计算机网络

1.通信的两个元素

通信双方的地址：

ip

端口号

ip加端口是一个应用

规则：网络通信协议

2.七层参考协议

3 ip地址

127.0.0.1 本机的ip

一个ip地址是一个电话号码

dos命令：

```
ipconfig
```

用来查看网络关卡什么的一些东西

ping命令：

```
ping 127.0.0.1
```

用来连接一个计算机的网络

ping得来就是互相连接 不来就不是

ip是唯一确定标识计算机的东西

localhost

IP的分类：

a.ipv4/ipv6

ipv6 8个无符号整数

ipv4 4个字节

b.公网和私网

4.端口号

端口号是计算机里面唯一标识进程应用程序得一个编号

通过端口号我们可以确定一个进程，但是注意 端口号是逻辑得 我们是看不到的

每次一个程序进程打开 ， 就会随机分配一个端口号

端口号 2个字节组成

0，65535

注意事项

1024之前被系统分配 我们不可以使用；

端口号不重复

5.tcp通信协议

两端通信的步骤：

例子：我们用qq来聊天

客户端：我们

服务端：腾讯 提供qq聊天服务

步骤：

1.启动服务端，等待客户端连接

2.客户端主动连接服务器，连接成功才可以

这个连接会建立一个对象 这个对象就是io对象

客户端和服务器端就可以用io对象通信

通信的数据不仅仅是字符 于是乎 io对象是字节流对象

客户端：配置比较低的电脑 ip 端口号 socket

服务端：配置比较高的电脑 ip 端口号 serversocket

过程：

一次交互数据需要4个io对象

服务器必须明确2件事情：

1.多个客户端可以同时和服务器进行交互

要知道和那个客户端交互

服务器端有一个accept可以获取到请求服务的对象

2.要多个io流对象

服务器是没有io流的，服务器可以获取到请求的对象 socket 使用每个客户端的socket 的io流和服务器进行交互

服务器使用客户端的字节输入流读取数据

服务器使用客户端的字节输出流回写数据

```
socket S1=server.accpet()
```

6.网络通信协议

通过计算机互相链接需要遵守的交通规则，在计算机中这些协议的通信规则被称为网络协议。

对数据的格式 速率等做出相应的规定

tcp/ip协议：internet最基本 最广泛的协议，数据之间如何传输的协议，定义了计算机如何链接因特网 数据之间如何传输。

有四层分层模式。

udp 网络的数据报协议 面向无连接的一种协议

数据发送端和数据接受端 不用建立连接（不用建立链接 不用回复）消耗资源特别小 通信效率高 一般用于音频和视频的数据传输，udp协议

数据限制在64kb超过这个范围就无法传送了

tcp协议 ： 先在传送端和接受端 先建立链接，再进行发送数据，发送了可靠的无差错的数据传输

再tcp里面有一个客户端和一个服务器，发送链接请求，每次链接创建都需要三次握手协议

1.一次握手：发送请求服务，等待服务端确认

2.二次握手：服务端回传一个响应，通知客户端收到了连接请求

3.三次握手：客户端再次向服务端发送确认消息 ， 发送连接

7.tcp通信的代码：

客户端：

java.net.socket

套接字（包含了网络ip地址和端口号的网络基本单位）

构造方法：

socket(服务器的名字或者ip地址,端口号);

成员方法：

getoutputstream获得字节输出流

getinputstream获得字节输入流

close()关闭套接字

实现步骤：

1.创建一个客户端的对象，绑定一个服务端的ip地址和端口号

2.用socket getoutputstream获得网络输出流对象

3.使用outpitstream的writer，给服务器发数据

4.使用iputstream获取网络字节输入流对象

5.用iputstream的read读取网络服务器回写的数据

6.释放资源

注意：

a.客户端和服务端进行交互要使用本身提供的流对象 不能是自己的流对象

b.我们建立客户端socket 就会请求服务器

如果服务器没有启动就会抛出异常

客户端代码；

```
package lll;

import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;
import java.net.UnknownHostException;

public class tcpclient {
 public static void main(String args[]) throws UnknownHostException, IOException
 {
     Socket socket=new Socket("127.0.0.1",8888);//建立一个客户端

    OutputStream outputstream= socket.getOutputStream();//字节输出流
    outputstream.write("你好服务器".getBytes());//发送数据 变为字节流
     InputStream is=socket.getInputStream();
    byte[] buffer=new byte[1024];
    int len=is.read(buffer);
    System.out.print(new String(buffer,0,len));
     /*以上都是读取我们的数据*/
    socket.close();//关闭流
 }
}
```

服务器：

1构造方法：

serversocket(端口号)

使用我们的accpet来获取到请求的对象socket

步骤：

1.创建一个服务器seversocket和一个系统指定的端口号

2.使用seversocket里面的accpet获取客户端的socket

3.使用outpitstream的writer，给客户端发数据

4.使用iputstream获取网络字节输入流对象

5.网络字节输出流 write回写数据

6.关闭数据流

代码：

```
package lll;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class tcpserver {
 public static void main(String args[]) throws IOException
 {

    ServerSocket serversocket=new ServerSocket(8888);
    Socket socket=serversocket.accept();
    InputStream is=socket.getInputStream();
    byte[] buffer=new byte[1024];
    int len=is.read(buffer);
    System.out.print(new String(buffer,0,len));
    OutputStream os=socket.getOutputStream();
    os.write("收到谢谢".getBytes());
    serversocket.close();
    socket.close();
 }
}
```

8.文件上传的案例：

原理：客户端上传一个文件到服务器里面去 服务器保存到服务器的硬盘里面去

输入流是读入 输出流是上传

1.客户端使用本地的字节输入流，读取要上传的文件

2.客户端 使用网络字节输出流 把文件上传到服务器

3.服务器使用网络字节输入流读取客户端上传的文件

4.服务器使用本地字节输出流 把文件保存到服务前硬盘

5.服务使用网络字节输出流 给客户端 回写一个请求成功

6.客户端使用网络字节输入流 读取请求

注意：

和本地互动 用本地字节流 本地流

和网络互动 用网络字节流 网络流

9.文件上传代码

客户端代码：

```
package com.deu.sz;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;

public class client {
  public static  void main(String args[]) throws Exception
  {
     FileInputStream fis=new  FileInputStream("c:\\1.jpg");/*读取本地对象*/
     Socket socket=new Socket("127.0.0.1",8888);/*建立一个socket*/
     OutputStream os=socket.getOutputStream();/*获取网络字节输出流对象*/
     int len=0;
     byte[]bytes=new byte[1024];//建立一个数组好计算
     while((len=fis.read(bytes))!=-1)//当读取的长度没有到结尾的时候不断写出去
     {
         os.write(bytes,0,len);//输出我们的数据
     }
     InputStream is=socket.getInputStream();//获得一个输入流来标识服务器返回的

     while((len=is.read(bytes))!=-1)//
     {
         System.out.print(new String(bytes,0,len));
     }
     socket.close();
     fis.close();
  }
}
```

服务器代码：

```
package com.deu.sz;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class server {
 public static void main(String args[]) throws Exception
 {
     ServerSocket server=new ServerSocket(8888);//建立一个服务器socket
     Socket socket=server.accept();//建立一个socket 来获取我们的请求
     InputStream is= socket.getInputStream();//拿一个字符输入流来获得我们字节输入
     File file=new File("d:\\upload");//打开一个新的文件夹
     if(!file.exists())
     {
```

```
            file.mkdir();//如果不存在的话 我们就建立一个文件夹来保存
        }
        FileOutputStream fos=new FileOutputStream(file+"//1.jpg");//存入我们的数据
        int len=0;
        byte[]bytes=new byte[1024];//新开一个数组来保存我们的数据
        while((len=is.read(bytes))!=-1)//如果我们读取的网络输入流长度没有结束的话 一直写入我们的文件中去
        {
            fos.write(bytes,0,len);
        }
        socket.getOutputStream().write("上传成功".getBytes());//返回一个上传成功
        socket.close();//关闭资源
        fos.close();
        server.close();

    }
}
```

10.优化文件

1.防止文件重复名字被覆盖

方法:

域名+毫秒值+随机数

2.让服务器一直处于监听状态 不让服务器进入死循环

while(true)

11.模拟bs服务器

先写好服务器的代码:

```
package com.deu.sz;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class server {
 public static void main(String args[]) throws Exception
 {
    ServerSocket server=new ServerSocket(9760);
    Socket socket=server.accept();
    InputStream is=socket.getInputStream();
    byte[] bytes=new byte[1024];
    int len=0;
    while((len=is.read(bytes))!=-1)
    {
        System.out.print(new String(bytes,0,len));
```

```
    }
  }
}
```

