

Index by bossjoker1

[计算机网络学习](#)

[1. 计算机网络和因特网](#)

[因特网](#)

[协议](#)

[网络边缘](#)

[接入网:](#)

[网络核心](#)

[电路交换](#)

[分组交换网络](#)

[协议层次和服务模型](#)

[2. 应用层](#)

[应用层协议原理](#)

[进程通信](#)

[web和HTTP](#)

[HTTP概况:](#)

[HTTP报文格式](#)

[用户和服务器交互: cookie](#)

[Web缓存](#)

[因特网中的电子邮件](#)

[SMTP](#)

[DNS: 因特网的目录服务](#)

[分布式、层次数据库](#)

[DNS缓存](#)

[DNS报文格式](#)

[P2P文件分发](#)

[BitTorrent](#)

[3. 运输层](#)

[无连接运输: UDP](#)

[可靠数据传输原理](#)

[rdt 1.0](#)

[经具有比特差错信道的可靠数据传输: rdt 2.x](#)

[经具有比特差错的丢包信道的可靠数据传输: rdt 3.0](#)

[流水线可靠数据传输协议](#)

[回退N步:](#)

[选择重传 \(SR协议\)](#)

[面向连接的传输: TCP](#)

[TCP报文段首部结构](#)

[往返时间的估计与超时](#)

[可靠的TCP数据传输](#)

[流量控制:](#)

[TCP连接建立和释放](#)

[TCP拥塞控制 I](#)

[TCP拥塞控制算法](#)

[4. 网络层: 数据平面](#)

[概述](#)

[网络服务模型](#)

[路由器工作原理](#)

[输入端口处理和基于目的地转化](#)

[网际协议](#)

[IPv4数据报格式](#)

[IPv4数据报分片](#)

[IPv4编址](#)

[网络地址转换\(NAT\)](#)

[IPv6](#)

[通用转化和SDN\(软件定义网络\)](#)

[5.网络层：控制平面](#)

[路由选择算法](#)

[链路状态路由选择\(LS\)算法](#)

[距离向量路由选择算法\(DV\)算法](#)

[因特网上的距离向量算法-RIP协议](#)

[因特网中自治系统内部的路由选择：OSPF](#)

[开放最短路优先\(OSPF\)](#)

[AS间的路由选择：BGP\(Broder Gateway Protocol\)边界网关协议](#)

[通告BGP路由信息](#)

[确定最好的路由](#)

[IP任播](#)

[路由选择策略：](#)

[ICMP 因特网控制报文协议](#)

[6.链路层和局域网](#)

[链路层概述](#)

[提供的服务：](#)

[差错检测和纠正技术](#)

[奇偶校验](#)

[检验和\(checksum\)方法](#)

[循环冗余检测\(Cyclic Redundancy Check CRC\)](#)

[多路访问链路和协议](#)

[1. 信道划分协议](#)

[2. 随机访问协议](#)

[ALOHA\(随机接入协议\)](#)

[载波侦听多路访问\(CSMA\)](#)

[CSMA/CD](#)

[3. 轮流协议](#)

[交换局域网](#)

[以太网](#)

[信号编码](#)

[链路层交换机](#)

[网桥互联](#)

[虚拟局域网VLAN](#)

[数据中心网络](#)

[回顾：Web页面请求的历程](#)

计算机网络学习

[\[Top\]](#)

1.计算机网络和因特网

[\[Top\]](#)

因特网

[Top]

1. 连接在因特网上的数以十亿计的互联计算机设备：主机(端系统)
2. 连接因特网上各种设备的通信链路
3. 转化数据的分组交换机

协议

[Top]

定义两个或多个通信实体之间交换的报文格式和次序，以及报文发送和接收或其它事件所采取的动作

网络边缘

[Top]

- 端系统(主机)：用于运行应用程序
- 网络应用通信模型：
 1. C/S
 2. P2P
 3. 混合模型

接入网：

[Top]

将网络边缘与网络核心连接起来，通常是将端系统连接到边缘路由器上。**边缘路由器**指端系统到任何其它远程端系统的路径上的第一台路由器。

- 家庭：拨号 => DSL(频分复用) => HFC
- 企业：以太网 / Wi-Fi
- 公司或大学：局域网(LAN)
- 以太网 通过共享或专有链路来连接端系统和路由器

网络核心

[Top]

电路交换

[Top]

- 数字通信
- 预留端系统沿路径通信所需要的资源
- 交换期间用户始终占有端到端的固定传输信道
- 实时进行没有延迟，实际传输容量是链路总容量的平均

时分复用TDM：时间被划分为固定期间的帧，每个帧被划分为固定数量的时隙(用于传输该连接的数据)

频分复用FPM

‡ 不适合用于计算机间的数据交换

分组交换网络

[Top]

- 分组：源将长报文划分为较小的数据块(将数据段左边添加首部)
- 分组交换机传送分组(路由器和链路层交换机)
- 使用通信链路的最大传输速率

存储转化传输：分组交换机将整个分组收下并存储后再发送(数据校验)。

特征：网状拓扑结构。主干线路由高速链路构成。

传输时延：每个分组经历每一个设备都需要 $\frac{L}{R}$ 的时间，其中L为分组长度(bit)，R为链路比特速率(bps)

则N条链路(N-1个路由器)的传输时延为 $N \frac{L}{R}$ 。

排队时延：缓存过程中排队

丢包：缓存溢出，随机丢弃。

分组转发的路径：

- 分组首部包含目的地址
- 路由器里面有转发表，在转发表里检索目的地址，获取转发链路
- 转发表基于 <路由选择协议形成>

分组交换的时延：

- 节点处理时延 d_{proc} ：检测比特差错，确定输出链路
- 排队时延 d_{queue} ：取决于路由器的拥塞程度
- 传输时延 d_{trans} ：分组比特流发送到链路上的时间 L / R
- 传播时延 d_{prop} ： d / s

$$\text{总时延 } d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

分组交换/电路交换对比：P21的例子

协议层次和服务模型

[\[Top\]](#)

🔗如何组织复杂的系统

2. 应用层

[\[Top\]](#)

应用层协议原理

[\[Top\]](#)

主流体系结构：

- 客户-服务器体系结构(C/S)

客户端：

- 客户之间不能直接通信
- 可能有动态的IP地址
- 可能是间歇性连接

服务器：

- 总是打开
- 拥有固定的IP地址
- 缩放的数据中心

- 对等(P2P)体系结构

- 不需要总在线
- 任意节点之间可以直接通信
- 节点的地址以及他们之间的连接可能随时发生变化
- 任何一方既可以做客户端也可以是服务端
 - 自扩展：新的节点带来新的服务能力，同时带来新的服务需求
- 节点之间是间歇性连接，地址也可以变化

- 混合体系结构

进行通信的实质上是进程而不是程序。

不同主机之间的进程通信采用**报文交换**方式(应用层协议)。

套接字：每个网络应用进程都有一个属于自己的套接字，包含

- 主机地址：32位IP地址标识网络应用进程运行在哪一台主机上
- 端口地址：16位端口号
- 即共48位，形如 `xxx.xxx.xxx.xxx:port`

web和HTTP

[\[Top\]](#)

web的应用层协议是超文本传输协议：HTTP；

web页面是由对象组成的，一个对象只是一个文件，多数web文件包含一个HTML基本文件和引用对象。

HTTP概况：

[\[Top\]](#)

定义了web客户向web服务器请求web页面的方式以及服务器向客户传送web页面的方式。

使用TCP作为支撑运输协议，为HTTP提供可靠数据传输服务。

HTTP是**无状态协议**。默认方式下使用持续连接。

1. 非持续性连接：

- 必须为每一个请求对象建立和维护一个全新的连接
- 每个对象经受两倍往返时间(RTT)的交付时延，一个用于创建TCP，一个用于请求和接收对象

2. 持续连接

服务器在发送响应后，保持连接，用于后续对象的传送。只有超过一个可配置的超时间隔后仍未使用，则关闭该连接。

减少了服务器端套接字资源的占用，提高了服务器的负载能力。

持久连接可分为：

- 非流水线方式：一个完成后再继续下一个
- 流水线方式：一次性发送所有请求，慢慢接收

HTTP报文格式

[\[Top\]](#)

