1. 0叙述tcp/ip四层，每层协议（两个）

应用层：负责处理特定的应用程序细节

Telnet、FTP和e-mail等

传输层：主要为两台主机上的应用提供端到端的通信

TCP和UDP

网络层：处理分组在网络中的活动，比如分组的选路。

IP、ICMP和IGMP

链路层：包括操作系统中的设备驱动程序、计算机中对应的网络接口卡

设备驱动程序及接口卡

以太网，令牌环网，FDDI，IEEE802.3

1. 标识一次链接需要几个要求

协议，本地IP地址，本地端口号，远程IP地址，远程端口号。

//服务器端和客户端的ip端口号，连接套接字。

1. 创建tcpsocket udpsocket程序片段

TCP: SOCKET sockfd = socket(AF\_INET,SOCK\_STREAM,0);

UDP: SOCKET sockfd = socket(AF\_INET,SOCK\_DGRAM,0);

if(sockfd = INVALID\_SOCKET){

printf(“socket error: %d\n”,WSAGetLastError());

}

1. tcp通信模型，单线程、多线程p132

单线程： 多线程

服务器 客户机 服务器 客户机

socket socket socket socket

bind bind

listen listen

accept

等待客户连接请求 connect accept connect

recv send thread{

处理 recv send

send recv send recv

closesocket closesocket }closesocket closesocket

1. udp通信模型，循环服务器

服务器 客户机

socket socket

bind sendto

recvfrom recvfrom

sendto closesocket

closesocket

1. 服务器两种绑定bind含义区别、代码片段

程序类型 IP址 端口号 说明

服务器 INADDR\_ANY 非零值 指定服务器的公认端口号

该bind方式表示它愿意接收来自任何网络设备接口的客户机连接。

服务器 本地IP地址 非零值 指定服务器的IP地址和公认端口号

该bind方式表示服务器只接收来自对应于这个IP地址的特定网络设备接口的客户机连接。

1. listen函数功能，第二个参数含义是什么

功能：（1）将一个已绑定的套接字转换为倾听套接字，使该套接字可以接收连接请求。

1. TCP协议将到达的连接请求排队，第二个参数指定这个队列的最大长度。
2. accept函数功能，返回的socket与参数中的socket区别

功能：从倾听套接字的已完成连接队列中接受一个连接。

区别：前一个socket是在服务器进程中创建的倾听套接字，用来接收所有客户机的连接请求，服务器进程终止前关闭这个倾听套接字；而对于每个接受的连接，TCP协议都创建一个新的连接套接字，来标识这个连接，服务器使用这个连接套接字与客户机进行通信操作，当服务器处理完这个客户机请求时，关闭此连接套接字。

1. 连接型udp是解决什么问题，用于什么场合，用于哪一方、创建一个连接型的udp

只用于udp客户机。原因是要求内核进行udp包的过滤。

1. 给出一个结构化传输的方案，如包含中文的字符串，彩色身份证，二叉树

先将需要传输的数据统一编码为UTF-8，传输时先传输字符长度，再传输字符的内容。

1. win32线程写代码

步骤：

WSAStartup() 初始化winsock

socket()，创建TCP类型的socket

bind()，绑定服务器的IP、PORT

listen()，监听socket

while(continue){

accept()，接收客户端的连接

创建新线程，传入通信用的socket，与客户端进行通信

}

closesocket()，关闭socket

WSACleanup()，释放winsock资源

代码：

#include <stdio.h>

#include <process.h>

#include <WinSock2.h>

unsigned int CALLBACK myThread(void \*p);

int main(){

HANDLE handles[2];

handles[0] = (HANDLE)\_beginthreadex(NULL,0,myThread,NULL,0,NULL);

handles[1] = (HANDLE)\_beginthreadex(NULL,0,myThread,NULL,0,NULL);

WaitForMultipleObjects(2,handles,true,INFINITE);

CloseHandle(handles[0])

CloseHandle(handels[1]);

return 0 ;

}

unsigned int CALLBACK myThread(void \*p){

for(int i = 1;i <= 10;i++){

printf(“Thread:%u 打印: %d\n”,GetCurrentThreadId(),i);

sleep(2000)

}

return 0;

}

1. udp数据的最大长度

udp包的大小应该是1500 - IP头(20) - UDP头(8) = 1472(Bytes)

1. ip包记录路由最多几个

15个

1. 写出域名解析程序对应ip地址

int main(){

WSADATA wsa;

