01 网络基础概念

课程安排

网编编程 5d
- 第一天:网络基础 socket编程
- 第二天:tcp三次握手　高并发服务器(多进程多线程)
- 第三天:tcp转换状态　高并发服务器(select poll epoll)
- 第四天: 高并发服务器(poll epoll) epoll反应堆
- 第五天: 线程池 udp网编编程 本地套接字

libevnet第三方网络库编程 1d
web-server服务器开发 2d

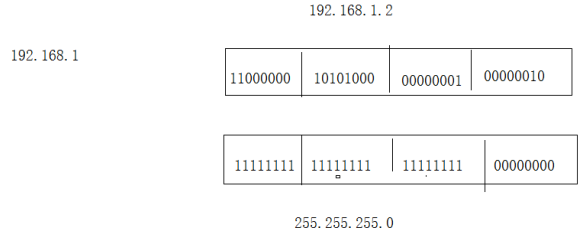
第一天学习内容

1　了解基础网络mac,ip,port
2　了解OSI七层模型
3　了解网络常见协议格式
4　掌握网络字节序和主机字节序之间的转换
5　tcp服务器通信流程
6　tcp客户端通信流程
7　独立写出tcp服务器端代码
8　独立写出tcp客户端代码

1 网卡
网络适配器 :作用 收发数据
mac地址 作用: 用来标识一块网卡, 6个字节 物理地址

```
ens33     Link encap:以太网  硬件地址 00:0c:29:ba:65:04
          inet 地址:192.168.21.30  广播:192.168.21.255  掩码:255.255.255.0
          inet6 地址: fe80::19b0:f7b7:c7e8:c2c9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  跃点数:1
          接收数据包:1959660 错误:0 丢弃:1 过载:0 帧数:0
          发送数据包:2853 错误:0 丢弃:0 过载:0 载波:0
          碰撞:0 发送队列长度:1000
          接收字节:177317197 (177.3 MB)  发送字节:274063 (274.0 KB)
```

2 ip
ip用来标识一台主机 逻辑地址
iPv4 ： ip地址是4字节 32位
ipv6: 128位 16字节
子网id ip中被子网掩码中1连续覆盖的位
主机id ip中被子网掩码中0连续覆盖的位
192.168.1.2/24 192.168.1.2/255.255.255.0

