《计算机网络》课程设计内容

计算机网络课程设计共分以下 2 个部分：

（1） 题目一：数据包的封装发送和解析（ARP/IP/TCP 均可）；

（2） 题目二：可选题目中选择一个，也可自拟（需和老师商定）

课程设计采取小组的形式，原则上每组 3 人，共同完成设计内容。课程设计报告以小组形式提交，小组成员在课程设计中分担的工作量在报告中体现，按照承担任务的复杂性和工作量的不同，最后评分会略有不同。

提交报告的格式见文件。

一．具体设计要求：

题目一：数据包的封装发送和解析（ARP/IP/TCP）

课程设计目的：

- ARP 协议用于完成 IP 地址与 MAC 地址之间的转换。IP 协议是网络层的重要协议，TCP 是运输层的可靠传输协议。通过封装与发送这些数据包，加深对网络协议的理解，掌握 ARP/IP/TCP 帧结构和工作原理及其对协议栈的贡献。

课程设计要求：

- 编写程序，根据 ARP/IP/TCP 帧的结构，封装数据包发送到局域网中。另外要捕获网络中的 TCP、IP 和 ARP 数据包，解析数据包的内容，并将结果显示，并同时写入日志文件。
- 以命令行形式运行或图形界面
- 在标准输出中显示捕获的 TCP、IP 和 ARP 报文的首部字段的内容。

课程设计分析：

- 使用原始套接字或者 winpcap，捕获 ARP/IP 和 TCP 数据包
- 定义 ARP/IP/TCP 首部的数据结构
- 自定义并填充数据包，发送数据包，捕获数据包

题目二：(以下题目中任选一个)

题目 1：FTP 客户机

设计目的：

- 设计并实现一个 FTP 客户机的程序，了解 FTP 服务的基本原理和 FTP 协议的工作过程，加深对 TCP 协议和流式套接字编程的理解。

设计要求：

- 要求在 FTP 客户机程序中至少实现目录变换（CWD），列表（LIST），下载（RETR）功能。
- 以命令行形式或图形界面运行
- 输出内容：FTP 客户机与服务器交互过程中的命令与应答信息。下载指定 FTP 网站的指定文件（学校 FTP 网址：202.204.60.11）

题目 2：局域网文字聊天室

设计目的：

- 设计并实现一个局域网内的文字聊天小程序，使用 Winsocket API 或 Winpcap 技术，完成局域网内主机之间文字信息传送

设计要求：

- 实现一对一文字对话
- 实现一对多文字通话
- 以命令行形式或图形界面运行

题目 3：HTTP 浏览器

设计目的：

- 设计并实现一个 HTTP 浏览器，使用 Win socket API 或 Winpcap 技术，实现在客户机上显示 WEB 文档

设计要求：

- 以命令行形式或图形界面运行
- 实现输入一个连接打开文档，该文档以 index.html 命名存在另外一台机器中

题目 4：多线程 Web 服务器

设计目的：

- 设计并实现一个 WEB 服务器，使用 Win socket API 或 Winpcap 技术，并行服务于多个请求，实现 HTTP1.0 的功能

设计要求：

- 以命令行形式或图形界面运行
- 主线程监听客户机的连接建立请求，为每次请求/响应创建一个单独

的 TCP 连接，一个单独的线程将处理这些连接。

- 用浏览器打开，显示请求页面内容即可；如有查错，则显示出错信息。
- 不能用 80 端口号，得另指定一个端口

题目 5：路由跟踪小程序

设计目的：

- 设计并实现一个基于 IP 的路由跟踪小程序，使用 IP 协议，记录该数据包经过的路由器，记录 IP 地址和往返时间，类似于 tracert。

设计要求：

- 以命令行形式或图形界面运行
- 如有余力，可考虑如果无法得到某路由器的 IP 地址，有什么替代办法

题目 6：子网内文件传送

设计目的：

- 设计并实现一个局域网内部的文件传送工具，使用 TCP 协议进行可靠文字传输。可参考飞鸽传书。

设计要求：

- 以命令行形式或图形界面运行
- 不同结点上文件自动同步

题目 7： WEB 代理服务器

设计目的：

- 设计并实现一个 WEB 代理服务器，使用 HTTP 1.0 协议，记录管辖范围内的 GET 请求，缓存 web 页面，接收浏览器发来的 GET 报文，向目的服务器转发 GET 报文和响应报文。存储返回的响应报文，记录生存时间