写好服务器端的代码：

先启动服务器

获得信息

服务器给客户端会写一个页面

回写的时候需要知道文件的地址

这个地址 就是第一个开头

请求消息第一行

我们可以使用http加ip加上端口号来访问我们的资源

```
http://127.0.0.1:8000/src
```

12.如何聊天室

一个人向服务器发送我们的消息 服务器帮我们转发给另一个人

另一个人接受消息给服务器 服务器又转发

websocket是实现浏览器和服务器之间全双工的协议

持久连接协议

13.websocket实现与使用

建立一个websocket

语法：

```
var socket=new WebSocket();
```

websocket方法：

计算机网络上到下

物理层

1.数字的还是模拟的

数字的好处：

更好实现

表示

模拟的好处：和世界相符合

缺点：专业 不通用 电路复杂

公式：

波长：

$$r = \frac{c}{f}/c = c * T$$

复合信号就是很多的信号复合在一起

具有离散的数值

模拟信号的变量：带宽

复合波的之间的频率范围就叫做带宽。

是两个数值的差值。

复合的频率是连续的

数字信号的变量：比特

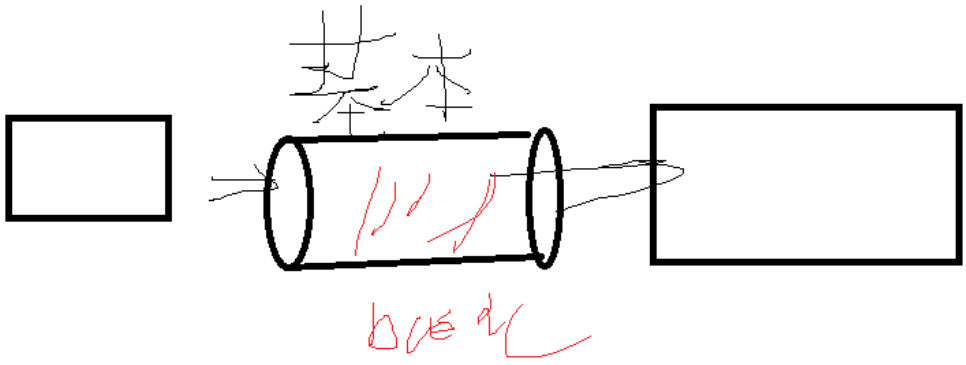
一秒发送了多少的比特就是数据的比特率

电平和数字的关系：v（电平数)=log2v个数字

数字信号是一种模拟的信号。

位长：速度*time

基带传输：通道传输数字信号。



宽带传输：

数字变成模拟

传输的损耗：

1.衰减

分贝用来测量不同的强度

负值（表示衰减）

正值 （表示增加）

$$db = 10 * \lg \frac{p2}{p1}$$

p1和p2是位置1和位置2

2.失真

3.噪声

噪声的影响量化：信噪比（snr）

