网络编程基础

网络编程初识

计算机的发展

- 一开始，计算机是互相独立的
- 因为需求的缘故，出现了网络互联
- 局域网LAN：通过交换机和路由器连接在一起

> ☆ 路由器：连接不同的局域网
>
> ☆ 交换机：完成局域网内的数据转发

- 广域网WAN：将千里之外的计算机连接一起

协议

> ☆ 协议是一种约定
>
> **约定本身一定要能够通过某种数据标识出来，通讯双方也要认**

```C++
struct cmd{
    int code: 4;    //编码为4位
    int data: 28;   //数据为28位
};
//假如有一个数据
struct cmd mycmd = {1, 0x1234};
//如果没有约定，那么我们就不会知道mycmd是什么意思
```

计算机之间的传输媒介是光信号和电信号，通过“频率”和“强弱”来表示0和1这样的信息。想要传递各种不同的信息，就需要约定好双方的数据格式

网络协议初识

- 为了让不同厂商之间生产的计算机能够互相顺畅的通信，于是就共同约定了一个标准

H3 协议分层

关于通讯，同层协议，可以认为自己在和对方层直接通讯，从而达到简化对于网络协议栈的理解

OSI七层模型

- OSI(Open System Interconnection，开放系统互联)七层网络模型称为开放式系统互联参考模型，是一个逻辑上的定义和规范
- 把网络从逻辑上分为了七层，每一层都有相关、相对应的物理设备
- OSI七层模型是一种框架性的设计方法，其最主要的功能就是帮助不同类型的主机实现数据传递
- 它的最大优点：将服务器、接口和协议这三个概念明确地区分开来，通过七个层次化的结构模型使不同的系统不同的网络之间实现可靠的通讯
- 缺点：负责又不实用

- 七层(从上到下)
  - ☆ 应用层： 针对特定应用的协议
  - ☆ 表示层： 设备固有数据格式和网络标准数据格式的转换

- ☆ 会话层： 通信管理。负责建立和断开通信连接。管理传输层以下的分层
- ☆ 传输层： 管理两个节点之间的数据传输。负责可靠传输
- ☆ 网络层： 地址管理和路由选择
- ☆ 数据链路层： 互联设备之间传送和识别数据帧
- ☆ 物理层： 以0、1代表电压的高低、灯光的闪灭。界定连接器和网线的规格

协议栈(TCP/IP协议)

> 网络标准化组织定义的，所有的操作系统都必须遵循

- 有五个(或四个)层次(从上到下)
  - 应用层： 在用户空间 ，根据特定的通讯目的，进行数据分析处理达到某种业务性目的
  - 传输层： TCP ，处理传输问题，主要保证数据可靠性
  - 网络层： IP ， 负责数据转发
  - 数据链路层： 网卡中，与驱动程序强相关，负责真正的数据传输
  - 物理层

> ☆ 网络协议栈，负责数据通信工作。第一层负责业务细节，其余层负责通信细节
>
> ☆ 层状结构本质：软件工程上面的解耦，层与层之间，只有接口的互相调动关系。增加代码的可维护性和可扩展性

