分三个大的部分：

1.网络基础

2.Socket编程

3.高并发服务器

1.网络基础

什么是协议：

一组规则（传输和解释的规则）

典型协议：

应用层：http协议，ftp协议、nfs、ssh、telnet

传输层:TCP/UDP

网络层：常见的有ip协议、ICMP、IGMP

链路层：以太网帧协议、ARP

分层结构：

OSI七层模型： 物、数、网、传、会、表、应

TCP/IP 4层模型：网（链路层/网络接口层）、网、传、应

c/s模型：

Client-server

b/s模型：

Browser-server

C/S B/S

优点： 缓存大量数据、协议选择灵活 安全性好、跨平台、开发工作量较小

速度快、迭代更新快。

缺点： 安全性差一些、不能跨平台、 不能缓存大量数据、严格遵守http

开发工作量较大

网络传输流程：

数据在没有封装之前是不能传输的

数据-应用层封装-传输层封装-网络层封装-网络接口层（链路层）封装

整个过程类似于快递一层层封装，传输，然后到集装地点最终到用户一层层解封。

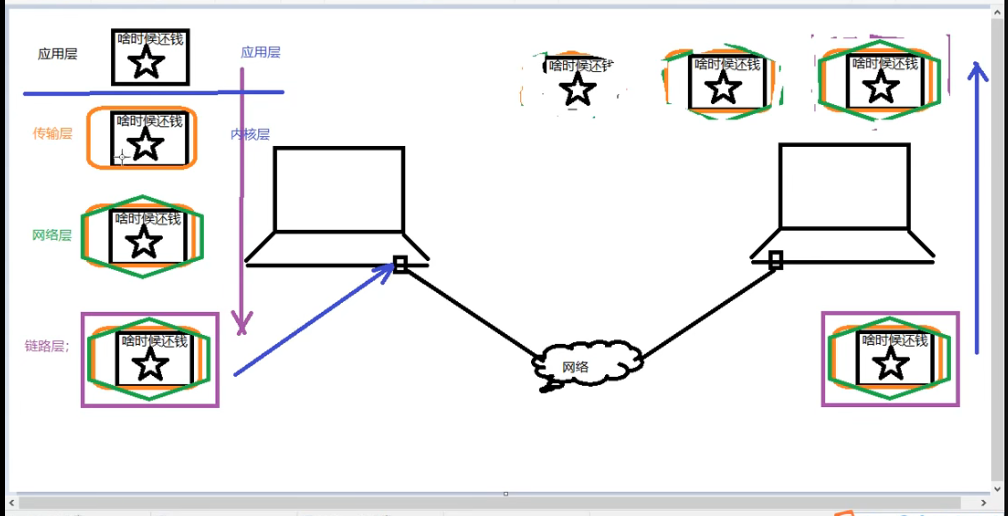
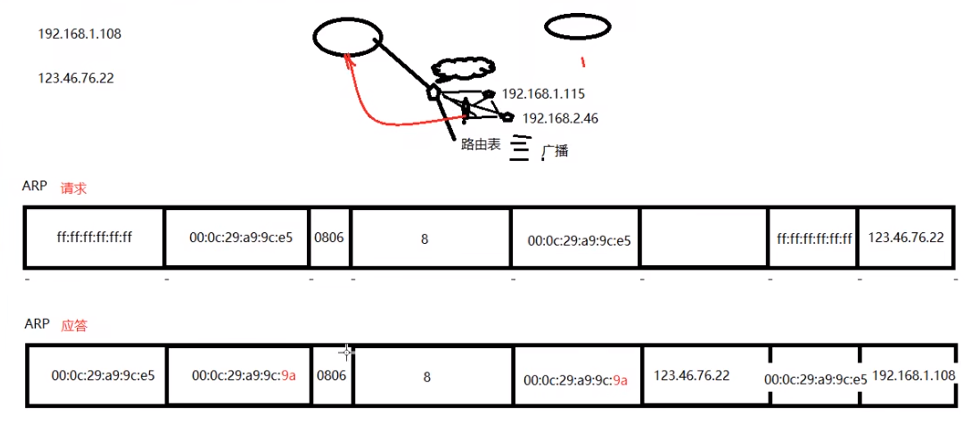