1. 请求报文：

◆一段典型的HTTP请求报文（ASCII）

```
GET /index.html HTTP/1.1\r\nHost: www-net.cs.umass.edu\r\nUser-Agent: Firefox/3.6.10\r\nAccept: text/html,application/xhtml+xml\r\nAccept-Language: en-us,en;q=0.5\r\nAccept-Encoding: gzip,deflate\r\nAccept-Charset: ISO-8859-1,utf-8;q=0.7\r\nKeep-Alive: 115\r\nConnection: keep-alive\r\n\r\n
```

请求行
(GET, POST,
HEAD 命令)

首部
诸行

单独一行回车、换行
表示报文首部结束

回车符
换行符

请求报文的一般格式：



2. 响应报文

- 状态行: 协议/版本 状态码 短语(状态信息)
- 首部行
- 空行
- 实体体

用户和服务器交互: cookie

[Top]

cookie在以下4个组件上都有部署:

1. HTTP响应报文中有一个cookie首部行
2. HTTP请求报文中也有一个cookie首部行
3. 浏览器管理用户的cookie文件(端系统)
4. web站点后端数据库保存

服务器为用户创建一个带时效的唯一标识, 则用户可以携带这个cookie在无状态的HTTP上建议一个用户会话层。

⚠ **隐患:**网站可以获取用户的信息, 搜索引擎、推荐算法基于cookie中的信息进行广告或者产品推送。

Web缓存

[Top]

web缓存器称为代理服务器。

1. 浏览器创建到web缓存器的TCP连接, 向web缓存器中的对象发送HTTP请求;
2. web缓存器检查本地是否存储了该对象的副本, 如果有则直接响应并返回该对象
3. 如果没有, web缓存器则向该对象的初始服务器发送TCP连接, 然后进行HTTP请求
4. web缓存器接收到该对象时, 在本地存储一份副本, 然后将副本发送给用户

P73 平均时延的计算和对比

eg: 内容分发网络(CDN)

因特网中的电子邮件

[Top]

- 用户代理
- 邮件服务器
- SMTP (简单邮件传输协议) 核心

- 必须使用7bit ASCII码格式
- 基于TCP连接，端口号为25
- 使用持续连接

主要流程

1. Alice调用邮件代理程序得到Bob的邮件地址，然后发送消息
2. 用户代理发送Alice的消息到她的邮件服务器，消息被保存在消息队列中
3. 邮件服务器上的SMTP客户端向Bob的邮件服务器建立TCP连接
4. 通过该连接，SMTP发送Alice的报文
5. 消息被存储在Bob的邮箱中

Alice代理 —(SMTP) > Alice邮件服务器 —(SMTP)> Bob邮件服务器 —(pop3\imap\http)> Bob代理

避免了我给你发邮件时你不在，接受不到的问题,最后一步不是SMTP是因为SMTP是推协议而不是拉协议。

邮件报文格式：

典型的报文首部为：

```
From: Alice@xxx.xx
To: Bob@xxx.xx
Subject: xxx
(空白行)
报文体(ASCII 编码)
```

POP3:

- 下载-删除：用户下载后如果更换客户机则无法再次阅读原来的邮件
- 下载-保存：在不同的客户机上都会下载并保存邮件的副本
- POP3无状态
- 重组织只能在邮件下载到本地上做

IMAP

- 将所有的邮件都保存在服务器上
- 允许用户在服务器上组织自己的邮件目录
- 维护了会话的用户信息即是有状态的

DNS:因特网的目录服务

进行主机名到IP地址转换的目录服务，运行在UDP上，使用53端口

- i. 一个分布式数据库
- ii. 使得主机能够查询分布式数据库的应用层协议

可提供的服务：

- 域名转换
- 主机/邮件别名
- 负载均衡
- i. 一个域名对应多个IP

ii. DNS服务器在多个IP中进行轮转

用户主机发送HTTP请求到Web服务器 `www.example.com/index.html` , 需要知道其对应的IP地址, 具体操作为:

- 该主机为DNS应用客户端
- 浏览器从url中抽取主机名 `www.example.com` , 返回给客户端
- 客户端向DNS服务器发送一个包含主机名的请求
- 客户端收到响应报文(包含对应主机名的IP地址)
- 浏览器接受到DNS的IP地址后, 可以向该IP地址80端口的HTTP进程发起TCP连接

分布式、层次数据库

[Top]

1. 根DNS服务器
2. 顶级域服务器 TLD (eg: com edu org net)
3. 权威DNS服务器 eg: `amazon.com`
4. 本地DNS服务器

查询分为:

- 迭代查询: 后续解析, (用的较多)
- 递归查询: 以自己的名义发起请求并获得映射

DNS缓存

[Top]

将映射缓存在本地存储器中, 有时效性。

DNS报文格式

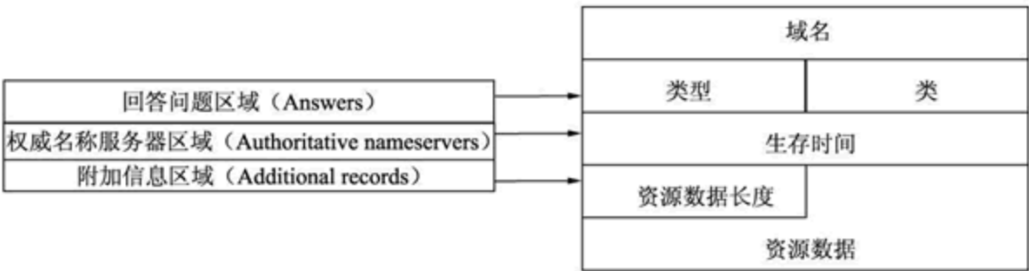
[Top]

存储的是**资源记录(RR)**:

(Name, Value, Type, TTL) 四元组

其中TTL表示记录的生存时间, 同时决定了资源记录应当从缓存中删除的时间。Type决定Name和Value, 具体含义可以查。

事务ID (Transaction ID)	标志 (Flags)
问题计数 (Questions)	回答资源记录数 (Answer RRs)
权威名称服务器计数 (Authority RRs)	附加资源记录数 (Additional RRs)
查询问题区域 (Queries)	
回答问题区域 (Answers)	
权威名称服务器区域 (Authoritative nameservers)	
附加信息区域 (Additional records)	



- head (前12字节):

- 标识符 (报文id)
- 标志字段。
- 问题计数: DNS 查询请求的数目。
- 回答资源记录数: DNS 响应的数目。
- 权威名称服务器计数: 权威名称服务器的数目。
- 附加资源记录数: 额外的记录数目 (权威名称服务器对应 IP 地址的数目)
- 问题区域: 包含正在进行的查询信息, 分为查询名, 类型(type字段), 类 (地址类型)
- 资源记录部分:
 - 回答问题区域
 - 权威区域
 - 附加区域

P2P文件分发

[\[Top\]](#)

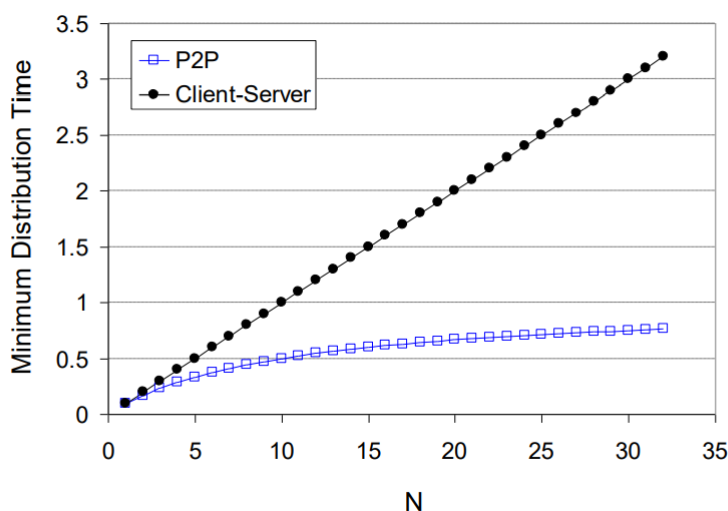
高扩展性, 对等方既是客户机又是服务器。

对于传统的C/S模式, 有 $T_{cs} \geq \max\{\frac{NF}{u_s}, \frac{F}{d_{min}}\}$

即随着分发数量的增大, 分发时间呈现线性性。

对P2P: $T_{cs} \geq \max\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^N u_i}\}$