$$snr = single/nnnn$$

数据的速率

a.带宽b.电平c.通道质量

nice准则

$$比特率 = w * 2 * \log_2 L$$

理论上增加l信号量就可以提高 但是越多的l 越难以分辨

香农：

$$c = w * log_2(1 + S/N)$$

性能

hz表示范围

s表示数据

吞吐量是实际的数据<带宽数据

$$延迟：传博 + 传送 + 排队 + 处理$$

带宽*延迟表示一个链路最多充满的字节数。

管道的容量就是包含的数据最多的位数

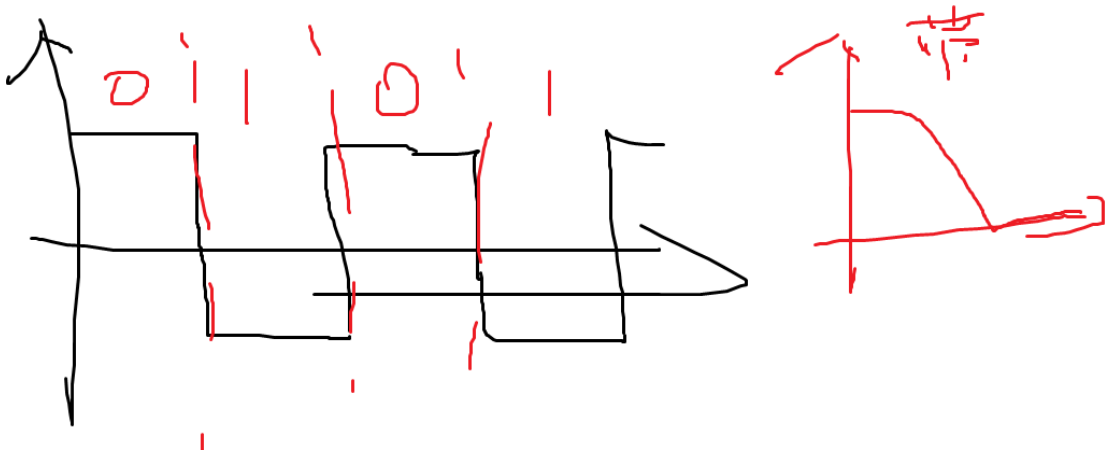
数字到数字

线路编码：数字数据变为数字信号

$$s_{ave} = c * N * \frac{1}{r}$$

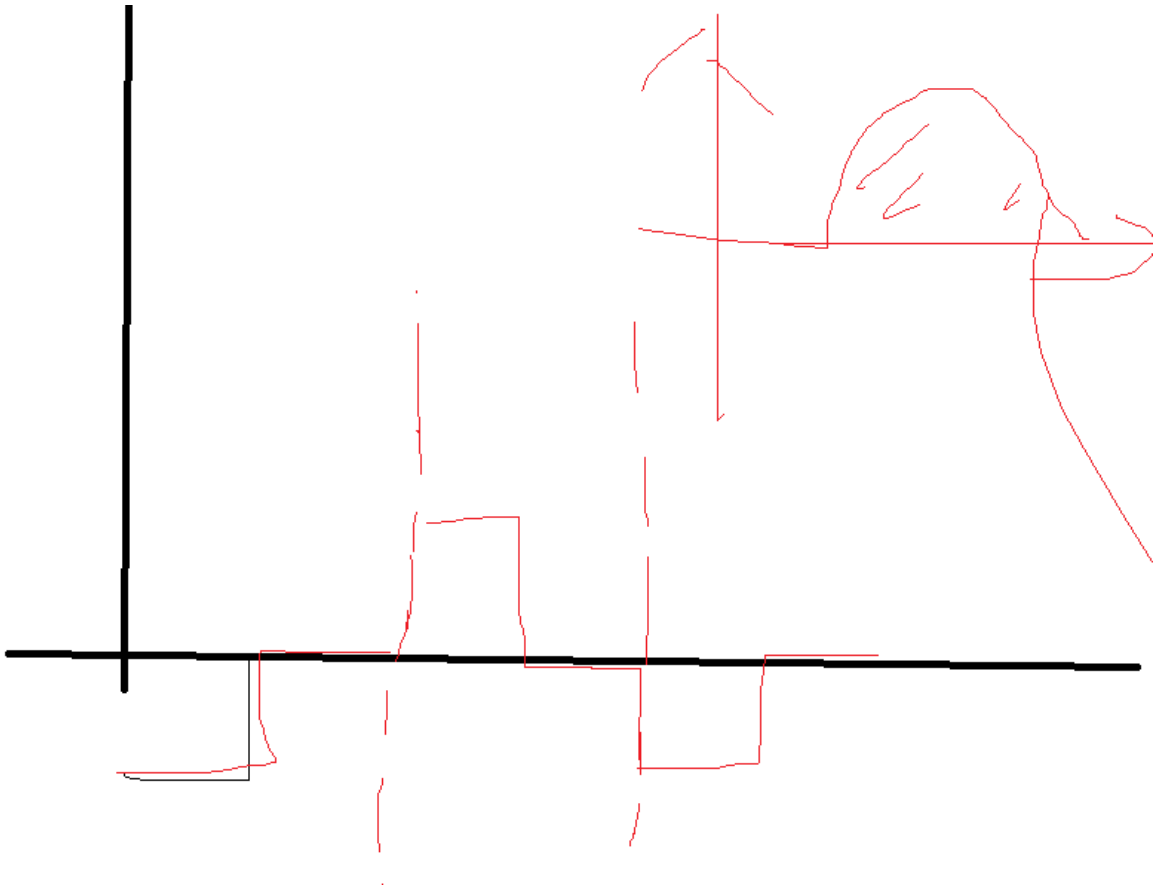
编码方案：

归零码和不归零码



这个里面就是需要一个同步的时钟 利用这个时钟来同步我们的信息 知道数据从哪里开始哪里结束

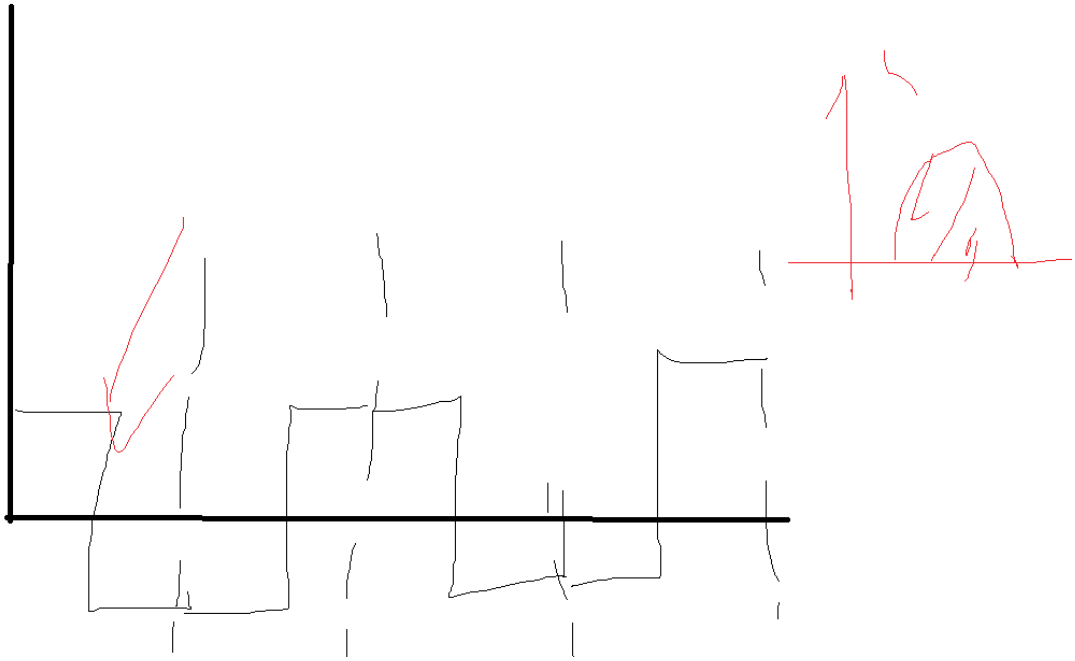
归零编码：



不归零码的缺点就是不能知道下一步在哪里开始 需要有个同步的时钟

归零码的有点就是归零 可以减少误码率 这样一个位需要2个电平 带宽也就变大了

综合的方法：曼切斯特编码



这个就是曼切斯特编码

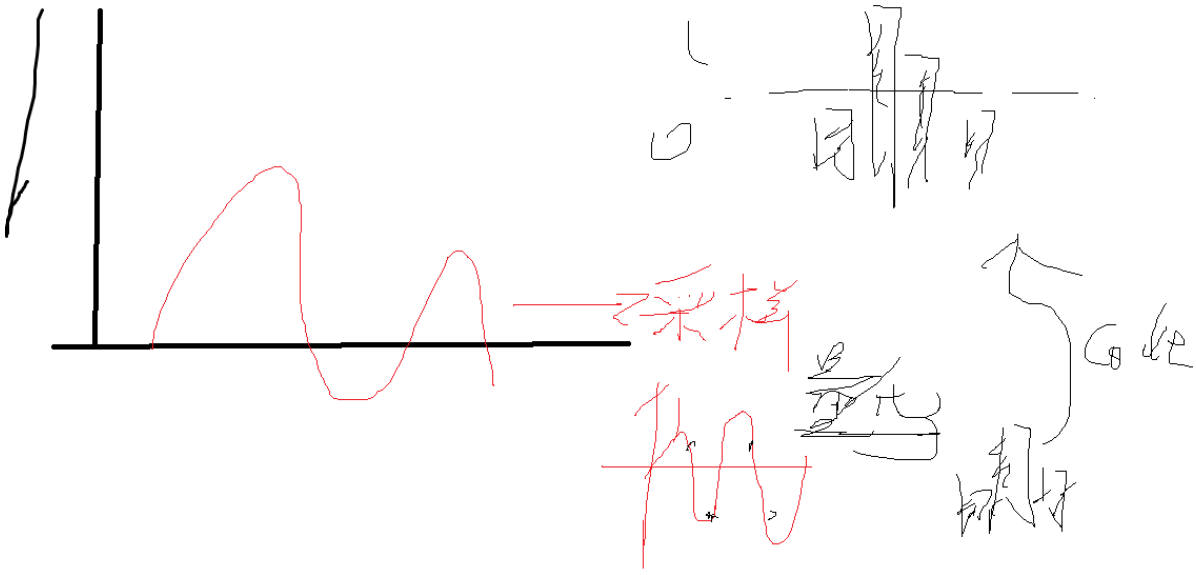
他把同步的时钟也带入了进去

这样我们在传数数据的时候就把时钟也带了过去 只用看这个跳变就是时钟的变化

4b/5b就是利用冗余来实现差错

模拟到数字

1.采样2.量化3.编码



采样：采样频率：

$$f_{采样} = 2 * f_{high}$$

带宽有限才进行采样

数字到模拟