图1-1网络传输流程

以太网帧协议：

ARP协议：根据IP地址获取mac地址

通过向网络接口广播ARP协议，以对端ip为基础，响应的对端IP进行回复得到对端的mac地址。



以太网帧协议：根据mac地址完成数据包传输。

因为以太网帧协议一般需要对端的mac地址，所以和ARP协议一般都是共用得

IP协议：

版本：IPv4、IPv6

TTL：time to live 设置数据包在路由器中的跳转上限，每经过一个路由节点，该值-1，当减为0的时候，有义务将该数据丢弃。

源IP：32位 --- 4字节 192.168.1.108 --- 点分十进制 IP地址（string）

目的IP：32位 --- 4字节

UDP协议：

16位：源端口号 2^16=65536

16位：目的端口号

IP地址：可以在网络环境中，唯一标识一台主机。

端口号：可以在网络的一台主机上，唯一标识一个进程。

IP地址+端口号：可以在网络环境中，唯一标识一个进程

8080端口：http协议使用端口

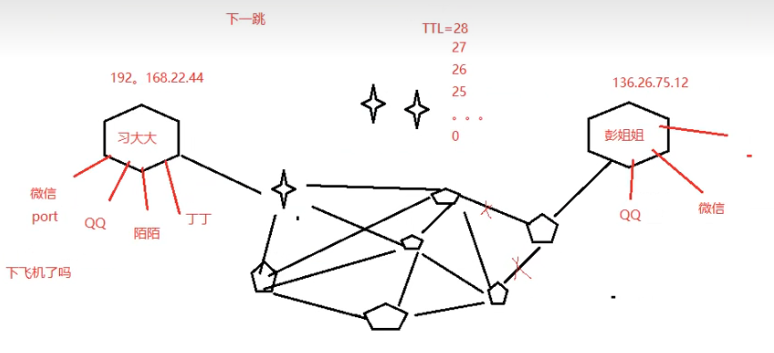


图1-2端口相关图

、

TCP协议：

TCP需要建立链接，UDP不需要建立链接

16位：源端口号。

16位：目的端口号

32序号

32确认序号

6个标志位

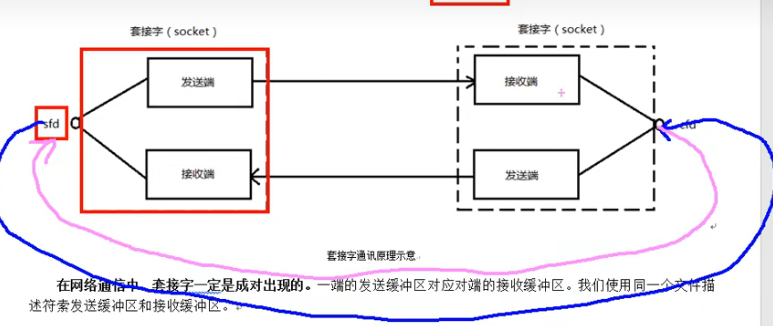
16位窗口大小

**Socket编程**

网络套结字：socket

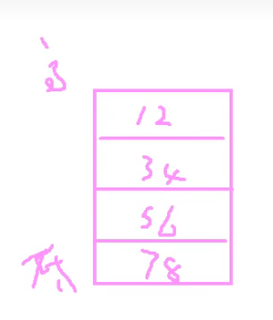
在网络通信过程中，套结字一定是成对存在的

一个文件描述符指向一个套接字（该套接字内部由内核借助两个缓冲区实现）



网络字节序：

小端法：高位高地址，低位低地址。 int a=0x12345678



大端法：高位存低地址，低位存高地址。

计算机采用小端法，网络存储采用大端法，所以需要进行网络字节序和主机字节序的转换。

htonl -->本地-->网络（IP）

htons-->本地-->网络（port）

ntohl-->网络-->本地（IP）

ntohs-->网络--本地（port）

IP地址转换函数：

Int inet\_pton(int af,const char \*src,void \*dest)

af: AF\_INET、AF\_INTE6

src：传入IP地址

dst:传出：网络字节序地址

const char \*inet\_ntop(int af,void \*src,char \*dst,socklen\_t size)

af: AF\_INET、AF\_INTE6

src：网络字节序地址

dst:传出：IP地址(点分十进制)

size:dst的大小

sockaddr数据

sockaddr地址结构

sruc sokaddr\_in

{

sa\_family\_t sin\_family; // AF\_INET

in\_prort\_t sin\_prot; //port in network byte order

struct in\_addr sin\_addr; //internet address

}

struct inaddr

{

Unint32\_t s\_addr; //address in network network byte order

}

使用：

struct sockaddr\_in addr;

addr.sin\_family = AF\_INET/AF\_INET6;