非线性增长, 且斜率越来越小。



BitTorrent

[\[Top\]](#)

- 洪流(torrent): 参与一个特定文件分发的所有对等方的集合
- 追踪器(tracker): 跟踪正参与在洪流中的对等方
- 文件块(chunk): 固定大小为256kb

基本工作机制:

- 向邻居请求哪些块: **最稀罕优先**
- 优先响应哪些请求: **对换算法**, 即对以最高速率给其提供数据的邻居给出优先权

3.运输层

[\[Top\]](#)

运输层协议: 为运行在不同主机上的应用程序之间提供逻辑通信功能。

eg:

快递公司：网络层协议，快递点：运输层协议

TCP/UDP基本责任：将两个端系统间IP的交付服务扩展为运行在端系统上的两个进程间的交付服务。该过程也称为运输层的**多路复用与多路分解**。

1. 无连接情形：

UDP的套接字由**目的IP地址**和**目的端口号**的二元组标识，那么如果两个UDP报文段有不同的源IP地址和源端口号但具有相同的IP地址和目的端口号，那么将通过目的套接字被定向到相同的目的进程。

2. 面向连接

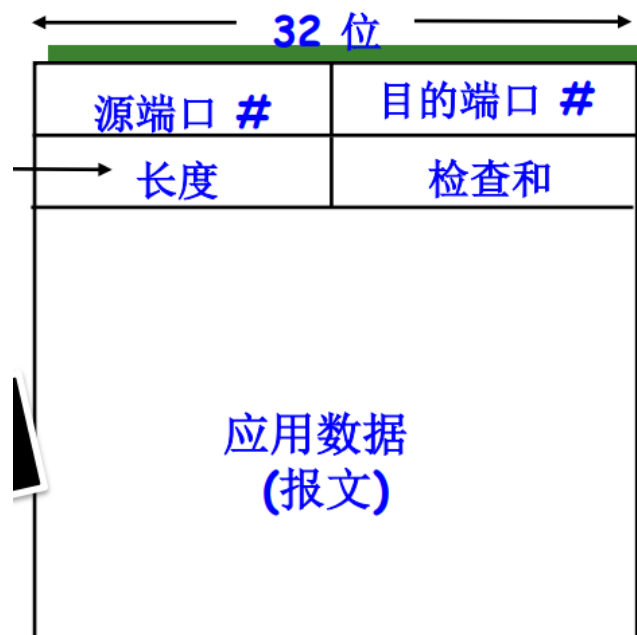
TCP套接字由四元组标识，即包括**源IP地址**和**源端口号**，服务器同时支持多个并发的TCP套接字。Web服务器为每个客户连接都产生不同的套接字。

无连接运输：UDP

[\[Top\]](#)

- 无需建立连接，即无建立连接的时延
- 无连接状态
- 分组首部开销小
- 无拥塞控制，利弊都有，比如会导致高丢包率

报文格式：



长度：指示UDP报文段中的字节数

UDP检验和：

发送方：

- 发送方的UDP对报文段中的所有16比特字的和进行反码运算，求和时遇到溢出则**回卷**
- 结果放到UDP报文段中的 `checksum`

接收方：

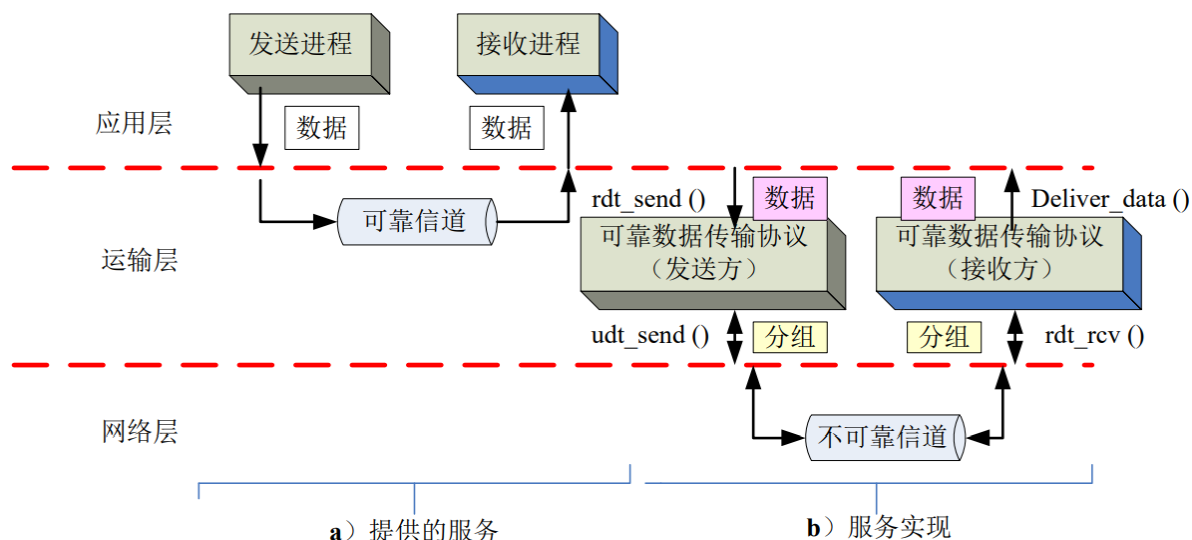
- 计算接受到的报文段的和
- 与发送方的检验和相加
 - 不为全1，有错（无差错恢复能力）

- 全1，没有检测到错（但还是可能有错）

可靠数据传输原理

[\[Top\]](#)

可靠数据传输协议(rdt)的下层协议也许是不可靠的(不可靠信道)导致了其困难性。



rdt 1.0

[\[Top\]](#)

完全可靠信道的可靠数据传输。

经具有比特差错信道的可靠数据传输：rdt 2.x

[\[Top\]](#)

1. rdt 2.0

假设：

- 分组比特可能受损
- 传输分组能保持顺序

处理：

- 差错检测
- 接收方反馈：肯定确认ACK(1)和否定确认NAK(0)
- 重传

停等协议：即仅当接收到ACK并离开该状态才能发生其它事件

缺陷 ↓：ACK/NAK分组可能也受损

2. rdt 2.1

- 重传--收到破坏的ACK或者NAK
- 接收方要对ACK计算检验和并封装
- 添加分组序号，接收方收到重复序号的数据则丢弃

3. rdt 2.2

- 接收方需要包括由一个ACK报文所确认的分组序号
- 实现无NAK的协议

经具有比特差错的丢包信道的可靠数据传输：rdt 3.0

[\[Top\]](#)

比特交替协议

由rdt 2.2的分组序号处理冗余数据分组情况。

设置倒计时定时器，实现基于时间的重传机制。

分为无丢包、分组丢失、丢失ACK、过早超时4个情况。

流水线可靠数据传输协议

[Top]

允许发送方发送多个分组而无需等待。

要求：

- 必须增加序号范围，每个分组有唯一的序号
- 发送方和接收方缓存多个分组
 - 差错恢复：回退N步(GBN)，选择重传(SR)

回退N步：

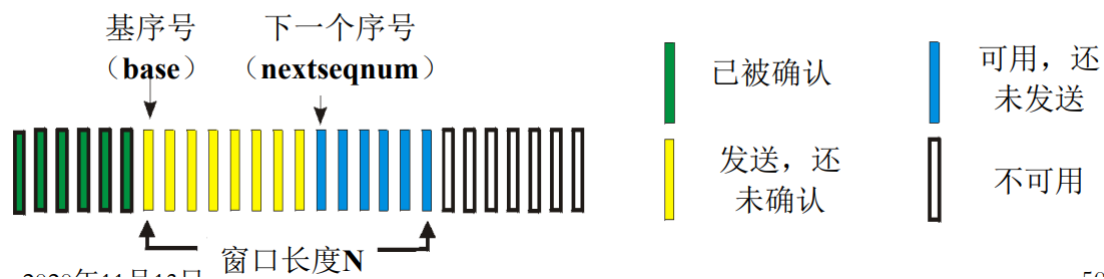
[Top]

滑动窗口：

◆ 分组首部用k-比特字段表示序号

◆ 未被传输和已被传输但还未确认的分组的许可序号范围可以看作是一个在序号范围内大小为N的“窗口(window)”