WSAStartup(MAKEWORD(2,2),&wsa);

honest \*phost;

char \*pstr;

phost = getpostbyname(“www.baidu.com”);

printf(“%s”,phost->h\_name);

int i = 0;

for(pstr = phost->h\_aliases[i];pstr!=NULL;pstr = phost->h\_aliases[++i]){

printf(“%s”,pstr);

}

if(phost->h\_addrtype == AF\_INET){

printf(“ipv4”);

}

else(phost->h\_addrtype == AF\_INET6){

printf(“ipv6”)

}

printf(“%d”,phost->h\_length);

i = 0;

for(pstr = phost->h\_addr\_list[i];pstr != NULL;ptsr = phost->h\_addr\_list[i]){

u\_long temp;

temp = \*(u\_long\*)pstr;

in\_addr in;

in.S\_un.S\_addr = temp;

printf(“%s”,inet\_ntoa(in));

}

WSACleanup();

return 0;

}

1. 基于tcp 的通信程序

服务器端

#include "winsock2.h"

#include "stdio.h"

void main(int argc, char \*\*argv){

WSADATA wsaData;

SOCKET s;

SOCKADDR\_IN ServerAddr;

int Port = 9999;

int Ret;

if (argc <= 1){

printf("用法: tcpclient <服务器IP地址>.\n");

return;

}

// 初始化 Winsock version 2.2

if ((Ret = WSAStartup(MAKEWORD(2,2), &wsaData)) != 0) {

printf("WSAStartup 出错。错误号： %d\n", Ret);

return;

}

// 建立连接

if ((s = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP))

== INVALID\_SOCKET){

printf("socket 出错。错误号： %d\n", WSAGetLastError());

WSACleanup();

return;

}

// 设置 SOCKADDR\_IN 结构 ，在端口号 9999上连接。

ServerAddr.sin\_family = AF\_INET;

ServerAddr.sin\_port = htons(Port);

ServerAddr.sin\_addr.s\_addr = inet\_addr(argv[1]);

// 向服务器发起连接.

printf("正在连接..... %s:%d...\n",

inet\_ntoa(ServerAddr.sin\_addr), htons(ServerAddr.sin\_port));

if (connect(s, (SOCKADDR \*) &ServerAddr, sizeof(ServerAddr))

== SOCKET\_ERROR) {

printf("connect 出错。错误号： %d\n", WSAGetLastError());

closesocket(s);

WSACleanup();

return;

}

printf("连接成功.\n");

// 通信处理的代码.

printf("开始发送信息:你好！......\n");

Sleep(10000);

if ((Ret = send(s, "你好！", 6, 0)) == SOCKET\_ERROR){

printf("send 出错。错误号： %d\n", WSAGetLastError());

closesocket(s);

WSACleanup();

return;

}

printf("成功发送 %d 个字节.\n", Ret);

// 通信完成

printf("开始关闭连接.\n");

closesocket(s);

// 从内存中卸载DLL

WSACleanup();

}

客户端

#include <winsock2.h>

#include <stdio.h>

void main(void){

WSADATA wsaData;

SOCKET ListeningSocket;

SOCKET NewConnection;

SOCKADDR\_IN ServerAddr;

SOCKADDR\_IN ClientAddr;

int ClientAddrLen;

int Port = 9999;

int Ret;

char DataBuffer[1024];

// 初始化 Winsock version 2.2

if ((Ret = WSAStartup(MAKEWORD(2,2), &wsaData)) != 0)

{

printf("WSAStartup 出错。错误号： %d\n", Ret);

return;

}

// 创建一个倾听型Socket.

if ((ListeningSocket = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP))

== INVALID\_SOCKET){

printf("socket 出错。错误号： %d\n", WSAGetLastError());

WSACleanup();

return;

}

// 填充SOCKADDR\_IN 结构。

ServerAddr.sin\_family = AF\_INET;

ServerAddr.sin\_port = htons(Port);

ServerAddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

// 绑定Socket

if (bind(ListeningSocket, (SOCKADDR \*)&ServerAddr, sizeof(ServerAddr))

== SOCKET\_ERROR){

printf("bind 出错。错误号： %d\n", WSAGetLastError());

closesocket(ListeningSocket);

WSACleanup();

return;

}

// 转换成倾听Socket

if (listen(ListeningSocket, 5) == SOCKET\_ERROR){

printf("listen 出错。错误号： %d\n", WSAGetLastError());

closesocket(ListeningSocket);

WSACleanup();

return;

}

printf("在端口号： %d 上等待连接.\n", Port);

// 接受连接.