- 各层次的工作设备
  - 应用层：
  - 传输层：
  - 网络层： 路由器
  - 数据链路层： 以太网、令牌环网、无线LAN等标准，交换机设备

> - 物理层： 网线、同轴电缆、光纤、无线网、电磁波、集线器

☆ 集线器： 信号随着传输距离的变长，信号变得越来越弱，集线器会将电信号加强，使得信号传输距离变得更远

☆ 以太网： 采用☆ 碰撞检测的方法来实现局域网通讯，以太网是比较常见的局域网通讯规范

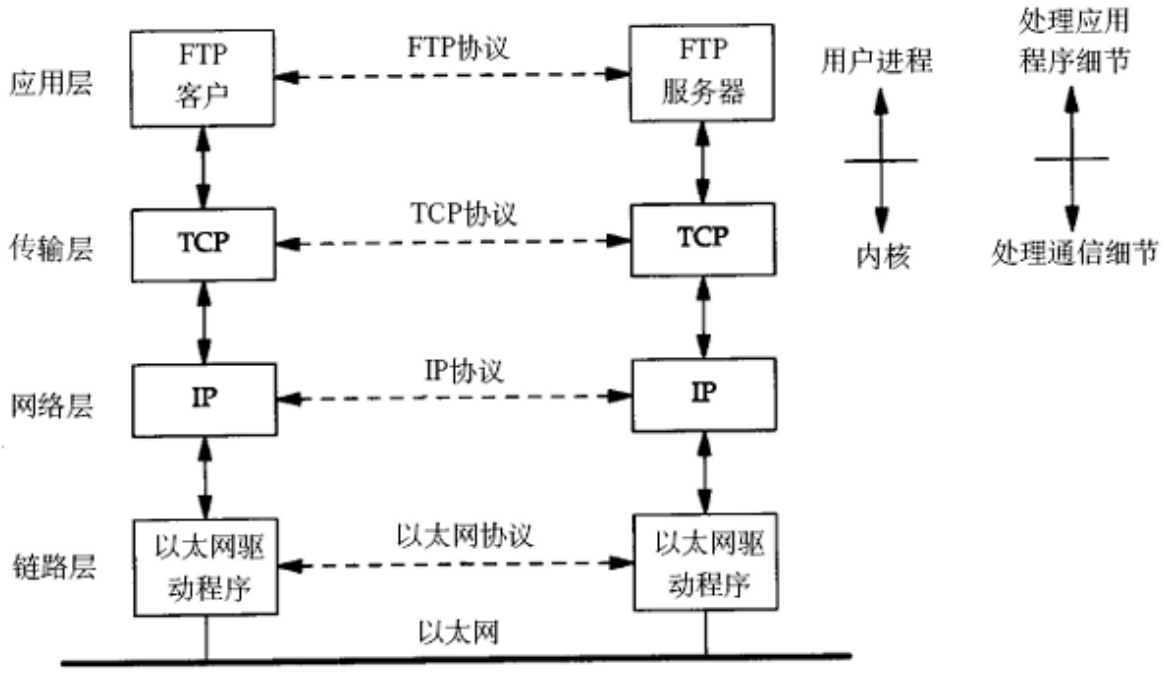
数据通讯是需要媒介的，或者说，任何信息的传递是需要媒介的

网络传输基本流程

H3 网络传输流程图

应用层	**Telnet、FTP和e-mail等**
传输层	TCP和UDP等
网络层	IP、ICMP和IGMP等
链路层	设备驱动程序及接卡口

H3 TCP/IP通讯过程



在同一个局域网，两台主机能否直接通讯?

能!