1.幅移模拟

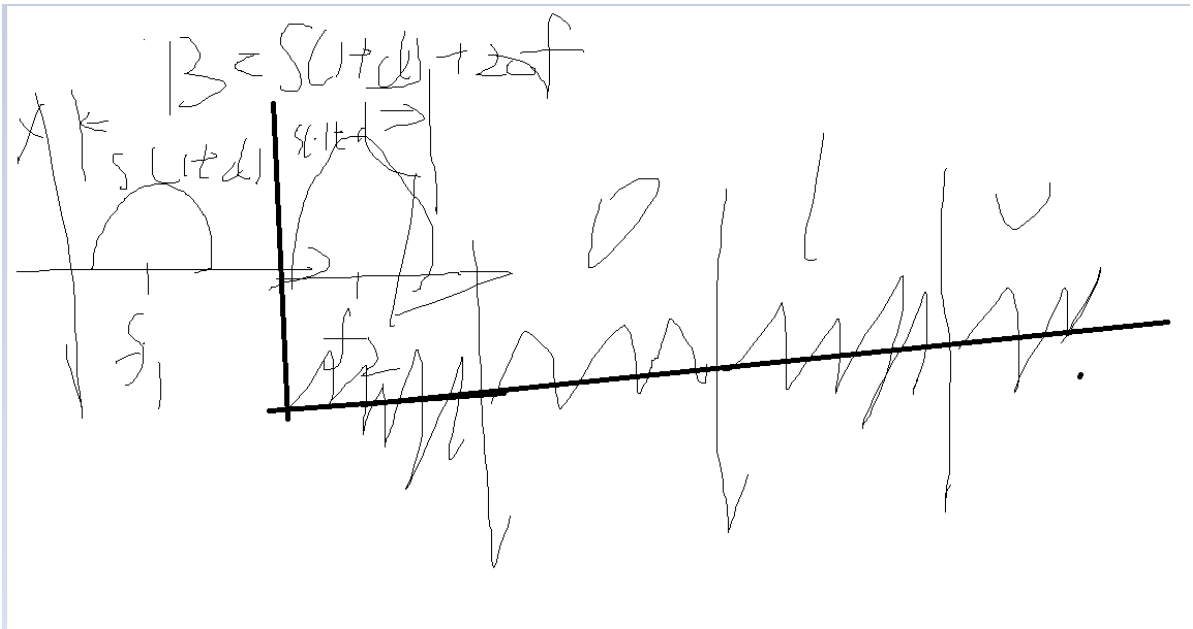
这里的带宽看作为

$$B = (1 + D) * S$$

最大就是2*s

载波频率在中间 就是可以改变带宽来匹配不同的数率

2.频移模拟



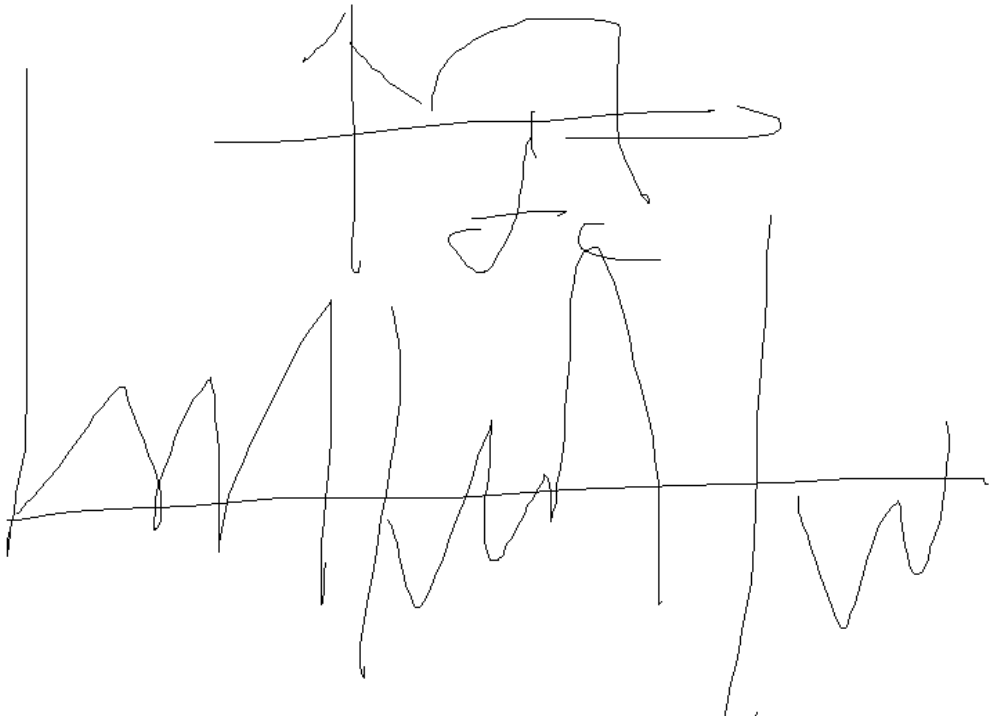
2个频率·之间的差值就是

$$f2 - f1 = 2 * f_{相对}$$

于是

$$b = S(1 + d) * S + 2 * f_{变化}$$

3.相移模拟

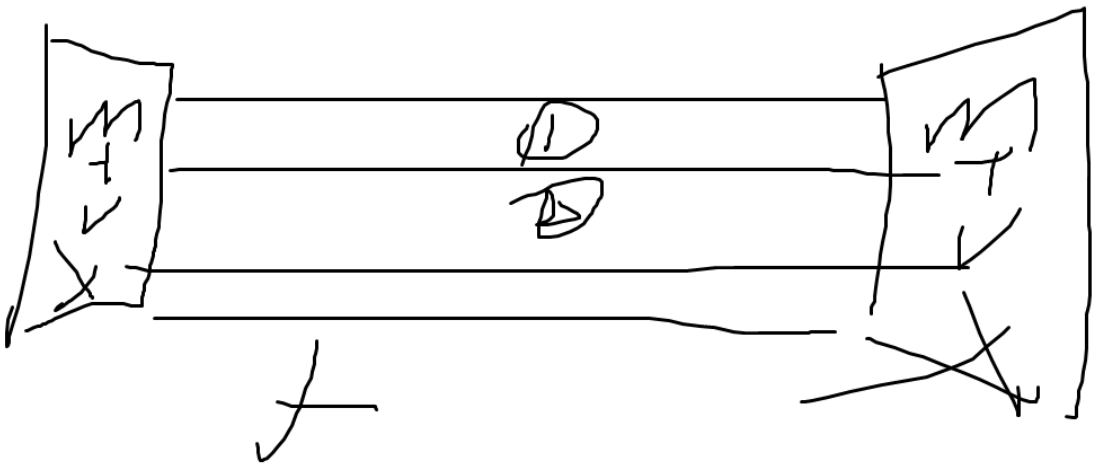


宽带的利用：

频分多用（FDM)

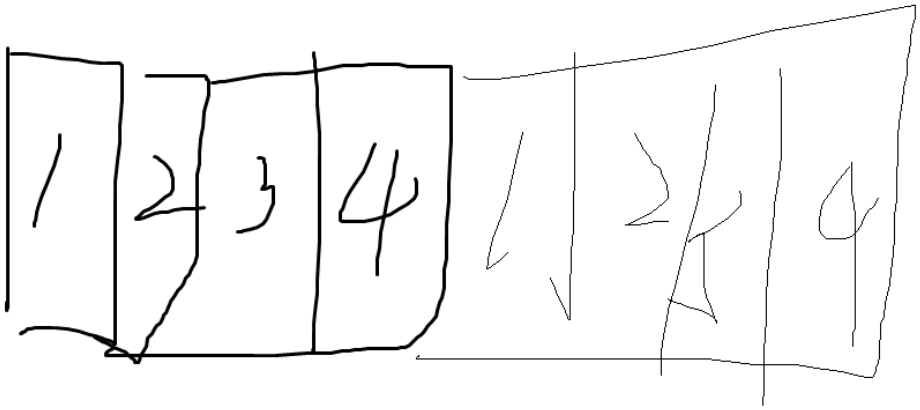
在不同的频率段来进行传输

中间做隔离带



时间上时并发的

时分多路（tdm):



每个时间片都进行一次传输

这样的话就是每个单位的持续时间就是、

$$\frac{t}{n}$$

输出速率时当个输入速率的n倍

物理设备：中继器（集线器）

在数据破坏之前 重生 恢复数据

发送新生成的数据

除了信号的接口 其他的端口都被广播出去

没有过滤能力

数据链路层：（有线网络）

下载高于上传

dlc(数据链路控制)成数据帧，流量控制，差错控制

面对字符 间隔传就可以了

面对位的数据帧：

成数据帧

连续五个1加一个0

01111110 表示标记 为了防止中途影响了于是不让出现连续的6个1 就连续5个1加个0

接受的时候连续5个1后面的0要去掉

流量控制：

在接收前 确定能够接受的数据

点到点

差错控制：

传输的时候会出现不同的变化

这个就是出现了传输的错误

冗余

利用冗余来处理被破坏的位

差错和纠错

差错easy 纠错难

code:

添加冗余位来改变

块编码：

k位时数据字

增加一个r位当作冗余码

$$n = k + r$$

叫做代码字

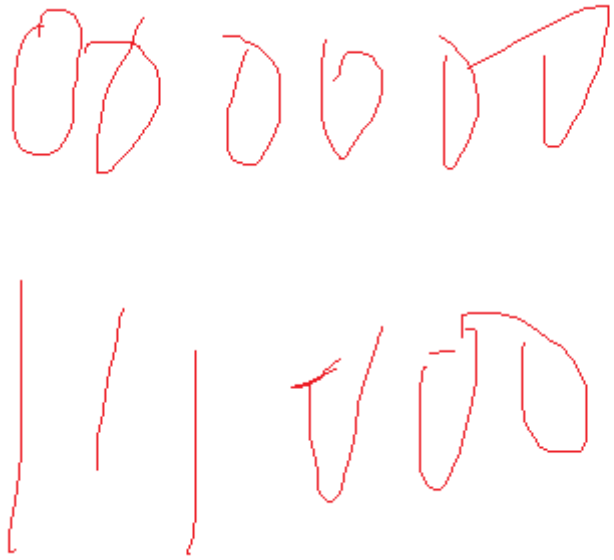
$$2^n > 2^k$$

多出来的有许多没有使用的 就是实现欺骗的原理

只可以检查它编码那种的类型差错 不能检查其他类型

汉明距离：

两个码字不同的数值位置



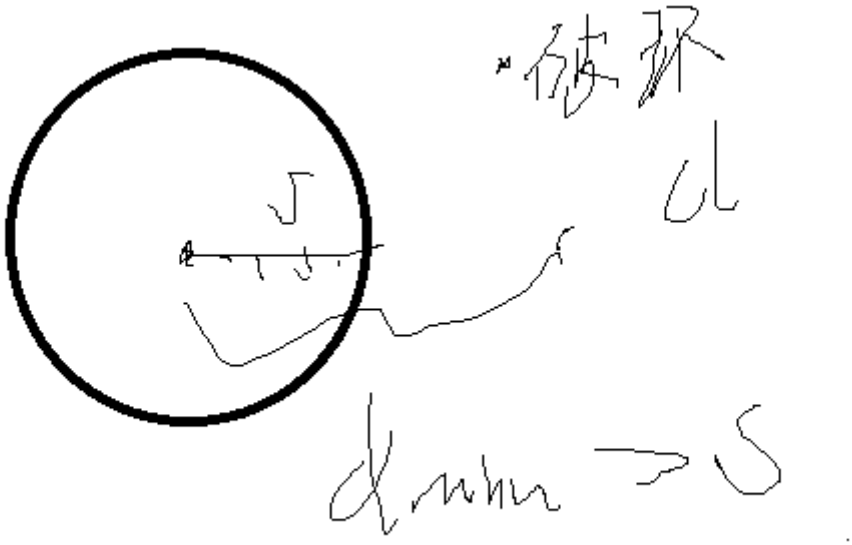
这个汉明距离就是3

两个数值异或就是汉明距离

111111^001111=110000

汉明距离大于等于0

检查s个错误 我们需要至少s+1的汉明距离



需要纠正s个差错 我们就需要s+1

我们需要检查圆心以及边界上的点

发生s个差错都是圆周上面的点

园心是代码字

有差错的点都在圆内

那么代有效的码字都在园外

要检查园内出错的码字一个是半径如何加上边界上的点

例子：

00	**000**
01	011
10	101
11	110

这个的汉明距离最小汉明距离 就是2

检查1个差错

代码字最小的距离就是最小汉明距离

如果出现一个位差错

001 这不是一个代码字 于是丢弃他

000就是一个代码字 接受他

最小汉明距离：

最小具有1的个数的非0有效代码字

上诉就是

$$d_{min} = 2$$

最常见编码：奇偶校验

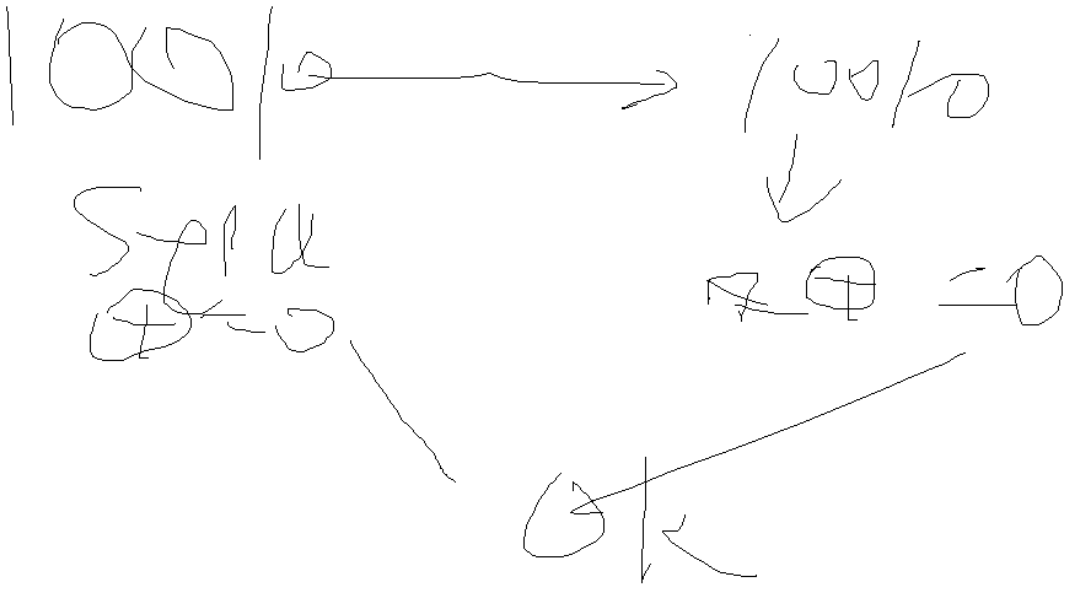
就是增加一个位让1是奇数还是偶数

00 00	**00000**
0001	00011
0010	00101
0011	00110

这个里面的最小汉明距离就是2

这个最小的只能检查1位的汉明码

检查就是每个位异或起来 如果是奇数个就是1 偶数个就是0



这样是匹配上的

如歌匹配不上的话就是

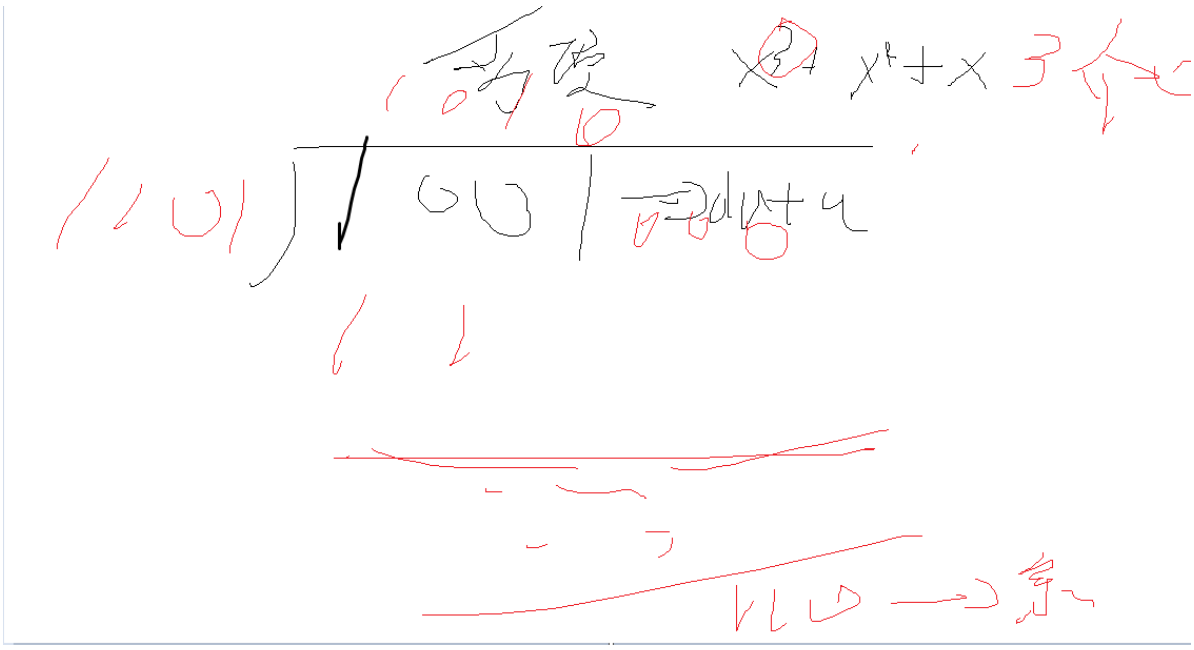
这个就是匹配不上的

于是奇偶校验码只能校验奇数个错误

循环检查

就是一个多项式的除法