◆ 窗口——序号集合、序号管理器



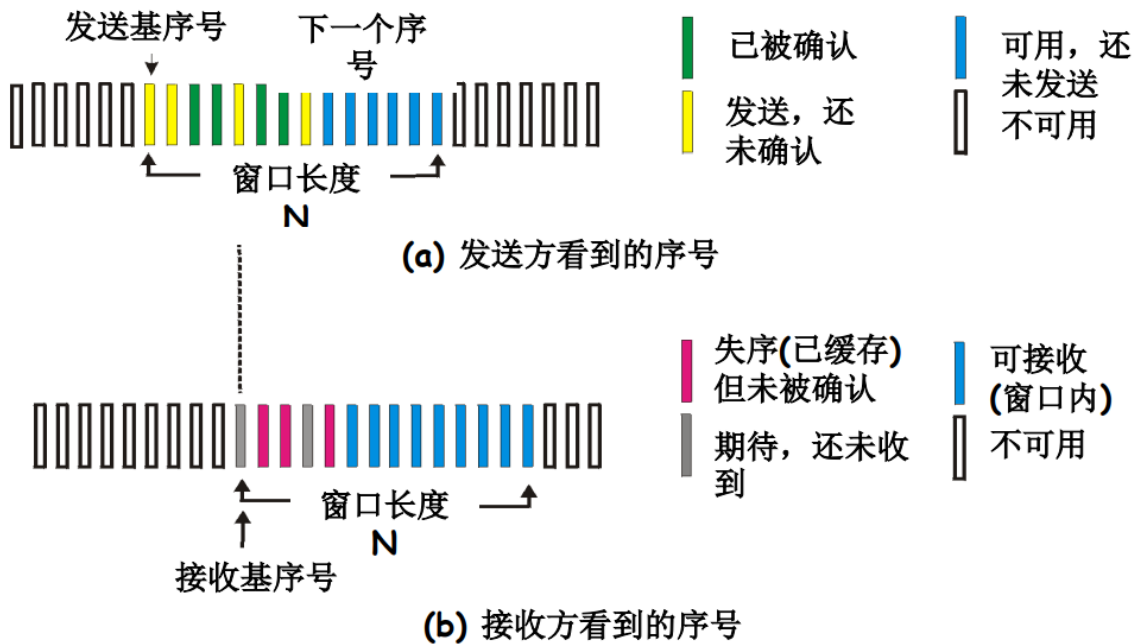
- 累计确认，表明接收方正确接受到序号为n的以前且包括n在内的所有分组
- 超时则重传所有已发送但还未被确认过的分组即在 $[base, nextseqnum - 1]$ 范围内。

收到重复的分组怎么处理 => 丢弃分组，重发ACK

分组失序 => 重发已经按序到达的最高序号分组的ACK

选择重传 (SR协议)

[Top]



发送方:

1. 从上层收到数据, SR发送方检查下一个可用于该分组的序号。如果序号在窗口内, 则将数据打包并发送
2. 超时, 为每一个分组定义定时器, 超时重传分组对应分组n并重置定时器
3. 收到确认n在 $[sendbase, sendbase + N - 1]$, 标记分组为已接收, 如果n是发送窗口基序号 $sendbase$, 则将窗口基序号前推到下一个未确认序号

接收方:

1. 分组序号在 $[rcvbase, rcvbase + N - 1]$ 范围内
 - 发送对应确认ACK(n)
 - 如果失序或者以前没接收过, 则缓存
 - 按序分组: 将该分组及以前缓存的序号连续的分组一起交付给上层, 将窗口前推到下一个未收到的分组
 - 序号在 $[rcvbase - N, rcvbase - 1]$ 即曾经确认过, 但是可能ACK丢失, 则重传确认ACK(n)
2. 其它情况: 忽略该分组

发送方窗口的长度和接收方窗口的长度之和小于等于 $\leq 2^k$, k为分组序号字段比特数。

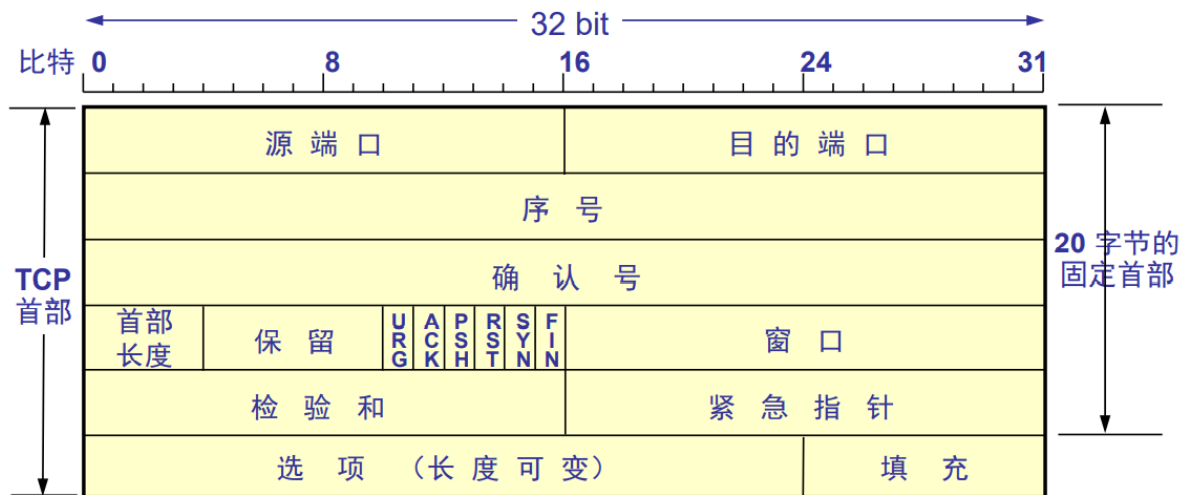
面向连接的传输: TCP

[\[Top\]](#)

- 面向连接
- 全双工服务: 双向同时传输数据
- 点对点连接
- 三次握手, 四次挥手
- 可靠的字节流

TCP报文段首部结构

[\[Top\]](#)



- 源目的端口各占2字节
- 序号字段，4字节，其值指的是本报文段所发送的数据的**第一个字节**在整个报文字节流中的序号
- 确认号字段，4字节，期望收到的对方**下一个报文段**的数据的第一个字节的序号
- 接收窗口，2字节，用于**流量控制**，表示接收方愿意接受的字节数量
- 首部长度，4比特，是以32比特为单位的TCP首部长度，即最大可表示60字节。选项长度可变，通过填充确保整个首部长度为4字节的整数倍
- 6比特标志位
- 检验和，2字节，检验范围包括首部和数据两部分，并在计算时，加上12字节的IP地址的伪首部

往返时间的估计与超时

[\[Top\]](#)

指数加权移动平均：

$$EstimatedRTT = 0.875 \cdot EstimatedRTT_{old} + 0.125 \cdot SampleRTT$$

由于测量RTT的变化也是有价值的，所以定义RTT偏差：

$$DevRTT = (1 - \beta) \cdot DevRTT_{old} + \beta \cdot |SampleRTT - EstimatedRTT_{old}|$$

参考值， $\beta = 0.25$ ，第一次计算 $DevRTT = 0.5 \cdot SampleRTT$

最终定义TCP中的超时间隔为

$$TimeoutInterval = EstimatedRTT + 4 \cdot DevRTT$$

可靠的TCP数据传输

[\[Top\]](#)

- 按字节编号
- 采用**唯一超时定时器**，而不是对每一个报文
- 选择重传和GBN的混合体 => **选择确认**，即由**接收方**有选择地确认失序报文段