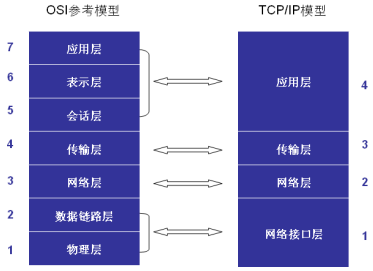


网段地址: 192.168.1.0
广播地址: 192.168.1.255

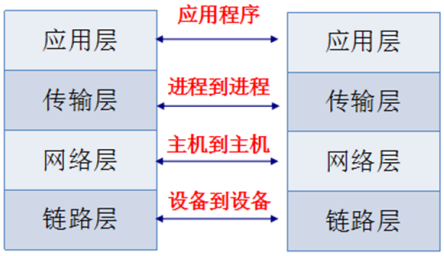
子网掩码 netmask: 用来区分子网id 和主机id

3 端口
作用: 用来标识应用程序(进程)
port: 2个字节 0-65535
0-1023 知名端口
自定义端口 1024 - 65535
netstat

4 OSI 七层模型
物理层: 双绞线接口类型, 光纤的传输速率等等
数据链路层: mac 负责收发数据
网络层: ip 给两台主机提供路径选择
传输层: port 区分数据递送到哪一个应用程序
会话层: 建立建立
表示层: 解码
应用层



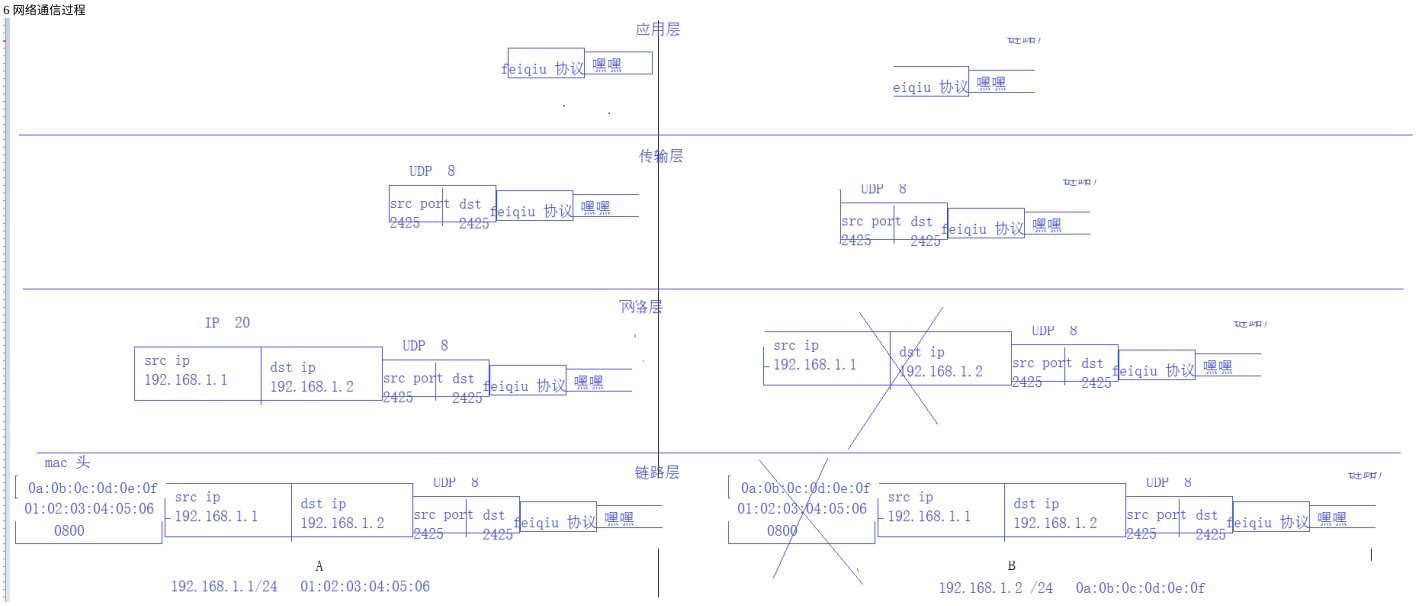
TCP/IP四层模型



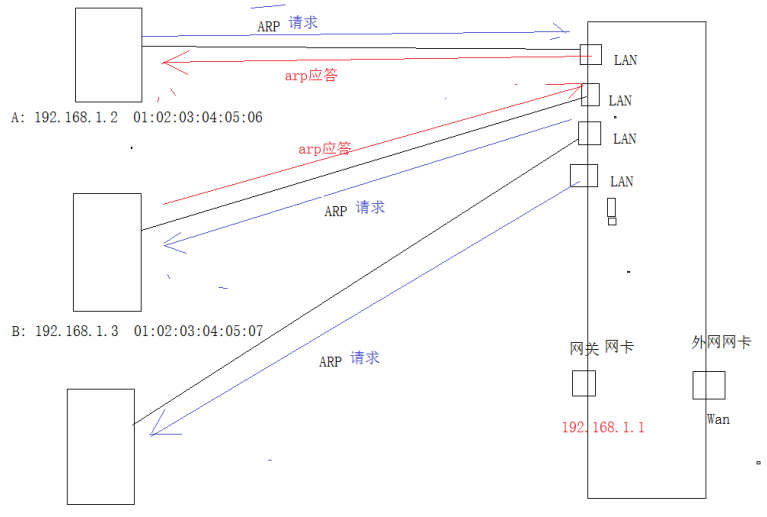
5 协议
规定了数据传输的方式和格式

应用层协议:
FTP: 文件传输协议
HTTP: 超文本传输协议
NFS: 网络文件系统
传输层协议:
TCP: 传输控制协议
UDP: 用户数据报协议
网络层:
IP:英特网互联协议
ICMP: 英特网控制报文协议 ping
IGMP: 英特网组管理协议
链路层协议:
ARP: 地址解析协议 通过ip找mac地址
RARP: 反向地址解析协议 通过mac找ip