ClientAddrLen=sizeof(ClientAddr);

if ((NewConnection = accept(ListeningSocket, (SOCKADDR \*) &ClientAddr,

&ClientAddrLen)) == INVALID\_SOCKET){

printf("accept 出错。错误号： %d\n", WSAGetLastError());

closesocket(ListeningSocket);

WSACleanup();

return;

}

printf("连接成功。连接来自 %s:%d.\n",

inet\_ntoa(ClientAddr.sin\_addr), ntohs(ClientAddr.sin\_port));

// 此时可关闭倾听Socket。因为不想再等待新的连接了。

closesocket(ListeningSocket);

// 开始接受数据

//设置读的超时选项：不超过３秒。３秒一到，不再等待读。

int time = 3000;

setsockopt(NewConnection,SOL\_SOCKET,SO\_RCVTIMEO,(char \*)&time,sizeof(time));

int ntime=0;

int isize=sizeof(int);

getsockopt(NewConnection,SOL\_SOCKET,SO\_RCVTIMEO,(char\*)&ntime,&isize);

printf("接受超时时间：%d\n",ntime);

printf("开始接受数据...\n");

if ((Ret = recv(NewConnection, DataBuffer, sizeof(DataBuffer), 0))

== SOCKET\_ERROR){

printf("recv 出错。错误号： %d\n", WSAGetLastError());

closesocket(NewConnection);

WSACleanup();

return;

}

DataBuffer[Ret]='\0';

printf("成功接受 %d 个字节.\n数据内容是：%s\n", Ret,DataBuffer);

printf("关闭连接.\n");

closesocket(NewConnection);

WSACleanup();

}

1. 简单的远程遥控程序：支持list、get、post、quit

服务器端

#include "winsock2.h"

#include "stdio.h"

void \_ERROR(char \* s)

{

printf("%s 出错！出错代码：%d \n",s,WSAGetLastError());

}

int count(char \*s)

{

int cc=0;

for (int i=0;s[i]!=0;i++)

{

if (isalpha(s[i]))

{

cc++;

}

}

return cc;

}

void main(void)

{

WSADATA wsaData;

if (WSAStartup(0x0202,&wsaData)!=0)

{

\_ERROR("WSAStartup()");

exit(-1);

}//if

SOCKET lsd = socket(AF\_INET,SOCK\_STREAM,IPPROTO\_TCP);

if (lsd==INVALID\_SOCKET)

{

\_ERROR("socket()");

WSACleanup();

exit(-1);

}//if

//server IP:127.0.0.1 port : 9999

struct sockaddr\_in server;

memset(&server,0,sizeof(server));

server.sin\_family=AF\_INET;

server.sin\_port=htons(9999);

server.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

if (bind(lsd,(struct sockaddr \*)&server,sizeof(server))==SOCKET\_ERROR)

{

\_ERROR("bind()");

closesocket(lsd);

WSACleanup();

exit(-1);

}

if (listen(lsd,5)==SOCKET\_ERROR)

{

\_ERROR("listen()");

closesocket(lsd);

WSACleanup();

exit(-1);

}

printf("TCP服务器程序在端口号：9999上等待连接......\n");

struct sockaddr\_in caddr;

int len = sizeof(caddr);

SOCKET clisd = accept(lsd,(struct sockaddr \*)&caddr,&len);

if ( clisd == INVALID\_SOCKET )

{

\_ERROR("accept()");

closesocket(lsd);

WSACleanup();

exit(-1);

}

printf("客户机IP：%s 端口号：%d 已连接进来！\n",inet\_ntoa( caddr.sin\_addr),ntohs(caddr.sin\_port));

char data[254];

int cc=0;

do {

int rc = recv(clisd,data,250,0);

if (rc==0) break;

if (rc==SOCKET\_ERROR)

{

\_ERROR("recv()");

closesocket(clisd);

closesocket(lsd);

WSACleanup();

exit(-1);

}

data[rc]=0;

printf("客户机数据：%s \n",data);

cc = count(data);

char rdata[254];

sprintf(rdata,"字母个数：%d \n",cc);

rc=send(clisd,rdata,strlen(rdata),0);

if (rc==SOCKET\_ERROR)

{

\_ERROR("send()");

closesocket(clisd);

closesocket(lsd);

WSACleanup();

exit(-1);

}

} while(strcmp(data,"quit")!=0);

printf("谢谢，再见！");

closesocket(clisd);

closesocket(lsd);

WSACleanup();

}

客户端

#include "winsock2.h"

#include "stdio.h"

void \_ERROR(char \* s)

{

printf("%s出错！错误代码：%d\n",s,WSAGetLastError());