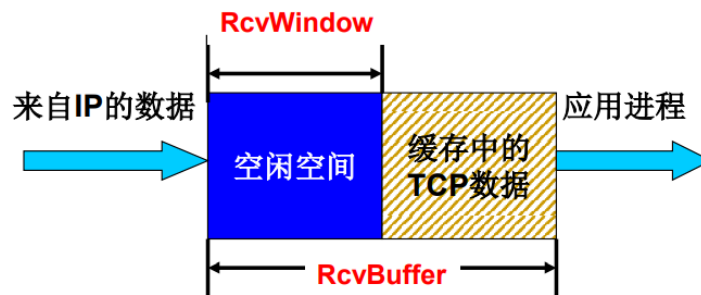
快速重传：

- 通过重复的ACK检测丢失报文段，即一旦收到**3个冗余的ACK**则执行快速重传

超时间隔加倍：

- 每一次TCP重传则将下一次**超时间隔**设为**先前值的两倍**
- 提供了一个形式受限的拥塞控制

防止发送方传数据过快导致接收方缓存溢出。通过**接收窗口**字段将缓冲区剩余空间反馈给发送方。

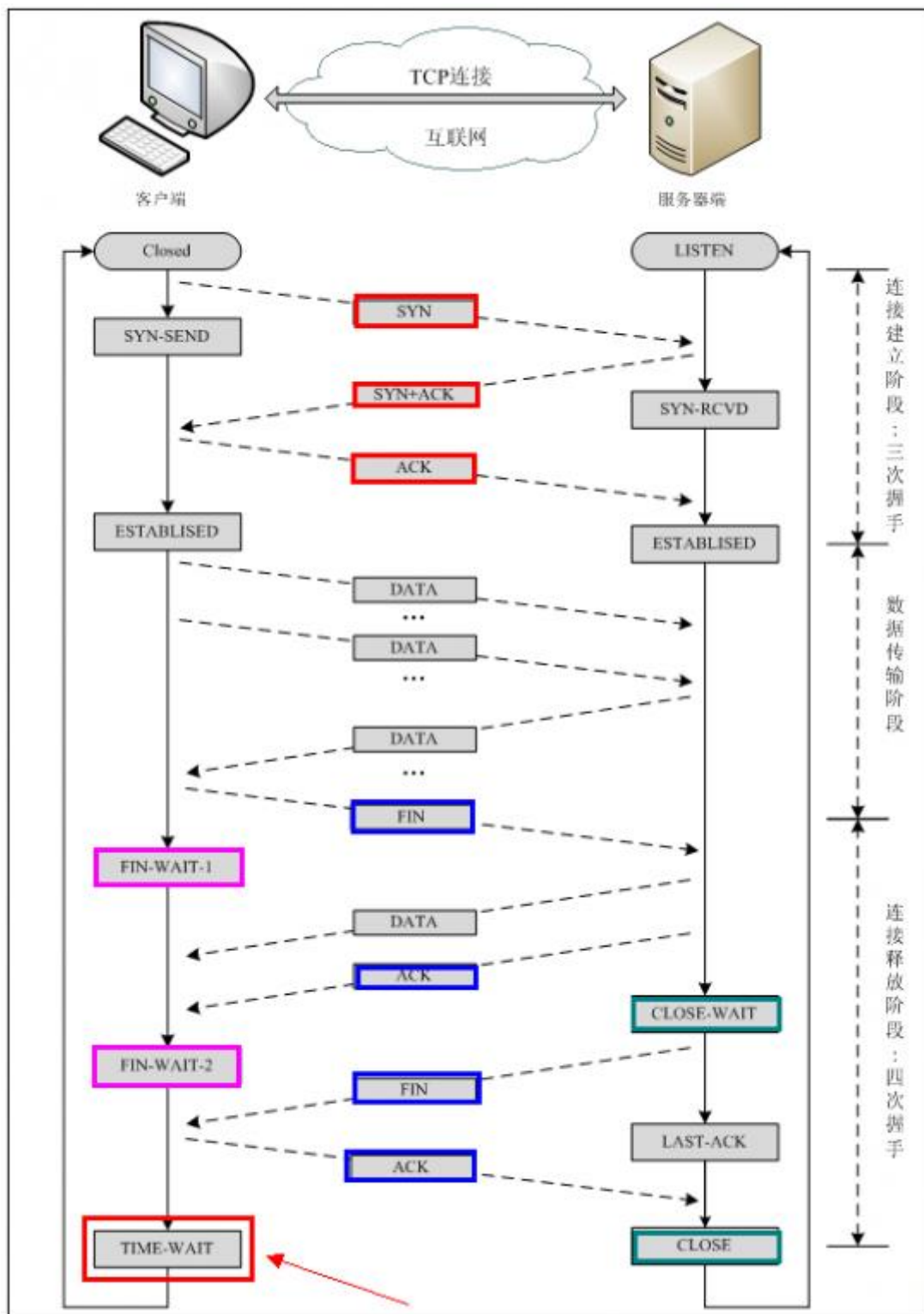


接收方: $\text{LastByteRcvd} - \text{LastByteRead} \leq \text{RcvBuffer}$

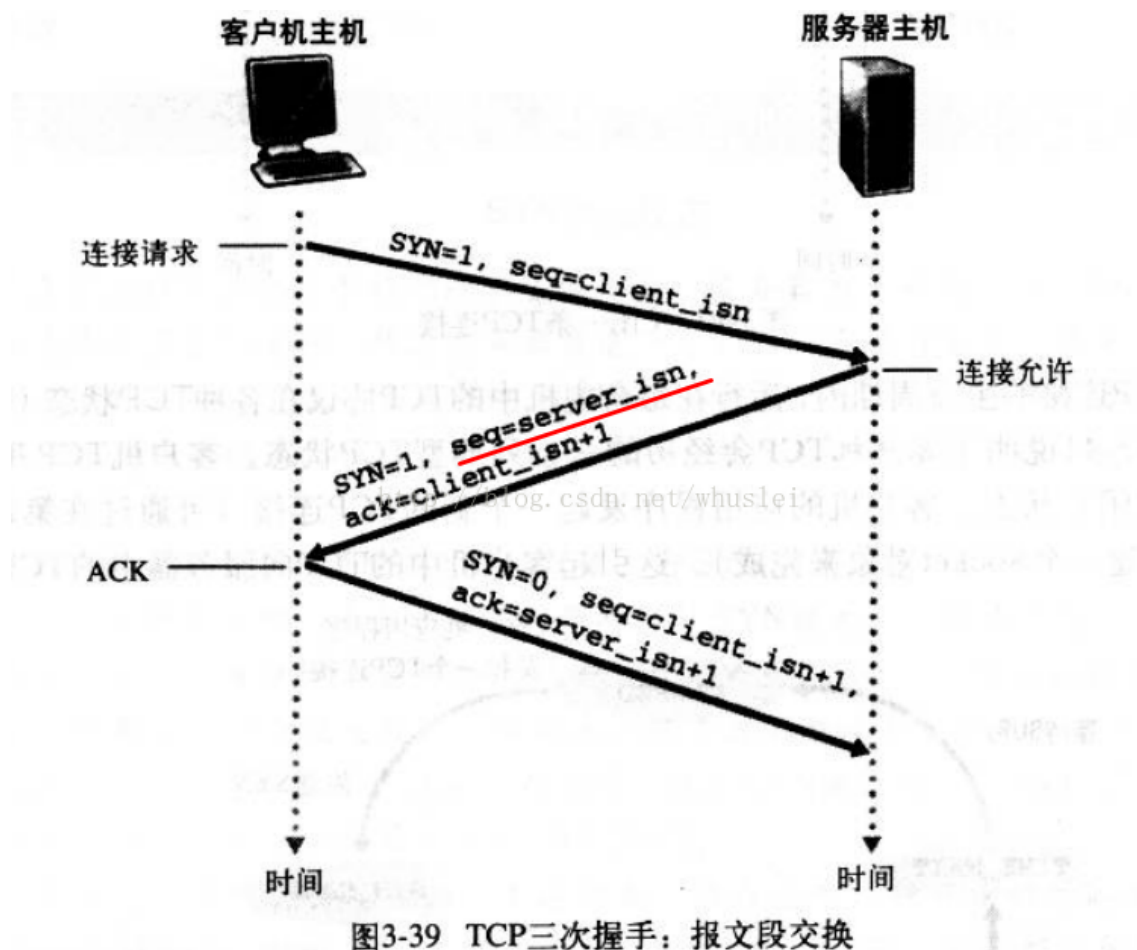
$\text{RcvWindows} = \text{RcvBuffer} - [\text{LastByteRcvd} - \text{LastByteRead}]$