- 在传输过程中，采用自顶向下，☆ 数据封包，也就是说，从用户发送数据经过各层次会依次封装对应的☆ 报头达到封包的效果

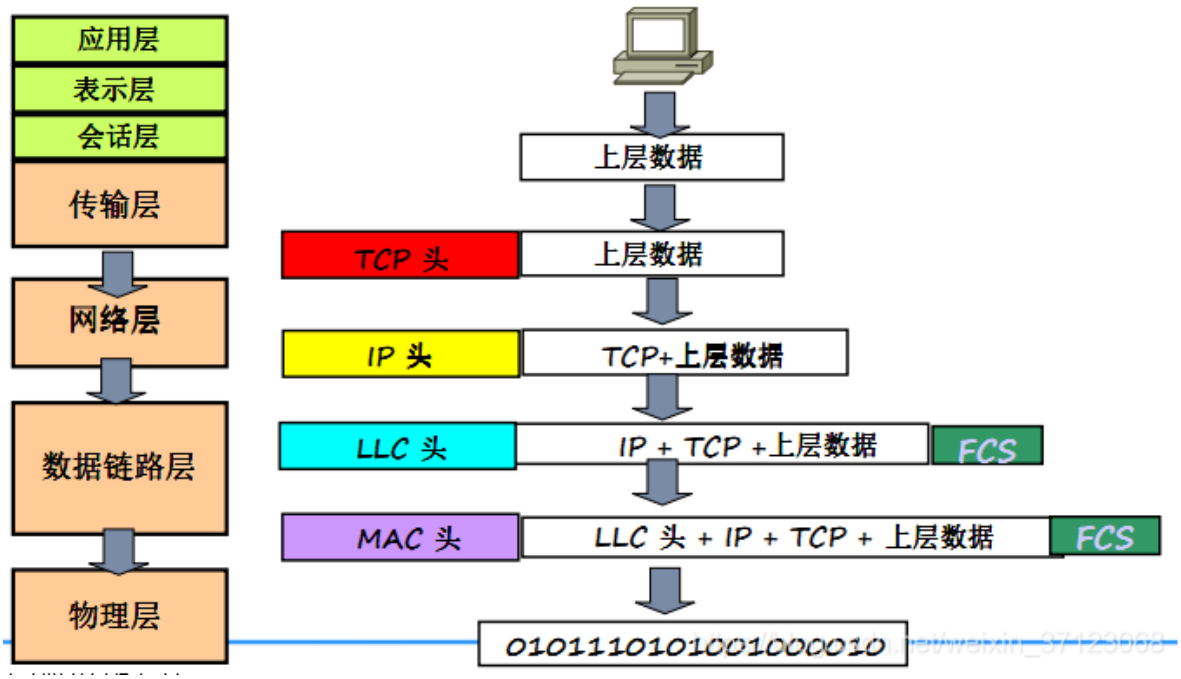
☆ 报头： 其实也是一种数据，实际上就是如下的形式

```C++
struct cmd{
    int code; //里面的具体数据根据具体结构而定
    int data;
};
```

☆ 数据封包： 实际上是操作系统提供的内核缓冲区，有一个指针指向数据的首地址，然后通过各层的报头封装后，该指针指向报头的首地址

```C++