设计要求：

- 以命令行形式或图形界面运行

题目 8：统计局域网内流量与网内各活动主机状态

设计目的：

- 设计并实现一个监听 TCP/IP 网络流量的小程序，统计该网卡子网内的不同协议流量。
- 记录局域网内活动主机数量和活动端口，记录该活动主机的 IP 地址和端口号。

设计要求：

- 以命令行形式或图形界面运行

题目 9：简单邮件代理器

设计目的：

- 设计并实现一个邮件收发器（类似于 outlook），使用 SMTP 协议和 POP3 协议，使用 socket 编程。

设计要求：

- 命令行或图形界面运行
- 按照指定地址，完成文字发送

题目 10：网页及链接下载

设计目的：

- 设计并实现一个下载程序，根据输入的网页 URL，将该网页和网页上链接的资源（包括链接的网页）都下载并存储。

设计要求：

- 命令行或图形界面运行
- 只下载输入 URL 路径下及子路径下的相关资源
- 可指定下载的目录层数，并保证资源不会重复下载
- 下载后的资源间要保持相对路径，下载的网页文件中的超链接要修改为指向下载得到的本地资源。

题目 11：简单远程程序调用

设计目的：

- 设计并实现一个 C/S 应用，客户端接受用户从键盘输入的命令，并发送给服务器，服务器端执行该命令，并将命令的执行结果返回给客户端，客户端显示命令执行结果。类似于 Telnet 应用。

设计要求：

- 以命令行方式运行。
- 如果服务器端无法识别该命令，给出错误提示。

题目 12：UDP 网络吞吐量测试小程序

设计目的：

- 设计并实现一个 C/S 应用，客户端向服务器的指定端口发送 UDP 数据包，服务器向客户端返回收到的数据量，由客户端计算吞吐量并显示。

设计要求：

- 命令行或图形界面运行。
- 使用 UDP 协议。
- 应用分为握手、发包测试，和统计结果三个阶段。
- 由客户端启动和停止发包。
- 由客户端指定发包大小和发包频率。

题目 13：隧道技术

设计目的：

- 设计并实现一个 IP 或 HTTP 隧道，客户端和服务器作为隧道的出入口，其他程序可以通过隧道程序向对端发送数据。

设计要求：

- 命令行或图形界面运行。

题目 14：域名查询程序

设计目的：

- 设计并实现一个类似 nslookup 的 DNS 客户端，可以根据用户输入的查询请求，向本地域名服务器或根域名服务器发出请求，并显示查询结果。

设计要求：

- 命令行或图形界面运行。

题目 15：实现一个可靠传输协议

设计目的：

- 设计并实现一个简单的可靠传输协议（类似停等协议），可以实现 2 个主机凭借该协议实现主机 A 发送的信息在 IP 数据协议的基础上，保证可靠性，设计并实现该可靠协议的首部和发送/接收端的动作。

设计要求：

- 命令行或图形界面运行。

题目 16：实现基于最短路径计算的路由算法

设计目的：

- 设计并实现一个简单的路由算法，可以根据路由器的拓扑图（节点数>10），计算每个节点的路由表；当某个节点失效时，能检测到该节点失效并重新计算路由，显示各个节点的路由表。