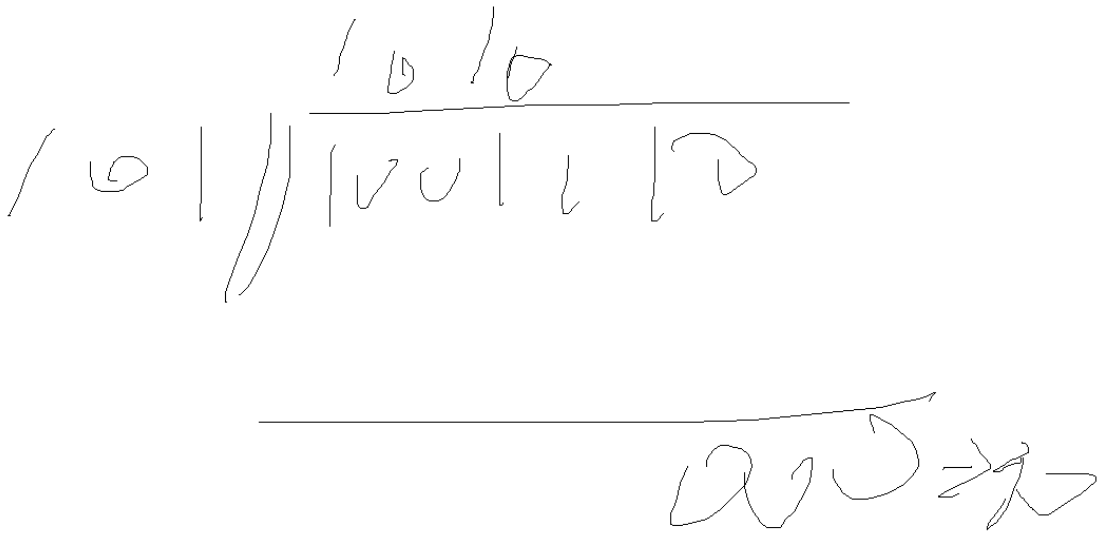
发送方加上校验子



这个最后发送的时候就是发送加上余数的序列

1001110

检查的时候和校验子做方法



该模式最少2位

左边右边都是1

最小的多项手机校验子式11

性质：

1。检查个数的差错

2奇数个的话 就是被11除

3.突发错误：l是突发错误长度 有几个位置错了

r是余数的长度 l<=r 就是所有都可以检查

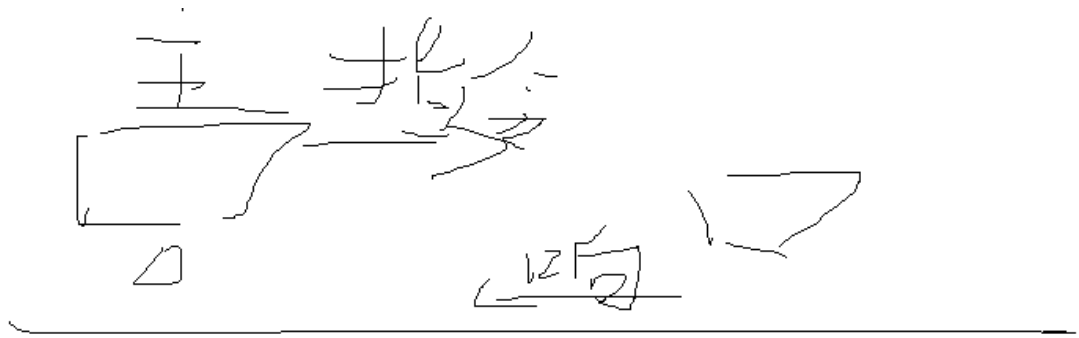
l=r+11-(0.5)^r

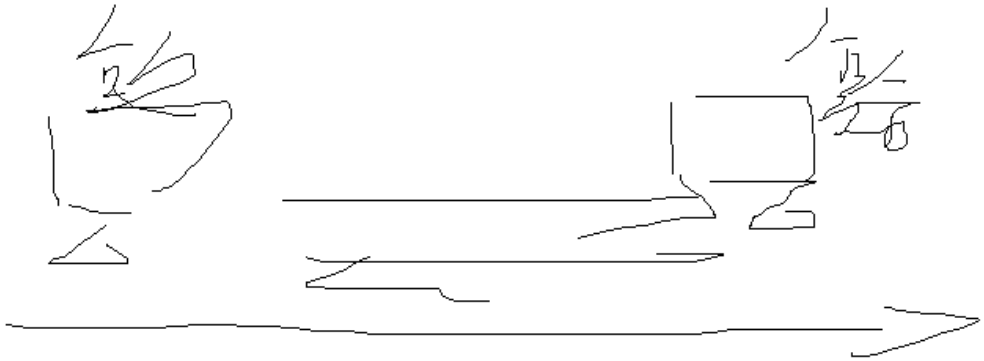
##

DLC协议

HDLC 面向位的协议

正常响应和异常响应





数据帧：无奸细

ppp协议

转发 交换

建立 认证

不提供的协议：

流量控制

乱序接收

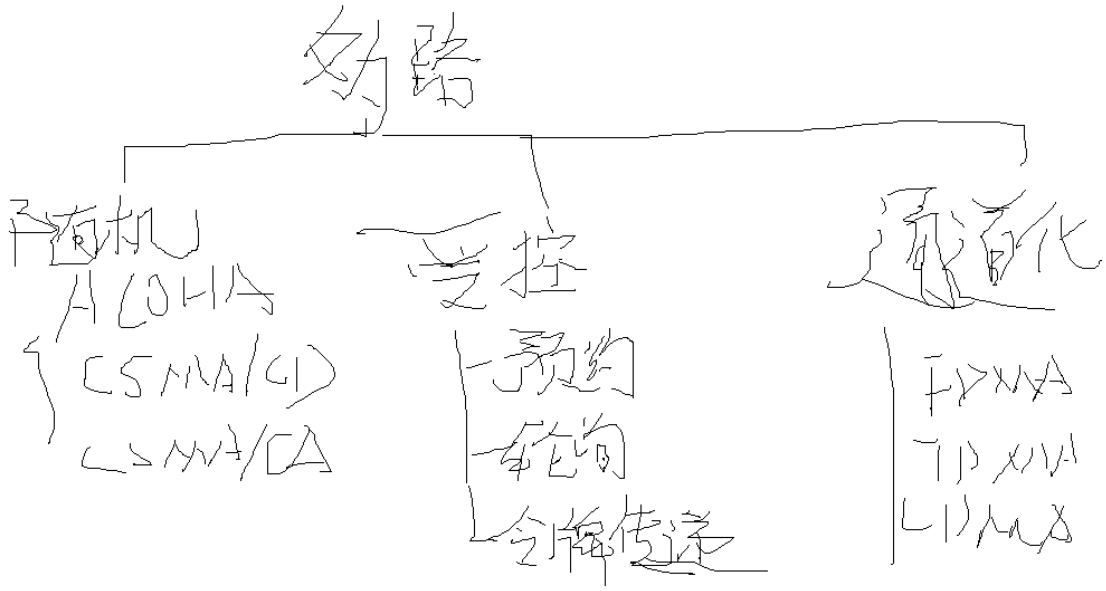
帧就是转义的字符要注意

链接--->传输---->释放

链路控制协议（LCP)

建立链接和释放链路

多路访问协议



随机访问

时间还有站点都是随机的

问题：

一个是碰撞

一个是时间

如何判断

ALOHA

任何人都可以发

如果有冲突

一个数据帧破坏了 另一个也破坏了

一个数据帧和另外一个数据帧

两个数据帧

需要一个确认帧 如果重发后没收到 需要重发

冲突后每个站都会等一段时间 这个时间叫补偿时间

超时周期是往返传输延迟的最大可能性

Tb是一个随机值

$$T_B = R * T_p$$

R选择为0到2^K-1（k是重传次数)

吞吐量
一个帧时间内传输的平均数量。

$$S = G * E^{-2G}$$

例子：

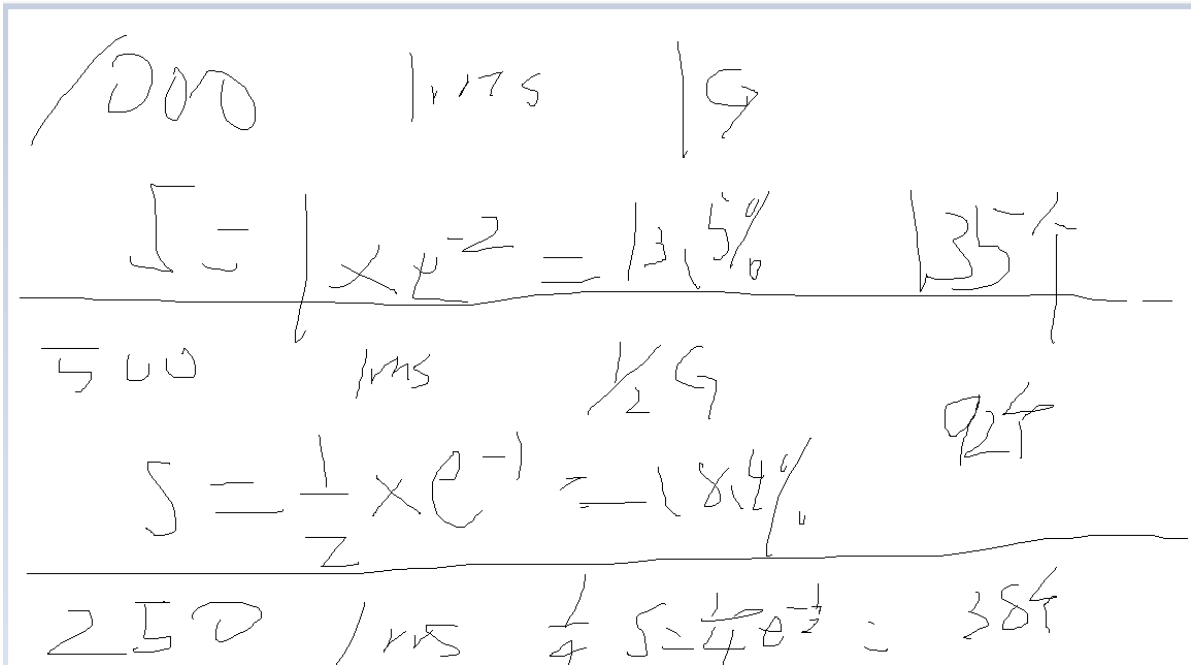
200kbp/s传输200bit

求吞吐量：

每秒1000帧 每秒500个帧 每秒250个帧

200/200kbps 1ms

计算



csma (载波监听)