发送方: $\text{LastByteSent} - \text{LastByteAcked} \leq \text{RcvWindow}$

LastByteSent - *LastByteAcked*即为发送方发包的长度。



- 连接建立：三次握手
注意随机数产生，用于确认连接是否合法



• 连接释放：四次挥手

1. 客户端发送 $FIN=1$ (也可以是服务端)，说明客户端没有数据要发送了，但是服务端还是可以发数据过来
2. 服务端发送ACK确保收到消息，同时也可以继续发送data包
3. 服务端确定数据发完，则发送 $FIN=1$ 给客户端
4. 客户端接收 FIN ，发送ACK，经过2MSL(最大报文段生存时间)才到close态，确保ACK发到服务端。

为什么连接的时候是三次握手，关闭的时候却是四次挥手？

简单地说，建立连接是同步的，而关闭是异步的。

TCP拥塞控制！

[Top]

TCP选择端到端的拥塞控制，因为IP层不向端系统提供显示的网络拥塞反馈。

三个问题：

1. TCP发送方如何限制向连接发送流量的速率？
2. 如何感知路径上存在阻塞？（超时和3个冗余ACK）
3. 采用何种算法来改变发送速率？

发送方跟踪额外变量**拥塞窗口** $cwnd$ ，满足

$$lastByteSent - lastByteAcked \leq \min\{cwnd, rwnd\}.$$

1. 慢启动

- TCP连接开始, `cwnd` 初始设置为MSS(最大报文段长度)的较小值, 则初始发送效率为 MSS/RTT
- `cwnd` 以1个MSS开始, 当传输的报文段首次被确认就增加1个MSS ($1 \Rightarrow 2 \Rightarrow 4 \dots$), 呈现指数级增长
- 存在超时导致的丢包事件, `cwnd` 设置为1, 并将阈值 `ssthresh` 设置为 `cwnd/2`, 重启慢启动过程。

当 `cwnd` 大于等于阈值, 结束慢启动转移到**拥塞避免模式**

- 丢包检测到3个冗余ACK, 执行快速重传并进入到**快速恢复状态**。

补充: TCP分岔 => 优化云服务性能

2. 拥塞避免

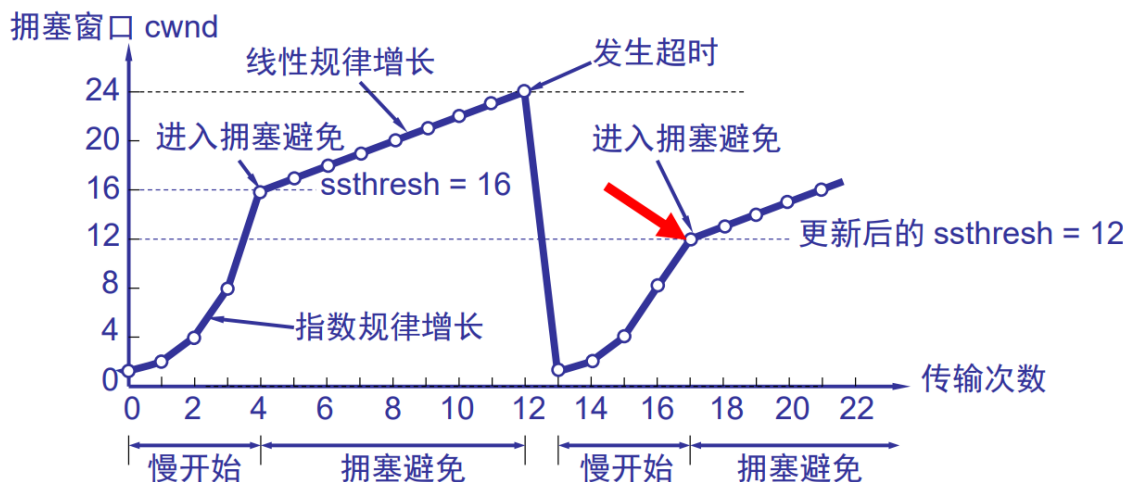
整个RTT将 `cwnd` 增加一个MSS。即每个达到ACK增加 $MSS / cwnd(old) * MSS$ 。

对于丢包或者接收3个冗余ACK的场景, 处理方式与慢启动一致。

3. 快速恢复(推荐, 非必需)

对于缺失报文段, 收到的每个冗余ACK, `cwnd` 值加一个MSS。最后当收到丢失报文段的ACK时, 降低 `cwnd` 进入拥塞避免状态。存在两个版本, 其中TCP Reno综合了快速恢复。


图1: 超时; 图2: 出现3个冗余ACK



Reno总结:

1. `cwnd` 小于门限值 `threshold`，慢启动，`cwnd` 指数增长
2. `cwnd` 大于门限值 `threshold`，拥塞避免，`cwnd` 线性增
3. 收到三个冗余ACK，`threshold` 设为 `cwnd` 的1/2，`cwnd` 改为(门限值+3) (3个ACK)，然后线性增
4. 超时，`threshold` 设为 `cwnd` 的1/2，`cwnd` 设为1个MSS，指数增长

TCP拥塞控制被称为"加性增，减性乘"，在图中呈现锯齿状。

函数公式 ：一条TCP连接的平均吞吐量为 $\frac{1.22 \times MSS}{RTT\sqrt{L}}$ ，其中L为丢包率。

补充：谷歌QUIC协议(很强)。

4.网络层：数据平面

[Top]

网络中的每一台主机和路由器都有网络层部分。

概述

[Top]

数据平面：从路由器输入链路向其输出链路转发数据报

控制平面：协调本地的每路由器转发，使得数据报沿着源和目的地主机之间的路由器路径最终进行端到端传送。

路由器不运行在应用层和运输层协议上。

两种重要的网络层功能：

- 转发：当一个分组到达某路由器的一条输入链路时，该路由器必须将该分组移动到合适的输出链路。是数据平面的唯一功能
- 路由选择：确定分组从源到目的地所采取的端到端路径的网络范围处理过程。网络层决定分组采用的路径（路由选择算法）

转发表：每台路由器中的关键元素。路由器检查到达分组首部的一个或多个字段值，使用这些**首部值**在转发表中索引，以此来转发分组。

控制平面包含传统方法和SDN方法。

网络服务模型

[Top]

因特网网络层提供**尽力而为服务**，与适当带宽供给相结合被证明超过"足够好"。

路由器工作原理

[Top]

路由器包含4个组件：

- 输入端口：在路由器中执行终结入物理链路的物理层功能，与位于链路远端的数据链路层交互执行数据链路层功能，执行查找功能，**通过查询转发表决定路由器的输出端口**
- 交换结构：将输入端口连接到路由器的输出端口
- 输出端口：存储从交换结构接收的分组，通过执行必要的链路层和物理层功能在输出链路中传输这些分组
- 路由选择处理器：执行控制平面功能(第5章具体介绍)

输入端口处理和基于目的地转化

[Top]

1. **IP地址前缀匹配**（最长前缀匹配规则），结合硬件设计和快速查找算法实现纳秒级的执行时间。通过查找确定了某分组的输出端口，则该分组能够发送进入交换结构，但是有可能被暂时阻塞，此时需要在输入端口处排队。
2. **交换**：交换结构位于路由器的核心部位

- 经内存交换：在CPU(路由选择处理器)的直接控制下完成，分组通过**中断方式**向CPU发出信号，然后被复制到处理器内存中。处理器从首部提取目标地址，在转发表中找出适当的输出口，将分组复制到输出端口的缓存中。