}

void main(void)

{

WSADATA wsaData;

if (WSAStartup(0x0202,&wsaData)!=0)

{

\_ERROR("WSAStartup()");

exit(-1);

}

SOCKET sd = socket(AF\_INET,SOCK\_STREAM,IPPROTO\_TCP);

if (sd==INVALID\_SOCKET)

{

\_ERROR("socket()");

WSACleanup();

exit(-1);

}

struct sockaddr\_in saddr;

memset(&saddr,0,sizeof(saddr));

saddr.sin\_family=AF\_INET;

saddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

saddr.sin\_port=htons(9999);

printf("向服务器：127.0.0.1 端口号：9999 发起连接...");

if (connect(sd,(struct sockaddr \* )&saddr,sizeof(saddr))==SOCKET\_ERROR)

{

\_ERROR("connect()");

closesocket(sd);

WSACleanup();

exit(-1);

}

printf("已连接成功！\n");

char data[254];

char rbuf[254];

do {

gets(data);

send(sd,data,strlen(data),0);

int rc=recv(sd,rbuf,254,0);

rbuf[rc]=0;

printf("服务器响应：%s\n",rbuf);

} while(strcmp(data,"quit")!=0);

printf("再见！\n");

closesocket(sd);

WSACleanup();

}

1. 基于udp的网络通讯程序

服务器端

#include "winsock2.h"

#include "stdio.h"

#include "stdlib.h"

int count(char\*s)

{

int cc=0;

for (int i=0;s[i];i++)

{

if ( isalpha(s[i]) )

{

cc++;

}

}//for(i)

return cc;

}

void main(void)

{

WSADATA wsaData;

WSAStartup(0x0202,&wsaData);

SOCKET s = socket(AF\_INET,SOCK\_DGRAM,0);

sockaddr\_in sip;

memset(&sip,0,sizeof(sip));

sip.sin\_family=AF\_INET;

sip.sin\_port=htons(9999);

sip.sin\_addr.s\_addr=htonl(INADDR\_ANY);

bind(s,(sockaddr\*)&sip,sizeof(sip));

printf("UDP服务器在9999上接受UDP包....\n");

int stopIt=0;

while (!stopIt)

{

char rd[254];

int len=sizeof(sip);

int n=recvfrom(s,rd,253,0,(sockaddr\*)&sip,&len);

rd[n]='\0';

printf("客户机IP：%s 端口号：%d 发来UDP包，内容是：%s\n",

inet\_ntoa(sip.sin\_addr),ntohs(sip.sin\_port),rd

);

char sd[254]="";

sprintf(sd,"字母个数：%d",count(rd));

sendto(s,sd,strlen(sd),0,(sockaddr\*)&sip,sizeof(sip));

}

closesocket(s);

WSACleanup();

}

客户端

#include "winsock2.h"

#include "stdio.h"

#include "stdlib.h"

void main(void)

{

WSADATA wsaData;

WSAStartup(0x0202,&wsaData);

SOCKET s=socket(AF\_INET,SOCK\_DGRAM,0);

char data[254];

sockaddr\_in sip;

memset(&sip,0,sizeof(sip));

sip.sin\_family=AF\_INET;

sip.sin\_port=htons(9999);

sip.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

do{

gets(data);

sendto(s,data,strlen(data),0,(sockaddr\*)&sip,sizeof(sip));

char rd[254];

int n=recvfrom(s,rd,253,0,NULL,NULL);

rd[n]=0;

printf("UDP服务器的响应是：%s\n",rd);

}while(strcmp(data,"quit")!=0);

printf("再见！");

closesocket(s);

WSACleanup();

}

1. 广播（定时器每隔几秒发一个通知）

服务器端

#include "winsock2.h"

#include "stdio.h"

#include "time.h"

#define BUFLEN 255

void getcurtime(char\* curtime)

{

time\_t tm;

time(&tm);

sprintf(curtime, "%s\n", ctime(&tm));

}

void main(int argc,char\*\* argv)

{

struct sockaddr\_in peeraddr;

int sockfd, on = 1;

char msg[BUFLEN+ 1 ];

if (argc!=3) {

printf("Usage: %s <ip address> <port>\n", argv[0]);

exit(0);

}

WSADATA wsaData;

if( WSAStartup(0x0202,&wsaData)!=0 )

{

printf("WSAStartup error!\n");

return ;

}

sockfd = socket(AF\_INET, SOCK\_DGRAM,0);

if (sockfd<0) {

fprintf(stderr, "socket creating error \n");

exit(1);