设计要求：

- 命令行或图形界面运行。

题目 17：自拟题目

设计要求：使用网络底层开发，使用上学期课堂上学过的网络协议（路由协议，其他应用层协议）。

二、课程设计相关参考

1. Winpcap 结构体和函数说明

a) 结构体

在程序中，将要进行分析的各数据包头格式用结构体进行定义。这样便于对数据包的解析，使每个字段清楚易懂。

如：

- 以太帧头格式结构体，共 14 个字节：

```
typedef struct ether_header {
    unsigned char ether_dhost[6];      //目的 MAC 地址
    unsigned char ether_shost[6];      //源 MAC 地址
    unsigned short ether_type;              //协议类型
}
```

- IPv4 报头格式结构体，共 20 个字节：

```
typedef struct ipv4_header {
unsigned char ver_ihl;              //版本  (4 bits) +  首部长度  (4 bits)
unsigned char tos;                  //服务类型
unsigned short tlen;                //数据报总长度
unsigned short identification;      //标识
unsigned short flags_fo;            //标志  (3 bits) +  片偏移  (13 bits)
    unsigned char ttl;                   //生存时间
unsigned char proto;                //协议
unsigned short crc;                 //首部校验和
u_char ip_src[4];                   //源 IP 地址
u_char ip_dst[4];                   //目的 IP 地址
}
```

- IPv6 报头格式结构体，共 40 个字节：

```
typedef struct ipv6_header {
    u_char ver_tf;                  //版本号（4 bit）
```

```
    u_char traffic;                 //优先级（8 bit）
    u_short label;                  //流标识（20 bit）
    u_char length[2];               //报文长度（16 bit）
    u_char next_header;             //下一头部（8 bit）
    u_char limits;                  //跳数限制（8 bit）
    u_char Srcv6[16];               //源 IPv6 地址（128 bit）
    u_char Destv6[16];              //目的 IPv6 地址（128 bit）
}
```

- TCP 报头格式结构体，共 20 个字节：

```
typedef struct tcp_header {
    WORD SourPort;          //源端口号
    WORD DestPort;          //目的端口号
    DWORD SeqNo;            //序号
    DWORD AckNo;            //确认序号
    BYTE HLen;              //首部长度（保留位）
    BYTE Flag;              //标识（保留位）
    WORD Window;            //窗口大小
    WORD ChkSum;            //校验和
    WORD UrgPtr;            //紧急指针
}
```

- UDP 报头格式结构体，共 8 个字节：

```
typedef struct udp_header {
    u_short sport;          //源端口号
    u_short dport;          //目的端口号
    u_short len;            //数据报长度
    u_short crc;            //校验和
}
```

b) 常用函数

LPPACKET PacketAllocatePacket(void)

VOID PacketInitPacket(LPPACKET lpPacket, PVOID Buffer, UINT Length)

VOID PacketFreePacket(LPPACKET lpPacket)

VOID PacketCloseAdapter(LPADAPTER lpAdapter)

BOOLEAN PacketGetAdapterNames(LPSTR pStr,PULONG BufferSize)

LPADAPTER PacketOpenAdapter(LPTSTR AdapterName)

BOOLEAN PacketReceivePacket(LPADAPTER AdapterObject,LPPACKET lpPacket, BOOLEAN Sync)

BOOLEAN PacketSetHwFilter(LPADAPTER AdapterObject,ULONG Filter)

BOOLEAN PacketGetNetInfoEx(LPTSTR AdapterNames,npf_ip_addr *buff, PLONG NEntries)

BOOLEAN PacketSendPacket(LPADAPTER AdapterObject,LPPACKET lpPacket, BOOLEAN Sync)

int pcap_findalldevs(pcap_if_t **alldevsp, char *errbuf)

char *pcat_lookupdev(char *errbuf)

int pcap_lookupnet(char *device, bpf_u_int32 *netp,bpf_u_int32 *maskp, char *errbuf)

pcap_dumper_t *pcap_dump_open(pcap_t *p,char *filename)