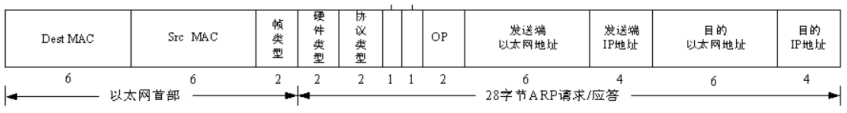
6 网络通信过程



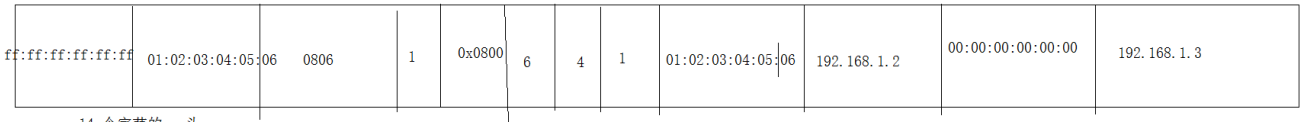
7 arp
地址解析协议: 通过ip找mac地址



arp请求包:



1.Dest



14 个字节的mac头

28字节的arp报文

8 网络设计模式
B/S browser/ server
C/S cilent/server

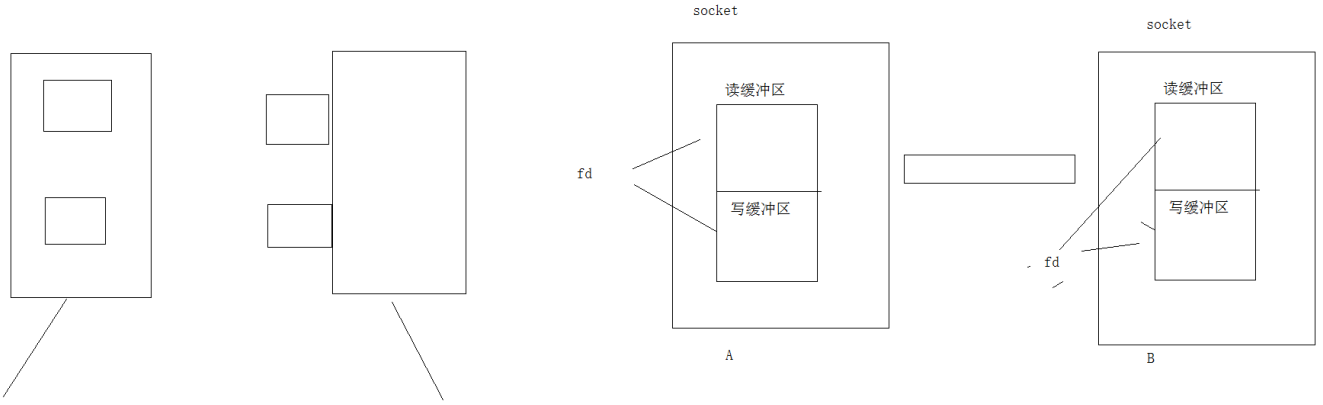
c/s 性能较好 客户端容易篡改数据 开发周期较长
b/s 性能低 客户端安全 开发周期短

9 进程间通信
无名管道
命名管道
mmap
文件
信号
消息队列
共享内存
只能用于 本机的进程间通信

不同的主机间进程通信方法: socket
socket必须成对出现

socket 插座

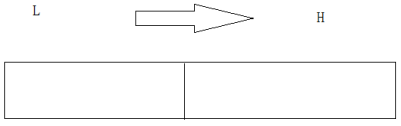
socket 是一个伪文件



10 字节序

大小端

0x0102

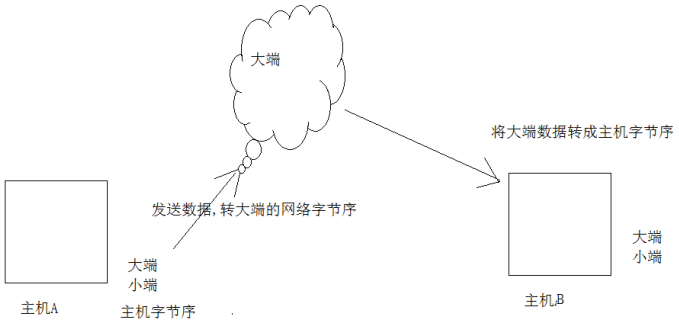


小端: 02 01

低位存低地址, 高位存高地址

大端: 低位存高地址, 高位存低地址

01 02



11 ip转换
#include <arpa/inet.h>
int inet_pton(int af, const char *src, void *dst);
功能: 将点分十进制串 转成32位网络大端的数据("192.168.1.2" ==>)
参数:
af :
AF_INET IPV4
AF_INET6 IPV6
src: 点分十进制串的首地址
dst : 32位网络数据的地址
成功返回1