传播前先监听 然后再传播

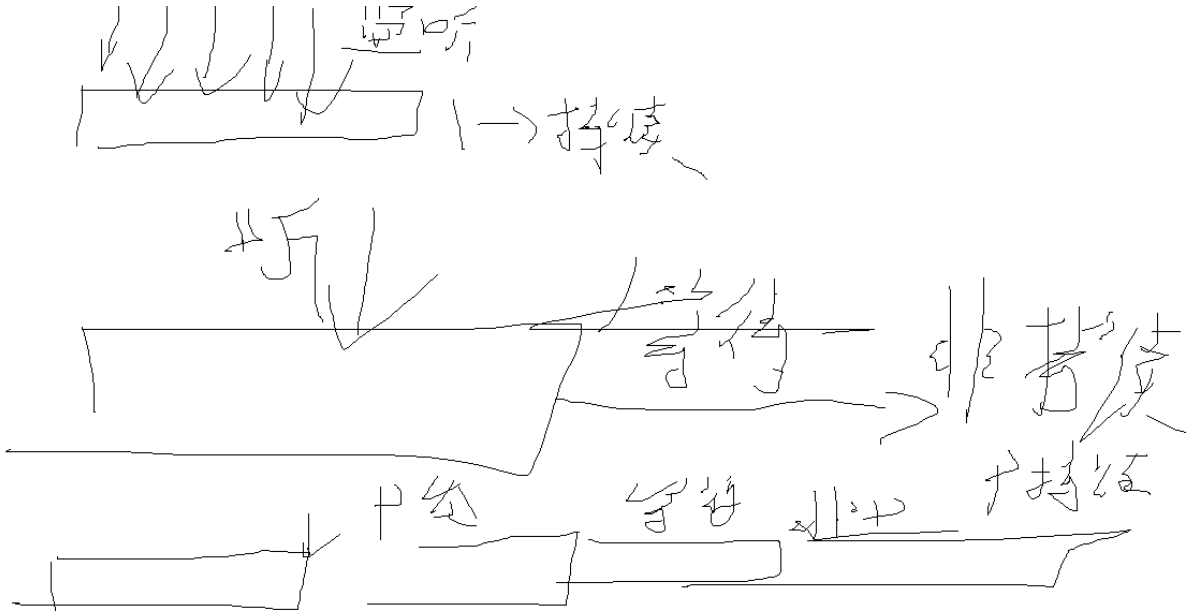
但是在传播中会有冲突

方法：

1.持续方法----> 马上发送一个帧

2.非持续---->如果是空闲 就发送 不是空闲等一会再监听一次 发送

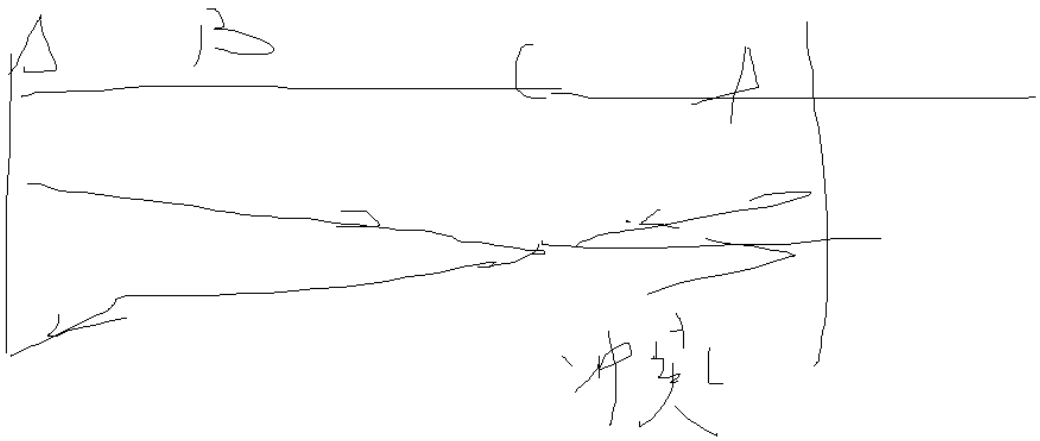
3.p持续 以p接收或者发送



csma\cd(带冲突检查的多路访问)

发送后查看站点是否有冲突

重新发这个数据帧



双方检测到变化 然后返回错误

受控协议：

预约：

如果一个站点没有其他站点的授权就不能够发送

轮询：

选择那个站

设备做主站 其他设别做从站

主站控制链接

从站遵循指令

主站询问那个站 那个站发就是那个发。

令牌传递：

令牌(token)在这个环境里面循环

用令牌 才能发

令牌的时间有管理的有限的

逻辑是环

链路层寻址

成为mac地址

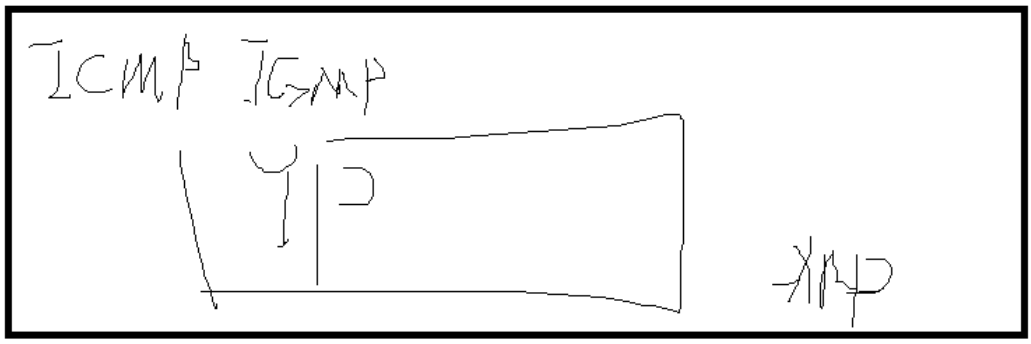
对于ip 源地址在目的地址前

对于mac 目的地址在源地址前

arp协议

链路层地址的长度。依赖于链路层使用的协议。

网络层（arp)