}

//setsockopt(sockfd, SOL\_SOCKET, SO\_BROADCAST, (char\*)&on, sizeof(on)); //允许发广播包

memset(&peeraddr, 0, sizeof(peeraddr));

peeraddr.sin\_family = AF\_INET;

peeraddr. sin\_addr.s\_addr = inet\_addr(argv[1]);

peeraddr.sin\_port = htons(atoi(argv[2]));

for(;;) {

getcurtime(msg);

printf("向组%s中发送时间:%s \n",argv[1],msg);

sendto(sockfd, msg, strlen(msg), 0, (sockaddr\*)&peeraddr, sizeof(struct sockaddr\_in));

Sleep(3000);/\* sleep 3 seconds between send \*/

}

closesocket(sockfd);

WSACleanup();

}

客户端

#include "winsock2.h"

#include "stdio.h"

#include "stdlib.h"

#include "Ws2tcpip.h"

void main(void)

{

WSADATA wsaData;

WSAStartup(0x0202,&wsaData);

SOCKET s=socket(AF\_INET,SOCK\_DGRAM,0);

sockaddr\_in sip;

int len = sizeof(sip);

memset(&sip,0,sizeof(sip));

sip.sin\_family=AF\_INET;

sip.sin\_port=htons(9999);

sip.sin\_addr.s\_addr=htonl(INADDR\_ANY);

if ( bind(s,(sockaddr\*)&sip,sizeof(sip))==SOCKET\_ERROR)

{

printf("bind() error\n");

closesocket(s);

WSACleanup();

}

char data[254];

int cc=10;

do{

char rd[254];

int n=recvfrom(s,rd,253,0,NULL,NULL);

rd[n]=0;

printf("接受到的时间广播包内容是：%s\n",rd);

cc--;

}while(cc>0);

printf("再见！");

closesocket(s);

WSACleanup();

}

1. 组播程序，每个成员接到消息打出

#include "winsock2.h"

#include "stdio.h"

#include "ws2tcpip.h"

#define BUFLEN 255

void main(int argc, char\*\* argv)

{

struct sockaddr\_in localaddr, fromaddr;

struct ip\_mreq mcaddr;

SOCKET sockfd;

int n, len;

char msg[BUFLEN+ 1 ];

if (argc<3) {

printf("Usage: %s <multicast ip> <port> [<local ip>]\n", argv[0]);

exit(0);

}

WSADATA wsaData;

if ( WSAStartup(0x0202,&wsaData)!=0 )

{

printf("WSAStartup() error!\n");

return ;

}

sockfd = socket(AF\_INET, SOCK\_DGRAM,0);

if (sockfd<0) {

printf("socket creating error \n");

exit(1);

}//if

memset(&localaddr, 0, sizeof(struct sockaddr\_in));

localaddr.sin\_family = AF\_INET;

localaddr.sin\_port = htons(atoi(argv[2]));

if (argc == 4) {

if ((localaddr.sin\_addr.s\_addr=inet\_addr(argv[3]) )==INADDR\_NONE)

{

printf("Wrong local IP address\n");

exit(0);

}

}

else

localaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

if ( bind(sockfd, (sockaddr\*)&localaddr, sizeof(struct sockaddr\_in)) < 0 )

{

printf("bind error \n");

exit(2);

}

memcpy(&mcaddr.imr\_interface, &localaddr.sin\_addr, sizeof(struct in\_addr));

if ((mcaddr.imr\_multiaddr.s\_addr=inet\_addr(argv[1]) )== INADDR\_NONE )

{

printf("Wrong multicast IP address\n");

exit(0);

}

if ( setsockopt(sockfd, IPPROTO\_IP, IP\_ADD\_MEMBERSHIP,(char\*)&mcaddr, sizeof(mcaddr)) == SOCKET\_ERROR )

{

printf("setsockopt error, join multicast failed.code:%d \n",GetLastError());

exit(4);

}

/\* get multicast msg \*/

int cc=10;

do{

len = sizeof(struct sockaddr\_in);

n=recvfrom(sockfd, msg, BUFLEN, 0, (sockaddr\*)&fromaddr, &len);

printf("Datagram from %s %d received\n", inet\_ntoa(fromaddr.sin\_addr),ntohs(fromaddr.sin\_port));

msg[n]=0;

printf("%s", msg);

cc--;

}while(cc>0);

/\* leave multicast group \*/

if ( setsockopt(sockfd, IPPROTO\_IP, IP\_DROP\_MEMBERSHIP,

(char\*)&mcaddr, sizeof(struct ip\_mreq)) < 0)