]
#include <arpa/inet.h>
const char *inet_ntop(int af, const void *src,
char *dst, socklen_t size);
功能: 将32位大端的网络数据转成点分十进制串
参数:
af : AF_INET
src : 32位大端的网络数 地址
dst : 存储点分十进制串 地址
size : 存储点分制串数组的大小

返回值: 存储点分制串数组首地址

12 网络通信解决三大问题
协议
ip
端口

ipv4套接字结构体
```
struct sockaddr_in {
        sa_family_t    sin_family; /* address family: AF_INET */
        in_port_t      sin_port;   /* port in network byte order */
        struct in_addr sin_addr;   /* internet address */
    };

    /* Internet address. */
    struct in_addr {
        uint32_t       s_addr;     /* address in network byte order */
    };
```

sin_family: 协议 AF_INET
sin_portL端口
sin_addr ip地址

ipv6套接字结构体

```
struct sockaddr_in6 {
    unsigned short int sin6_family;       /* AF_INET6 */
    __be16 sin6_port;                   /* Transport layer port # */
    __be32 sin6_flowinfo;               /* IPv6 flow information */
    struct in6_addr sin6_addr;          /* IPv6 address */
    __u32 sin6_scope_id;                /* scope id (new in RFC2553) */
};

struct in6_addr {
    union {
        __u8 u6_addr8[16];
        __be16 u6_addr16[8];
        __be32 u6_addr32[4];
    } in6_u;
    #define s6_addr         in6_u.u6_addr8
    #define s6_addr16       in6_u.u6_addr16
    #define s6_addr32       in6_u.u6_addr32
};

#define UNIX_PATH_MAX 108
    struct sockaddr_un {
    __kernel_sa_family_t sun_family;    /* AF_UNIX */
    char sun_path[UNIX_PATH_MAX]; /* pathname */
};
```

通用套接字结构体

```
struct sockaddr {
    sa_family_t sa_family;     /* address family, AF_xxx */
    char sa_data[14];          /* 14 bytes of protocol address */
};
```

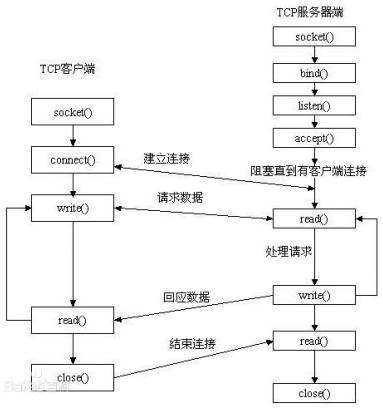
13 tcp
传输控制协议
特点: 出错重传 每次发送数据对方都会回ACK, 可靠
tcp是抽象打电话的模型
建立练级 使用连接 关闭连接

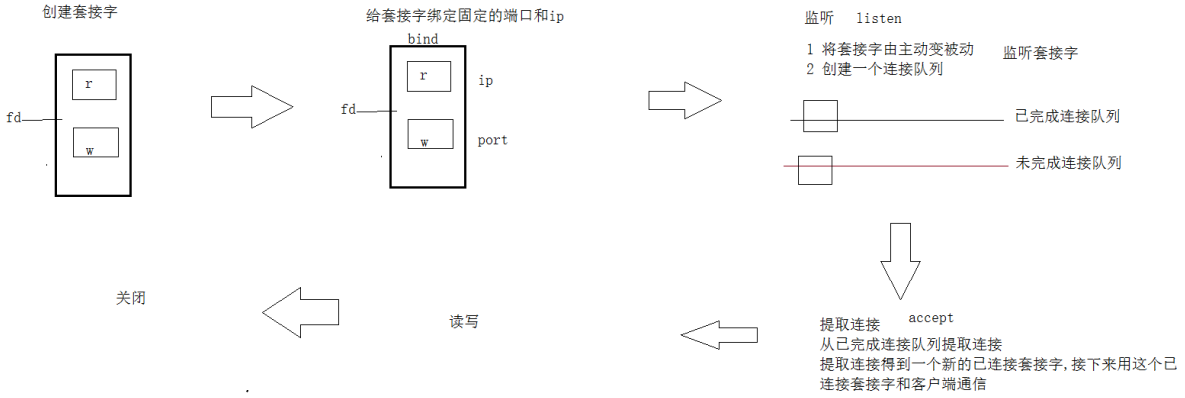
14 创建套接字API
```
#include <sys/socket.h>
int socket(int domain, int type, int protocol);
```
创建套接字
参数:
domain:AF_INET
type: SOCK_STREAM 流式套接字 用于tcp通信
protocol: 0
成功返回文件描述符,失败返回-1

