着16位为目的端口：0xebe4，即60388；，用于寻找发送方和接收方应用进程；接着32位为序列号：ac 15 13 54，单位为字节，即2887062356，表示该TCP报文中的第一个TCP数据字节在从TCP发送方向TCP接收方发送的数据字节流中的位置，用于对每个字节计数，只有SYN标志置1时有效；接着32位为确认号：a9 37 d5 5d，单位为字节，即2839008605，是上次已成功收到数据字节序号加1，只有ACK标志置1时有效；接着4位为TCP头长度（含选项）：0101，单位为32位，即20字节长；接着6位保留字段：000000；接着6位标志位：01 0001，分别对应紧急指针无效（URG为0），确认号有效（ACK为1），接收方无需尽快将该报文交给应用层可以做缓存（PSH为0），无需重置一个已经混乱的连接（RST为0），序列号无效（SYN为0），发端完成发送任务要释放连接（FIN为1）；接着16位为窗口大小：0x01f6，用于基于可变滑动窗口的流控，指示发送方从被确认的字节算起可在发送502字节；接着16位为校验和：0x86fd，接着16位为紧急指针：0x0000，用于指示紧急数据在当前数据段中的位置（相对于当前序列号的字节偏移值），URG置1时有效，该包URG为0，因此该字段无效。    图1.11 TCP格式  149d3c28c7dd9f5cfa780a4568dedf2  图1.12 TCP包跟踪结果（a）  c86f8abd896ff84fa6c6df0797eed3a  图1.13 TCP包跟踪结果（b）   * UDP   如图1.14所示，标黑的内容为UDP报头，根据图1.15中的UDP报头格式，分析可得，源端口为0xe94c（59724），目的端口为0x1fa3（8099），UDP长度为0x0030（48字节），UDP校验和为0x252e，而后的内容为UDP payload，共占48-8=40 bytes。结果如图1.16所示。    图1.14 UDP包头    图1.15 UDP包头格式    图1.16 UDP包头信息   * DNS   DNS包格式如图1.17所示。通过Wireshark抓包，通过设置过滤器过滤网络数据，筛选DNS数据包，如图1.18所示。具体而言，黑色背景部分为DNS包，通过UDP-IPV4-以太网帧层层封装。其中，前16位为标识：0xefb2；接着16位为标志：0x8180（1[QR] 0000[opcode] 0[AA] 0[TC] 1[RD] 1[RA] 000[zero] 0000[rcode]），即该报文为响应报文（QR为1），为标准查询报文（opcode为0），不授权回答（AA为0），不可截断（TC为0），期望是递归查询（RD为1），可用递归（RA为1），没有差错（rcode为0）；接着16位为问题数：0x0001，即一个问题；接着16位为回答数：0x0001，即一个回答；接着16位为授权资源记录数：0x0000；接着16位附加资源记录数：0x0000；接下来为一个问题区域和一个回答区域。问题区域：查询域名为cupgrade.imeitools.com，查询类型为A（即IPV4地址），查询类为IN（即Internet互联网地址）；回答区域：查询域名为cupgrade.imeitools.com，查询类型为A（即IPV4地址），查询类为IN（即Internet互联网地址），生存时间为177，即2分57秒，数据长度为4，查询结果IP地址为6a.37.db.ea，即106.55.219.234。总之该报文为递归式的标准查询响应报文，且没有差错。    图1.17 DNS格式    图1.18 DNS包跟踪结果  3、实验三  -v功能：默认模式    -q功能：仅输出结果    -t：设置为10，没有输出，设置为20，可以正常接收    -c：设置发2个包    -s：设置包大小    -b：广播地址ping，仅得到了百度的回应    -h：输出help信息    -i：在运行的时候，可以看到每隔2s，ping一次。本功能扩展没有通过print来体现出时间间隔。 |
| 1. 总结   实验一加深了对网络设备配置的理解，尤其是路由器的配置流程和静态路由的重要性。通过实际操作，掌握了网络设备的基本配置方法，对网络拓扑结构和数据传输过程有了直观的体验。  实验二的Wireshark的使用极大地提升了网络故障排查的能力，对网络协议的理解不再停留在理论层面。通过分析实际的网络数据，对协议的工作流程和数据包结构有了更深刻的认识，为今后的网络故障排查和优化打下了坚实的基础。  实验三不仅提升了代码分析和功能开发的能力，还加深了对Linux操作系统的理解。通过亲手实现Ping程序的扩展功能，对网络编程和系统调用有了更直观的感受，增强了在实际项目中解决问题的信心。  通过这三个实验，不仅巩固了理论知识，更重要的是，通过实际操作，将理论知识转化为实践能力，提升了网络设备配置、网络数据包分析和编程技能。实验过程中的挑战和解决过程，极大地锻炼了解决问题的能力，为未来的学习和工作打下了坚实的基础。实验不仅是一次技术实践，更是一次自我挑战和提升的过程，收获颇丰。 |

九、同组人分工情况

|  |  |  |
| --- | --- | --- |
| **学号** | **姓名** | **承担任务** |
| 21071014 | 李昂 | 实验一、实验二 |
| 21071003 | 高立扬 | 实验三 |
| 21071015 | 李博文 | 实验一、实验二 |