{

printf("setsockopt error in mtcli.c, drop multicast failed\n");

exit(4);

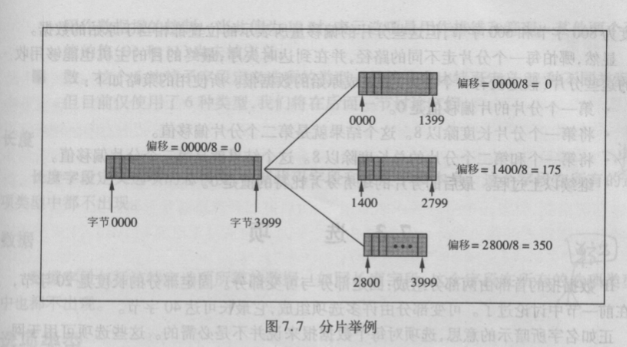
}

closesocket(sockfd);

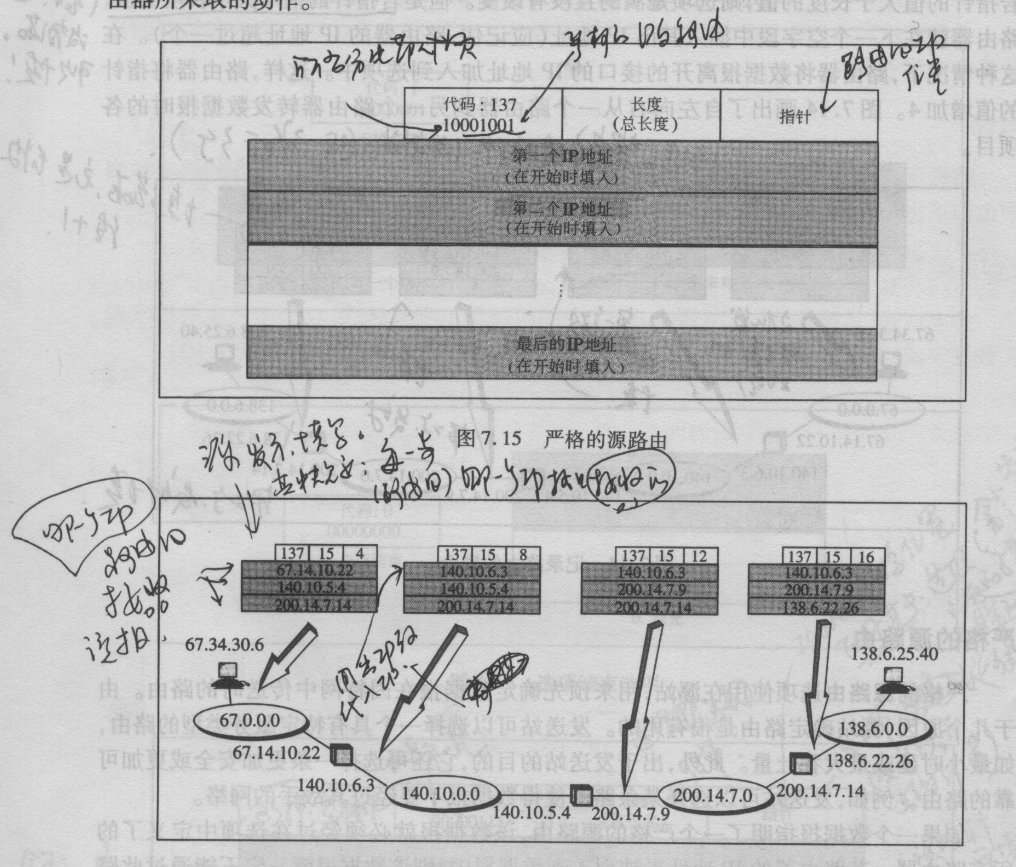
WSACleanup();