主机发arp广播

对应的协议返回arp回应

单播

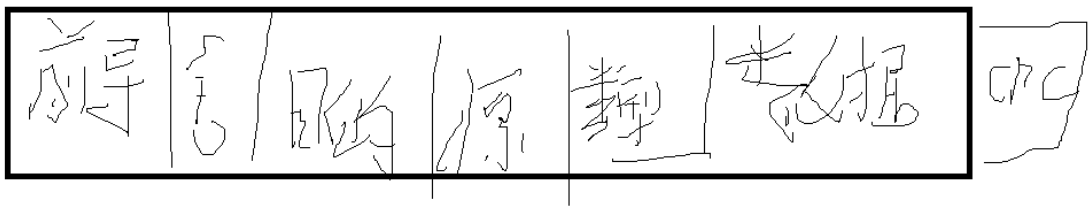
以太网帧

通过arp协议来进行转化变化

IEEE802

llc（逻辑控制层）和mac（介质访问层）

以太网帧的格式



SFD:前导吗 同步使用 121！121！ 最后2位是1

目的 源

crc检查数据帧的差错

非0就丢弃数据帧

无连接和不可靠

无连接就是不建立 想发就发

不管有没有受到 也是发

链接 不是不可靠

最小是64-18=46字节（目的6 源6 crc4类型2 -18个）

最小数据帧长度：64 最小数据长度：

每个站点都有自己的网卡

以太网地址：

4A:30:10:21:10:1A

地址位传输就是左到右边 一个一个字节

对于每个字节先是低位到高位

比如47:20

就是47： 20

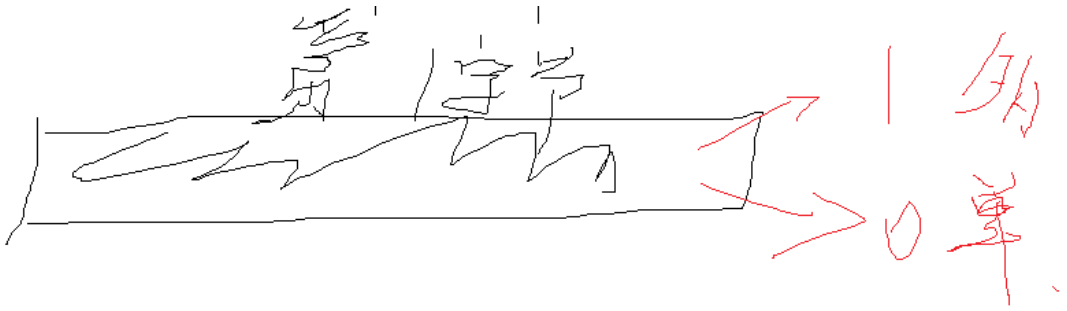
47如何转化

01000111

就反过来：

11100010

单播多播组播



所以第一个传播位就可以很快判断出来单还是多

单播：广播后收到接受方保留处理 其他的丢掉

多播：组中对应的受到处理其他丢掉

广播：整个站点接受处理

站点使用csma载波监听（半双工）

结论：发送了512位但是没有报错 就是完美的接受到了并且停止接听

数字512发送需要的时间

以太网的速度是10mbps 意味

512/10m =51.2us的时间。

第一位已经开始5120的往返

512没有监听冲突 这样就可以丢弃该数据帧

丢弃这个数据帧吧

第二个就是监听到了冲突，已经冲突了就抑制源点，并重发。退后n帧协议

每次就是等个2^K的时间重发。

每一次的补偿时间变长了

以太网的效率：

1/(1+6/4*a)

a是数据帧的数目

10basex base就是基本 基带的意思

快速以太网：

1.访问方法：

a.使用星形的拓扑

放弃总线的拓扑

2.自动协商：

允许一个点有一定的接受能力不用询问

3.实现：

双绞线

这里需要讨论：

p308

1.a发送了512位 10mbps

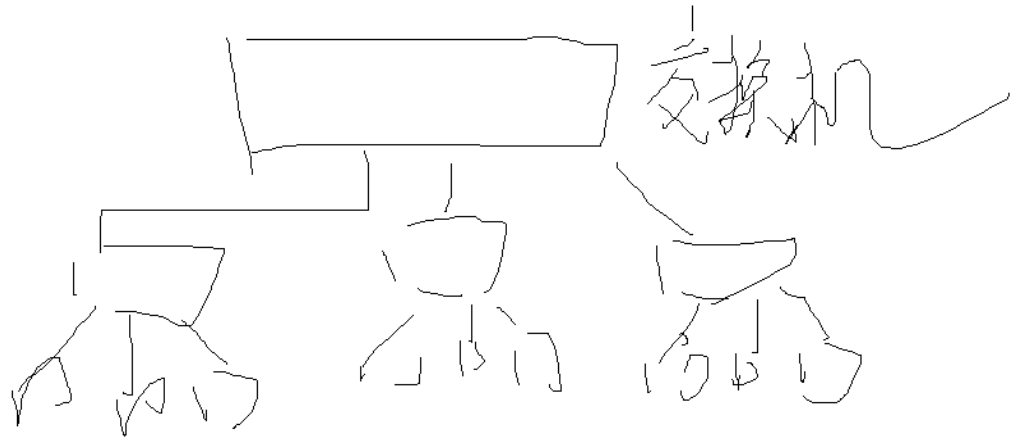
512位置需要发送时间 51.2us

就是最坏也要等到512us才发生

虚拟局域网：

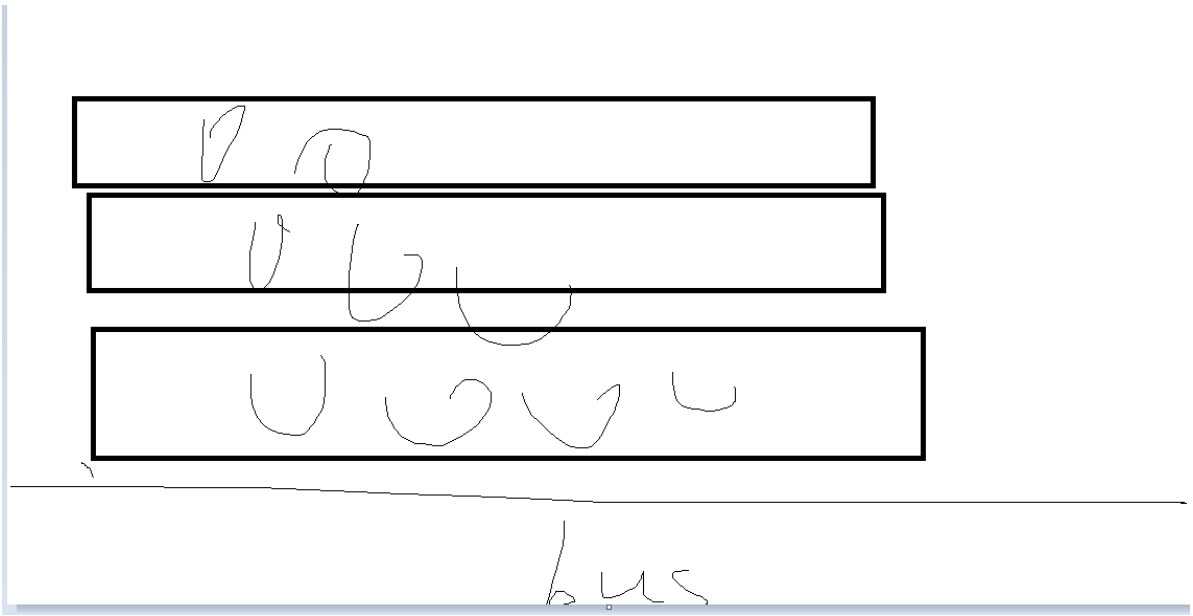
属于不同局域网之间的链接

交换机技术：



这个如果发生变化了就要重新布线很麻烦

但是如歌是虚拟的就不会有这样的问题



物理的只需要在逻辑上是分层的就可以了

这样只要软件变化就ok了

vlan的区分：

1.接口号：

定义了vlan1 vlan2之类的借口号

mac地址：

64位

配置：

手动

半自动

自动

通信：

表维护：

维护一张表格 用来记录哪里莱哪里去

帧标记

用额外的头部定义mac数据帧

时分多路复用：

后面讲解

IEEE802.1Q

1.减少配置时间

使用软件方法更加简单快速

2.创建虚拟工作组

相同的工作可以互相通信 但是不属于相同的部门

3.安全

arp的话所有人都会接收到这个数据帧 vlan有标记 其他接收不到

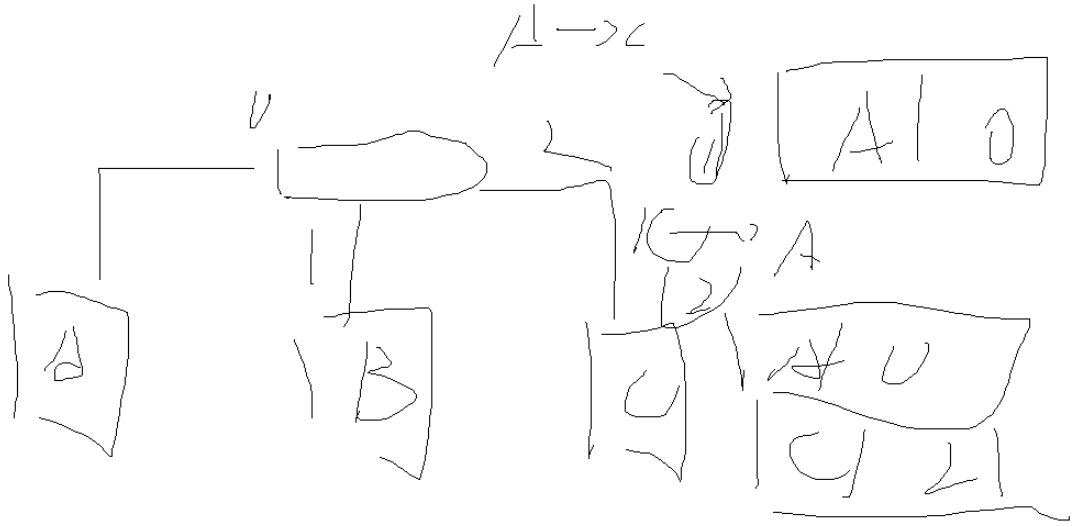
有线网络

物理设备:（交换机）

交换机 可以在表格中发现自己的表格 并且转发出去

有个表格 可以查看 不会改变mac地址

自学习过程



没有就洪范 否则就转发