//假如有一个http协议
struct http_header{
   ...
};
struct http_header httpheader = {...};
//指针指向httpheader的首地址完成封包
```

封装过程



- 在交付过程中，从下至顶，☆ 数据解包与☆ 分用，也就是说，用户接收数据的时候经过各层次依次解包然后使用对应的报头
- 不同层的报头，可能是下层协议的☆ 有效载荷

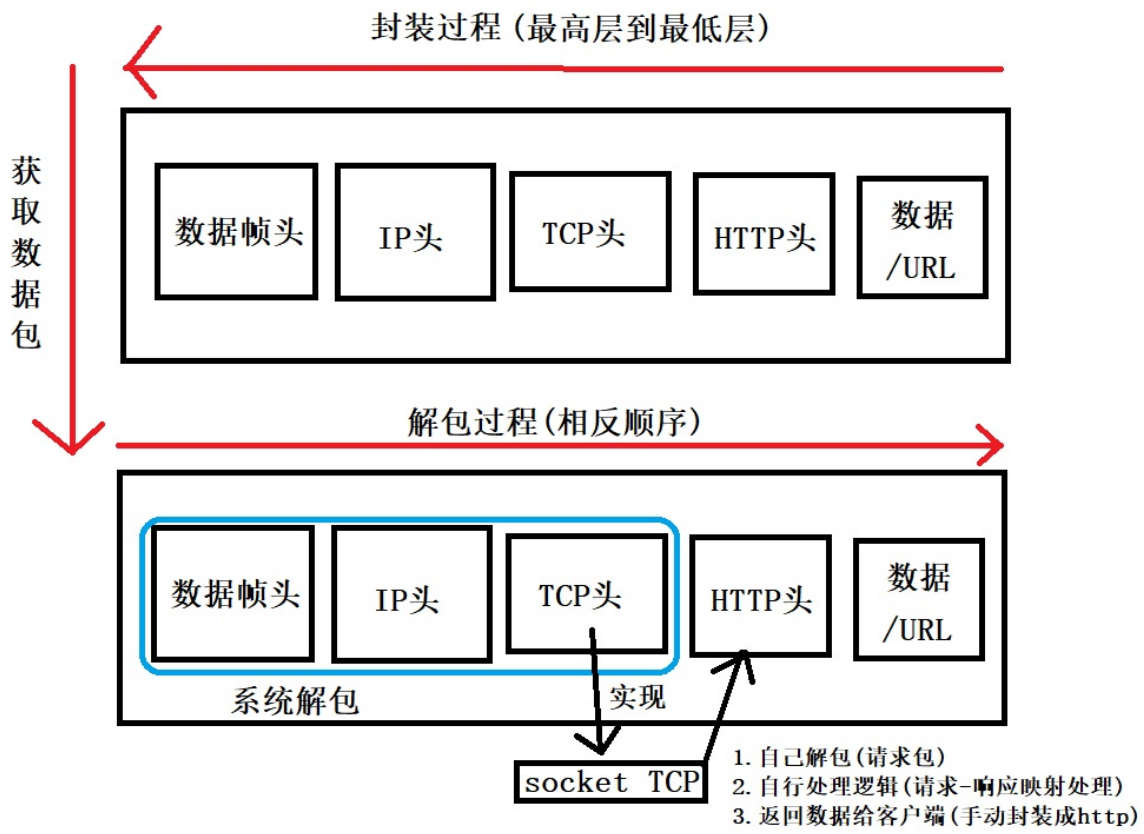
当前层，如何得知报头和有效载荷之间的分割?