‡ 不能同时转发两个分组

- 经总线交换：经一根共享总线传输分组，让输入端口为分组预先计划一个交换机内部标签，指示输出口，只有与该标签匹配的端口才能保存该分组。

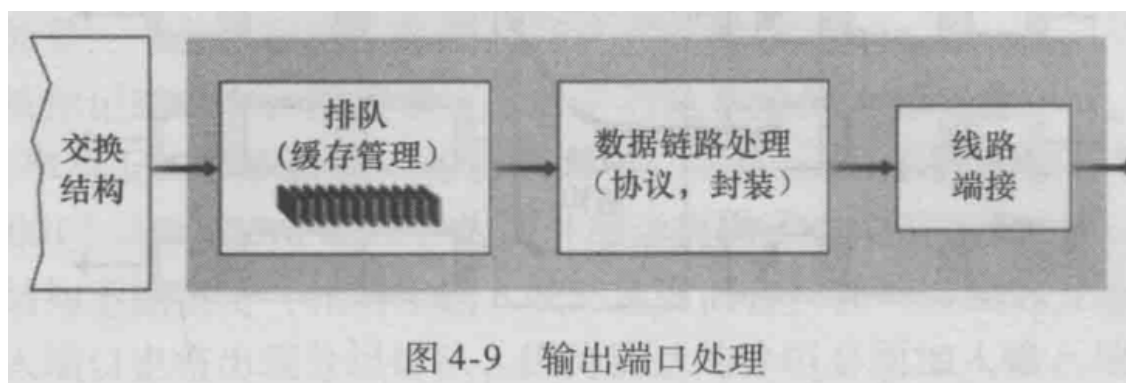
‡ 一次仅一个分组跨越总线，路由器带宽受总线速率影响

- 经互联网网络交换：

由2N条总线组成互连网络，交换机控制器闭合总线A和Y交叉点，然后端口A在其总线上发送该分组，仅由总线Y接收。能够并行转发多个分组，是非阻塞的。

- 三级非阻塞交换策略

3. 输出端口处理：



队列增长，路由器的缓存空间耗尽后，出现**丢包**。

i. 输入排队

典型代表：输入排队交换机中的**线路前部(HOL)阻塞**，即在一个输入队列中排队的分组必须等待通过交换结构发送，因为它被位于线路前部的另一个分组所阻塞。

有论证指出，由此输入链路分组到达其容量的58%，则输入队列会无限制地增大

ii. 输出排队

交换结构速度大于输出端口发送分组的速度。

弃尾策略或者标记丢弃分组地首部来给发送法提供一个拥塞信号。统称为**主动队列管理(AQM)**，使用较多的为随即早期检测。

路由器缓存容量，一般为 $RTT * C$ ，即**平均往返时延乘以链路容量**。

iii. 分组调度

- FIFO
- 优先权排队
- 循环和加权公平排队，简单的循环排队规则类似于轮询，对每个类轮番检测，并且基于保持工作排队规则。加权公平排队(WFQ)，每个类的轮询占比对应该类的加权。



- 版本号：4比特，规定数据报的IP协议版本，路由器确定如何解释IP数据报的剩余部分
- 首部长度：4比特，IPv4数据报包含一些可变数量的选项，确定IP数据报中载荷实际开始的地方。大多数IP数据报不包含选项
- 服务类型(TOS)：区分不同类型的IP数据报。例如将IP电话应用和非实时流量(如FTP)区分开来是有必要的
- 数据报长度：16比特，IP数据报的总长度，理论最大长度即为 $2^{16}=65536$ 字节。
- 标识，标志，片位移（8字节为单位）：用于分片
- 寿命：生存时间即TTL，每一台路由器处理后该字段减少1，为0时丢弃，以确保不会永远在网络中循环
- 上层协议：指示IP数据报的数据部分应交给哪个特定的运输层协议，如6=>TCP, 17=>UDP
- 首部校验和：将首部中每两个字节作为一个数，然后反码求和放入该字段，路由器计算首部校验和并判断是否一致。另外判断后路由器需再次计算并放回，因为TTL等字段会发生变化。

🔗：为什么TCP, IP都想执行差错检测？

i. IP只对首部进行检测，而TCP是对整个报文

ii. TCP/UDP 与 IP不一定属于同一个协议栈。（例如，TCP能够运行在ATM上，而IP能传给其他运输层）

- 源和目的地址
- 选项：允许IP首部被扩展，由于使得路由器IP处理所需的时间变化可能很大，IPv6去掉了该字段
- 数据(有效载荷)

IPv4数据报分片

[Top]

为什么分片？不是所有的链路层协议都能承载相同长度的网络层分组，受到了链路层帧能承载的最大数据量**最大传送单元 (MTU)** 的限制。

分片：将IP数据报中的**数据部分**分片成两个或更多个较小的IP数据报，用单独的链路层帧封装这些较小的IP数据报

端系统负责将分片后的数据报重组。

如何接收？

发送主机将它发送的每个数据报的**标识号**加1，目的主机通过检查数据报的标识号确定哪些数据是**同一较大数据报**的片。

如何确定丢失？

为最后一个片的**标志比特**被置为0，其他片都是1。另外为了确定丢失了哪个片且依然能按顺序重组，使用**偏移字段**指定该片在**初始IP数据报**的哪个位置。

IPv4编址

[\[Top\]](#)

主机与物理链路之间的边界叫做接口。

IP要求每台主机和路由器拥有自己的IP地址。

每个IP地址长度为32比特，总共有 2^{32} 个(约40亿)，采用点分十进制记法。一个接口的IP地址的一部分需要由其连接的子网决定。

子网：分开主机和路由器的每个接口，产生隔离的网络岛，使用接口端接这些隔离的网络的端点。每一个这样隔离的网络就叫做子网。

子网中的设备接口具有相同的网络号部分。

IP地址为子网分配一个地址如 `223.1.1.0/24`，其中/24称为**子网掩码**，左侧24bit定义了子网地址，其余xxx可分配。

子网划分：从主机号中借用一部分作为子网号，在网络号的子网号相应的位置全置1，主机号相应的位置全置0，则得到子网掩码，例如上面的就是 `255.255.255.0`。

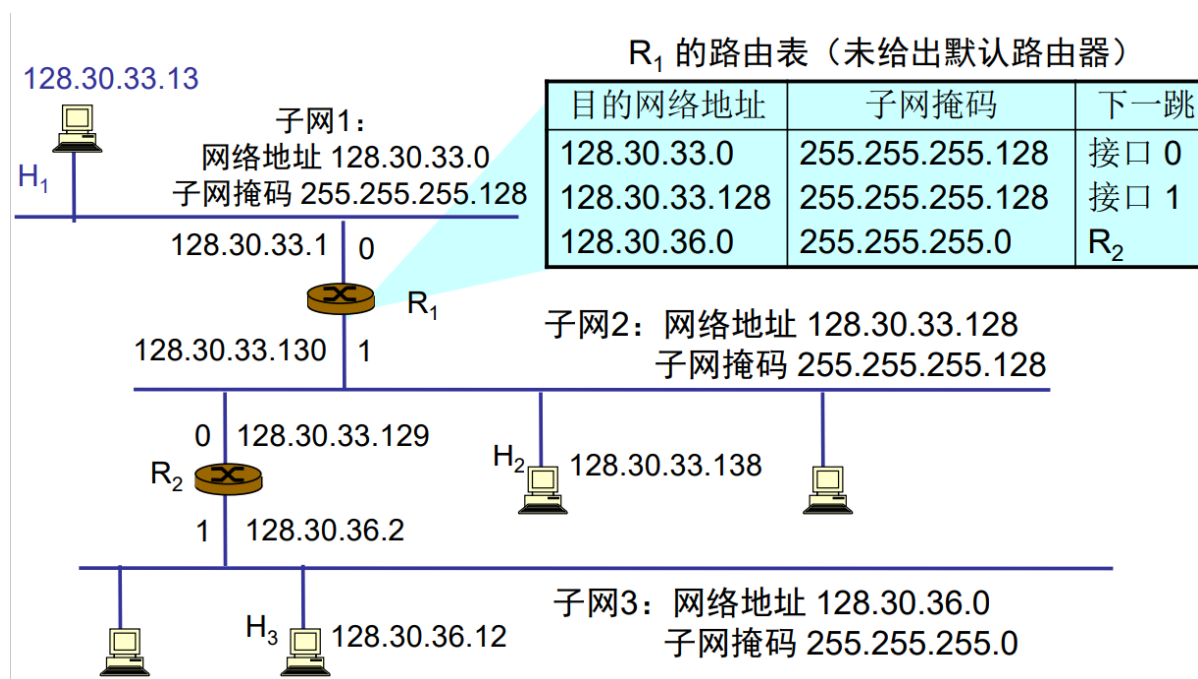
子网掩码的作用：对外隐藏子网的存在，对内指示网络号和子网号的位置。

网络地址：主机号全为0，网络号不变

广播地址：主机号全为1

这两个地址都不能被分配给具体的接口。

划分子网后的分组转发过程：



假设为主机H1发送分组给H2

1. H1检查H2是否在自己所在的子网中，如果是则可以直接发送。具体过程为H1的**子网掩码**与H2的IP地址作"与"运算，如果等于H1的**网络地址**则说明在一个子网中

2. 发现不在一个子网中，所以H1将分组交给子网的路由器R1由其转发
3. 路由器收到分组后，依次与路由表中的项目的子网掩码进行"与"运算，然后判断是否与对应项目的网络地址相等，如果是则由对应的接口发送到对应的子网中