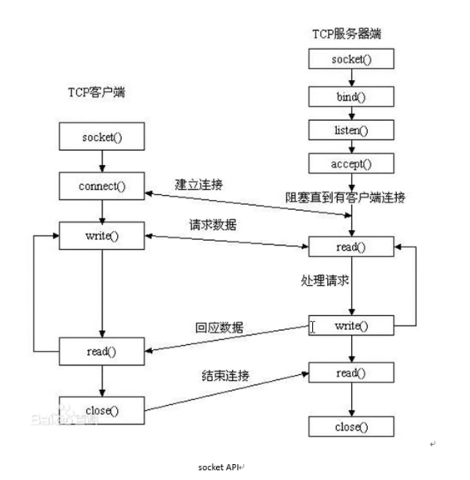
addr.sin\_port = htons(8000);

addr.sin\_addr.s\_addr = inet\_pton(AF\_INET,”192.168.0.1”(void \*)dst);

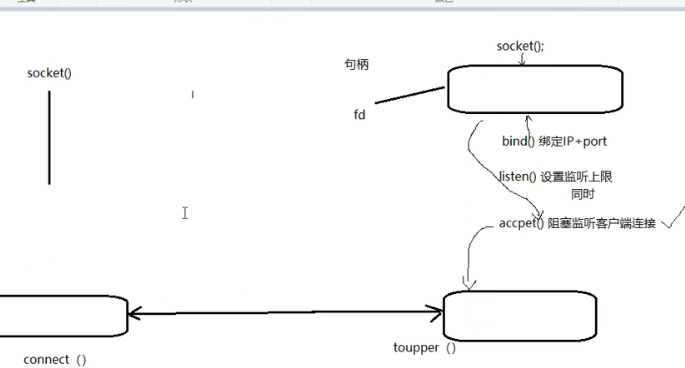
或：addr.sin\_addr.s\_addr = htonl(INADDR\_ANY); //INADDR\_ANY取出系统中有效的 ip地址，为二进制类型

bind(fd,(struct sockaddr \*)&addr,size)

**网络套接字函数**



整个socket模型通信中有三个套接字



通信流程图

socket函数： #include<sys/socket.h>

int socket(int domain,int type,int protocol);//创建一个套接字

domain:AF\_INET,AF\_INET6,AFUNIX

type: SOCKET\_STREAM,SOCK\_DGRAM

protocol：0

返回值：

成功：返回创建套接字的fd

失败 -1 errno

bind函数

#include<arpa/inet.h>

int bind(int sockfd struct sockaddr \* addr socklen\_t addrlen);//给socket 绑定一个ip+端口号

sockfd：socket 函数返回值

struct sockaddr\_in addr;

addr.sin\_family = AF\_INET/AF\_INET6;

addr.sin\_port = htons(8000);

addr.sin\_addr.s\_addr = inet\_pton(AF\_INET,”192.168.0.1”(void \*)dst);

或：addr.sin\_addr.s\_addr = htonl(INADDR\_ANY); //INADDR\_ANY取出系统 中有效的

addr:(struct sockaddr \*)&addr

addrlen:sizeof(addr) //地址结构大小

返回值：

成功：0

失败：-1 errno

listen函数：

int listen(int sockfd,int backlog);//设置与socket同时进行链接的上限数

sockfd:socket 函数返回值

backlog：上限数值，最大值128

返回值：

成功：0

失败：-1 errno

accept函数：

//阻塞等待客户端建立连接，成功的话返回一个与客户端成功建立连接的socket 文件描述

int accept(int sockfd,struct sockaddr \*addr socklen\_t addrlen);

sockfd：socket函数返回值

addr:传出参数，成功与服务器建立连接的客户端的地址结构（IP+port）

addrlen:传入传出参数，入：addr大小，出：客户端addr的实际大小

返回值：

成功：能与服务器进行数据通信的socket对应的文件描述符

失败：-1,erron

connect函数：

//使用现有的socket与服务器建立连接

int connect(int sockfd,const struct sockaddr \*addr,socklen\_t addrlen);

sockfd：socket函数返回值

addr：传入参数，服务器的地址结构

addrlen：服务器的地址结构大小

返回值：

成功：0

失败：-1，erron

如果不使用bind绑定客户端地址结构，采用隐式绑定。