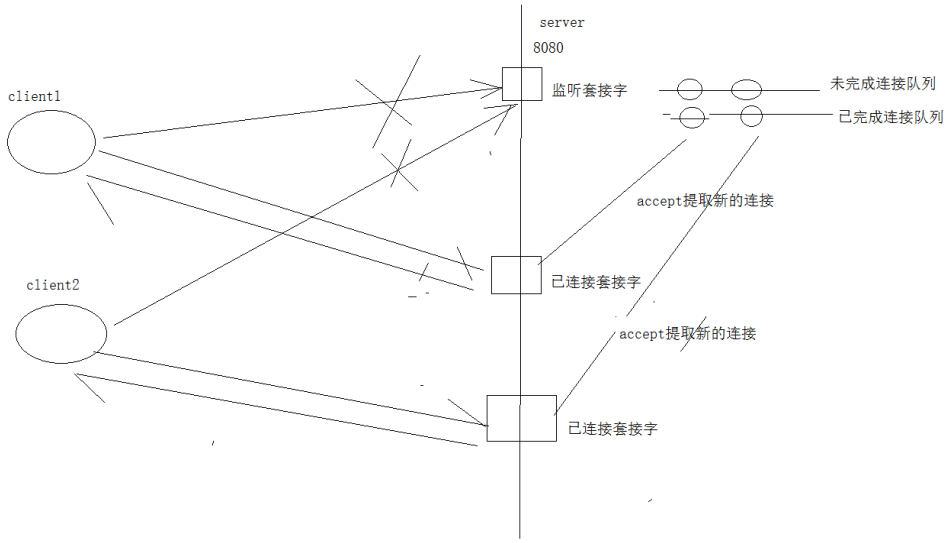
15 连接服务器
```
#include <sys/socket.h>
 int connect(intsockfd , const struct sockaddr *addr,
                socklen_t addrlen);
```
功能: 连接服务器

sockfd: socket套接字
addr: ipv4套接字结构体的地址
addrlen: ipv4套接字结构体的长度

16 tcp服务器通信流程







17 bind绑定
给套接字绑定固定的端口和ip
```
#include <sys/socket.h>
 int bind(int sockfd, const struct sockaddr *addr,
          socklen_t addrlen);
```

sockfd: 套接字
addr: ipv4套接字结构体地址
addrlen: ipv4套接字结构体的大小
返回值:
成功返回0 失败返回;-1

18 listen
```
#include <sys/socket.h>
int listen(int sockfd, int backlog);
```
参数:
sockfd : 套接字
backlog : 已完成连接队列和未完成连接队里数之和的最大值 128

19 accept
```
#include <sys/socket.h>
 int accept(int socket, struct sockaddr *restrict address,
        socklen_t *restrict address_len);
```
如果连接队列没有新的连接,accept会阻塞

功能: 从已完成连接队列提取新的连接
参数:
socket : 套接字
address : 获取的客户端的的ip和端口信息 iPv4套接字结构体地址
address_len: iPv4套接字结构体的大小的地址
```
socklen_t len = sizeof(struct sockaddr );
```

返回值: 新的已连接套接字的文件描述符

20 tcp服务器通信步骤
1 创建套接字 socket
2 绑定 bind
3 监听 listen
4 提取 accept
5 读写
6 关闭