Pcap_t * pcap_open_live(char * DeviceName,int snaplen,int promisc,int to_ms,char *errbuf)

int pcap_compile (pcap_t *p, struct bpf_program *fp, char *str, int optimize, bpf_u_int32 netmask)

int pcap_setfilter (pcap_t *p, struct bpf_program *fp)

int pcap_dispatch(pcap_t *p, int cnt,pcap_handler callback, u_char *user)

int pcap_loop (pcap_t *p, int cnt, pcap_handler callback, u_char*user)

pcap_read()

u_char *pcap_next(pcap_t *p, struct pcap_pkthdr *h)

void pcap_close (pcap_t *p)

int pcap_setbuff(pcap_t *p,int dim)

int pcap_setmode(pcap_t *p,int mode)

pcap_stats(pcap_t *p, struct pcap_stat *ps)

int pcap_sendpacket(pcap_t *p,char *buf,int size)

FILE *pcap_file(pcap_t *p)

int pcap_fileno(pcap_t *p)

2. WinSock 常用结构体和函数说明

a) 结构体：

```
struct sockaddr {
        u_short sa_family;              /* address family */
        char    sa_data[14];            /* up to 14 bytes of direct address */
};

struct sockaddr_in {
        short   sin_family;
        u_short sin_port;
        struct  in_addr sin_addr;
        char    sin_zero[8];
};
```

```
struct in_addr { //采用不同方式定义一个32比特无符号数
        union {
                struct { u_char s_b1,s_b2,s_b3,s_b4; } S_un_b;
                struct { u_short s_w1,s_w2; } S_un_w;
                u_long S_addr;
        } S_un;
#define s_addr    S_un.S_addr
                                        /* can be used for most tcp & ip code */
#define s_host    S_un.S_un_b.s_b2
                                        /* host on imp */
#define s_net     S_un.S_un_b.s_b1
                                        /* network */
#define s_imp      S_un.S_un_w.s_w2
                                        /* imp */
#define s_impno S_un.S_un_b.s_b4
                                        /* imp # */
#define s_lh       S_un.S_un_b.s_b3
                                        /* logical host */
};

struct    hostent {
        char       * h_name;                /* official name of host */
        char       * * h_aliases;    /* alias list */
        short      h_addrtype;              /* host address type */
        short      h_length;                /* length of address */
        char       * * h_addr_list; /* list of addresses */
#define h_addr    h_addr_list[0]           /* address, for backward compat */
};
```

b) 常用 socket 函数

```
int WSAStartup(WORD wVersionRequired, LPWSADATA lpWSAData);
int WSACleanup(void);

SOCKET accept (SOCKET s, struct sockaddr *addr, int *addrlen);
int bind (SOCKET s, const struct sockaddr *addr, int namelen);
int closesocket (SOCKET s);
int connect (SOCKET s, const struct sockaddr *name, int namelen);

int ioctlsocket (SOCKET s, long cmd, u_long *argp);
int getsockopt (SOCKET s, int level, int optname,
                                char * optval, int *optlen);
u_long htonl (u_long hostlong);
u_short htons (u_short hostshort);
unsigned long inet_addr (const char * cp);
```

```
char * inet_ntoa (struct in_addr in);
int listen (SOCKET s, int backlog);
u_long ntohl (u_long netlong);
u_short ntohs (u_short netshort);

int recv (SOCKET s, char * buf, int len, int flags);
int recvfrom (SOCKET s, char * buf, int len, int flags, struct sockaddr *from, int * fromlen);

int select (int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, const struct timeval *timeout);

int send (SOCKET s, const char * buf, int len, int flags);

int sendto (SOCKET s, const char * buf, int len, int flags, const struct sockaddr *to, int tolen);

int setsockopt (SOCKET s, int level, int optname, const char * optval, int optlen);

int shutdown (SOCKET s, int how);

SOCKET socket (int af, int type, int protocol);

struct hostent * gethostbyaddr(const char * addr,    int len, int type);

struct hostent * gethostbyname(const char * name);

int gethostname (char * name, int namelen);

struct servent * getservbyport(int port, const char * proto);

struct servent * getservbyname(const char * name, const char * proto);

struct protoent * getprotobynumber(int proto);

struct protoent * getprotobyname(const char * name);
```