}

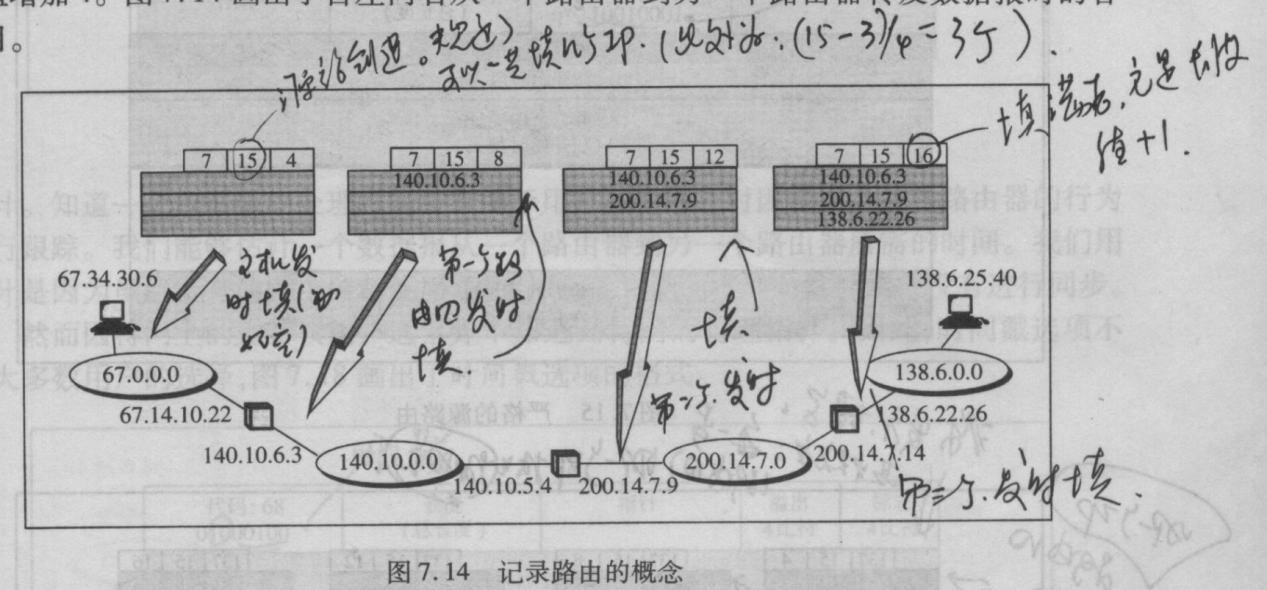
1. ip的分片填图



1. 记录路由填图



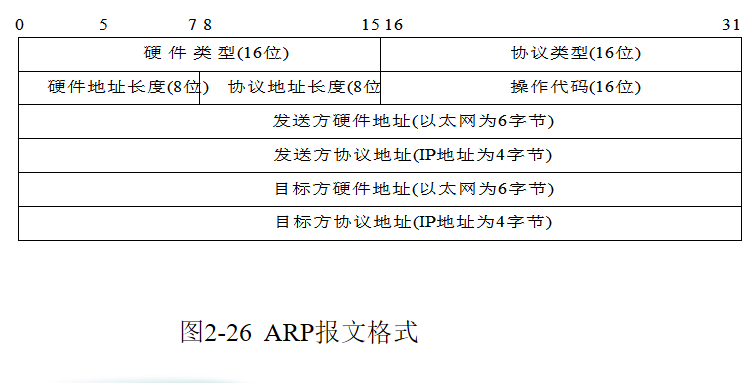
1. 严格的路由填图

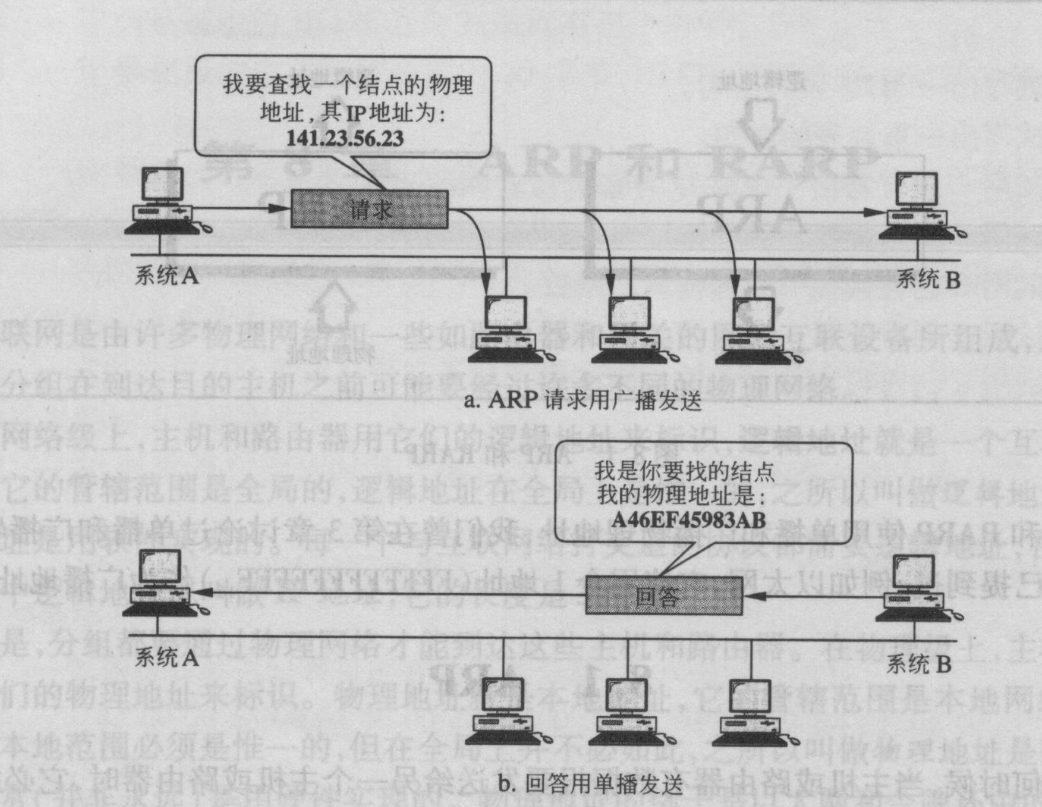


1. icmp找到路径mtu的算法

避免IP分片。源地址将数据包的DF位置，逐渐增大发送的数据报的大小，路径上任何需要将分组进行分片的设备都会将这种数据报丢弃并返回“数据报过大”的ICMP响应到源地址，这样源地址就获得了单次传输的最大传输单元。

1. myping基本考虑
2. ARP的使用简答：请求过程（报文格式），应答报文





1. tcp协议mss、窗口大小、tcp报文大小关系、4个定时器作用

MSS是TCP数据包每次能传输的最大数据分段，其中并不包括TCP首部。而且MSS只出现再syn报文中。窗口大小是个动态的值，传输数据的速率是根据窗口大小来调整的。窗口是为了控制传输过程中的素的。MSS只是控制TCP报文段大小。