- 定长报头
  - ☆ 协议共性1： 报头的大小，常规的就是这两种方法
  - 规定里面的报头是**确定大小**的
  - 自定义描述字段
    - 报头最终是结构类型
    - ```C++
      struct ip_header{
          int header_length;     //报头
          ...
          int load_length;       //有效载荷
      };
      ```

如何知道，将有效载荷交付给上层的那个协议?
      ```

> - ☆ 协议共性2： 几乎所有的协议，报头当中都包含一个字段，表明我们需要把有效载荷交付给上层的哪一个协议

- ☆ 分用： 指的就是各层交付的过程



H3 局域网(以太)的通讯原理

- 局域网中的某一台主机发出通讯信息
- 第一，局域网中的主机判断报头信息，是不是在呼叫自己
- 第二，如果不是，那么直接丢弃报文，如果是，就开始解包操作

> 本质是所有的主机在底层实际上都接收到了数据，只不过经过帅选提交上来得到了发给自身的数据

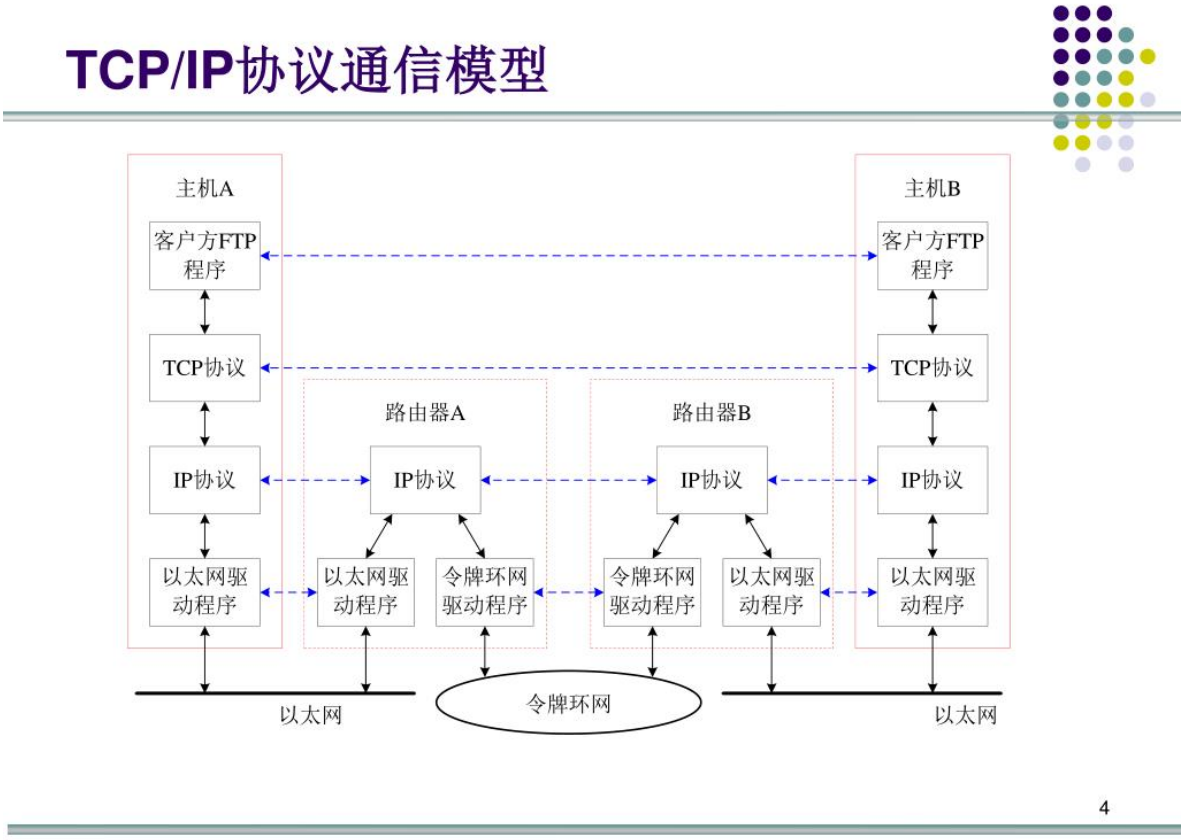
- 问题：局域网中的主机很有可能在不停的互相发送信息，这些信息可能对各自的信息造成干扰

> 因此：

1. 局域网中，网络本身的特性要求只能一个时刻只能有一个机器进行发送数据
2. **如果网络中的数据发生了碰撞，当前主机是可以检测到的**
3. 所有的主机都要进行☆ 碰撞避免算法

☆ 碰撞域： 局域网发送消息的总线发送碰撞的位置

H3 广域网的通讯原理



- 路由器，至少要能够横跨两个局域网
- 对于每个网络，都认为路由器是它局域网上的一台主机
- ☆ 令牌环网： 相当于一个控制器，如果有令牌环网那就可以传递讯息
- 有着令牌环网和以太网，那么说明底层使用的协议：**具体是完全不同的，也就是说，在最底层添加的MAC报头是不一样的**

工作原理

- 路由器通过查询路由表找到IP地址
- 然后路由器通过封装一个对应IP地址主机能够识别的报文
- 最后交付给目标主机

> 虽然在交付过程中，报头是不一样的(以太和令牌环)，但是同层次之间(TCP/IP)，目标主机和原主机看到的报文是一样的
>
> **通过IP地址，虚拟化了底层网络的差异**
>
> -> 虚拟地址空间
>
> -> 一切皆文件

MAC地址

> 网卡 --> 内置了48位的序列号，被叫做MAC地址
>
> **MAC地址是全球唯一的**

- MAC发送的数据叫做☆ MAC数据帧

> MAC数据帧的报头包含了srcMAC(源主机地址)， dstMAC(目的主机地址)
>
> - ☆ 单向发送： 将dstMAC设置为单一主机的MAC地址，进行单向的发送数据
> - ☆ 广播发送： 将dstMAC设置为类似FFFF...F也就是局域网内全体主机，向全体主机发送数据

- 在linux中查看MAC地址：ifconfig

☆ ether： 以太，也就是对应的MAC地址

☆ lo : loop, ☆ inet 代表本地换回地址，一般是127.0.0.1(本机地址)

IP地址

> 用来表示全网内唯一的一台主机
>
> **注：这里的全网指的是互联网**

H3 IP的分类

- IPV4

> 32位比特位标识☆ IP地址

- IPV6

> 128位比特位标识IP地址

总结

- 无论你的上层协议栈有多少层，最终，都必须在硬件(物理层)上进行数据传输