四大计时器：

重传计时器：计算TCP报文段的超时重传时间。

坚持计时器：解决零窗口大小通知可能导致的死锁问题。

保活计时器：为了防止两个TCP连接出现长时间空闲，发送探测报文段给客户端。

2MSL计时器：为了确保发送的最后一个ACK报文段能够到达对方，并防止之前与本连接有关的因为延迟等原因而导致已失效的报文被误为位有效。

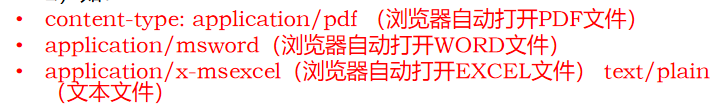
1. http给出所有有状态的方法（4种）

url重写、cookie，session，隐藏域

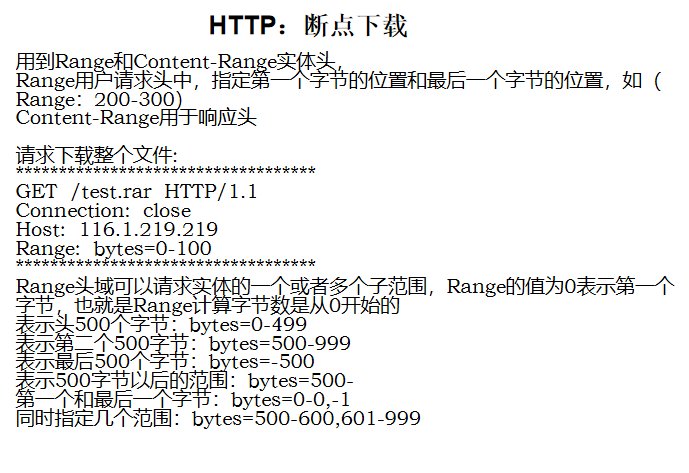
1. 如果cookie不能使用使用哪种方法（session）

session

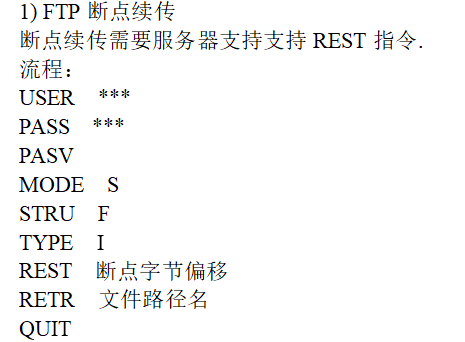
1. 浏览器端自动打开Word、pdf、Excel等



1. http多线程下载文件



1. 简述ftp断续



1. ftp主动、被动（场合）