¶ 因特网的地址分配策略：**无类别域间路由选择 (CIDR)**

子网寻址，将32比特的IP地址划分为两部分，形如 `a.b.c.d/x`，其中x指示地址第一部分的比特数。x最高比特构成IP地址的**网络部分**，被称为该地址的**前缀**。

p220实践原则：

路由聚合、路由摘要、地址聚合：使用单个网络前缀通告多个网络的能力

在CIDR采用前，IP采用**分类编址**，分A,B,C三类。但是对C类(/24)子网仅能容纳254台主机，太少，B类(/16)又可以容纳65534台主机，太多，会使其地址空间被迅速消耗，而已经被分配的地址空间的利用率很低。

IP广播地址：255.255.255.255；主机发出一个目的地址为该地址的数据报时，该报文会交付给同一个网络的**所有主机**，路由器也会有选择地向邻近子网转发该报文。

🔗 **如何从地址块中分配到一个地址**

1. 获取一块地址
 - ISP本身从ICANN得到一块地址
 - 从ISP获取一组地址，比如ISP的地址块为xxx.xxx.xxx.0/20，如果要分成8块，则每个组织的地址块应该为xxx.xxx.xxx.0/23，多出来的3bit用于区分8个子网。
2. 获取主机地址

动态主机配置协议(DHCP)：又称**即插即用协议**和**零配置协议**，允许主机自动获取一个IP地址，可以通过配置DHCP使得给定主机每次连网时分配相同的IP地址或者是一个临时的IP地址。

DHCP是客户-服务器协议，每个子网中都有一台DHCP服务器或者DHCP中继代理的路由器。

新到达的主机获取地址的4个步骤：

- **DHCP服务器发现**：使用发现报文，客户(端口68)在UDP分组中向端口67发送该发现报文。而客户主机此时**没有源地址也不知道目的地址**，所以生成包含DHCP发现报文的IP数据报，使用广播目的地址**255.255.255.255**，源地址为**0.0.0.0**。DHCP客户将该IP数据报传递给链路层，然后链路层将该帧广播到所有与该子网连接的节点
- **DHCP服务器提供**：用DHCP提供报文作出响应，仍然使用的是广播地址，因为子网中可能存在几个DHCP服务器，该客户也许会发现他处于能在几个提供者之间选择的优越位置)。每台服务器提供的报文包含由收到发现报文的**事务ID**，服务器本身的IP地址，向客户推荐的**IP地址、网络掩码**以及**IP地址租用期**（IP地址的有效时间量）。服务器租用期一般为几小时或者几天
- **DHCP请求**：客户从一个或者多个服务器提供中选择一个，并向选中的服务器用请求报文响应（所以仍然是广播地址），回显配置参数
- **DHCP ACK**：服务器用ACK报文对请求报文进行响应，证实所要求的参数

¶ 缺点：每当节点连接到新的子网，要从DHCP获取新的IP地址（即是变化的），不能维持与远程应用之间的TCP连接（解决方案：**移动IP**：对IP基础设施的扩展，允许移动节点在网络之间移动时使用其单一永久的地址）

网络地址转换(NAT)

[\[Top\]](#)

在家庭网络场景中，例如编址10.0.0/24，地址空间10.0.0/8保留，用于该专用网络，即10.0.0/24仅在给定的网络中有效。

NAT使能路由器对外界的行为如同一个具有**单一IP地址**的单一设备，对外界隐藏家庭网络的细节。运行着**DHCP服务器**并通过**NAT转换表**的映射关系，将家庭网络中的 `ip/port` 映射为**单一IP和唯一的port**，所以NAT协议可以支持超过60000个并行使用路由器广域网一侧单个IP地址的连接。

背景：32比特的IP空间即将用尽，并借此机会强化IPv4的其他方面。

数据报格式：

◆ IPv6数据报格式

版本	流量类型	流标签	
有效载荷长度		下一个首部	跳限制
源地址（128比特）			
目的地址（128比特）			
数据			

◆ 无检查和，中间结点无需计算

◆ 中间结点不再负责分片和重组，由端结点负责

◆ 首部长度固定，加速中间结点转发速度

变化及特点：

1. **扩大的地址容量**：IP地址长度从32bit -> 128bit。引入任播地址，可以使数据报交付给一组主机中的任意一个。
2. **40字节首部**：简化高效
3. **流标签**：给发送方特殊要求进行处理的流及特殊流的分组加上标签

定义字段：

- **版本**：4比特标识IP版本号，6标识IPv6
- **流量类型**：区分不同类型的IP数据报，与TOS字段类似
- **流标签**
- **有效载荷长度**：16比特，给出跟在首部后面的实际数据报的字节数量
- **下一个首部**：标识数据报中的内容需要交付给哪个协议(TCP or UDP)
- **跳限制**：转发数据报的每台路由器将该字段的内容减1，如果跳限制计数为0，则丢弃该数据报。（类似TTL？）
- **源/目的地址**
- **数据**：IPv6数据报的有效载荷部分，数据报到达目的地，则该有效载荷从IP数据报移出，交给下个首部字段中指定的协议。

"删除"的字段：

- **分片/重新组装**：v6不允许在中间路由器上进行分片和重新组装。这种操作只能在源与目的地执行。如果路由器收到的数据报太大而不能转发到链路上，则路由器会丢掉该数据报，并向分组发送一个"分组太大"的ICMP（Internet Control Message Protocol，网络层协议）差错报文，让发送方重新发送较小数据的报文。（为什么去除？分片和重组是一个耗时操作，该功能从路由器中删除，并放在端系统中，**加快网络中IP转发速度**）
- **首部检验和**：运输层和链路层都执行了检验操作，设计者觉得该项功能冗余。另外计算检验和也很耗时
- **选项**：不再是标准IP首部的一部分，但可能出现在IP分组的"下一个首部"

IPv4到IPv6的迁移：

建隧道：

将两台IPv6路由器之间的中间IPv4路由器的集合称为一个隧道。通过将整个IPv6数据报放入IPv4数据报的有效载荷中，实现隧道两端IPv6路由器的数据传输。

通用转化和SDN(软件定义网络)

[\[Top\]](#)

匹配加动作转发表

OpenFlow标准

5.网络层：控制平面

[\[Top\]](#)

控制沿着从源主机到目的主机的端到端路径间的路由器如何转发数据报，而且控制网络层组件和服务如何配置和管理。

两种可能的方法

- 每路由器控制：每台路由器都运行一种路由选择算法，并包含转发和路由选择功能
- 逻辑集中式控制：SDN采用。由逻辑集中式控制器计算并分发转发表以供每台路由器使用

路由选择算法

[\[Top\]](#)

目的：从发送方到接收方的过程中确定一条通过路由器网络的好的路径(最低开销的路径)。

分类

集中式还是分散式：

- 集中式路由选择算法：用完整的、全局性的网络知识计算出从源到目的地之间的最低开销。具有关于连通性和链路开销方面的完整信息。具有全局状态信息的算法常被称为**链路状态(LS)算法**。
- 分散式路由选择算法：以迭代、分布式的方式计算最低开销路径

静态还是动态

- 静态路由选择算法：路由随事件的变化非常缓慢，通常由人工调整
- 动态路由选择算法：随着网络流量负载或拓扑发生变化而改变路由选择路径

负载敏感还是负载迟钝：是否明确反映出底层链路的当前拥塞水平。

链路状态路由选择(LS)算法

[\[Top\]](#)

Dijkstra 算法求定点到其它点的最短路径。

LS算法终止后，对于每一个节点，我们都能得到从源节点沿着它的最低开销路径的**前驱节点**。再利用栈即可得到正确的路径。

链路开销非对称时，会出现链路震荡，即一开始有顺时针方向的最短路径，下一次LS算法运行时又得到的是逆时针的。一种方法是让每台路由器通过发送**链路通告**的方式来实现算法执行时机的不同。因为如果是单纯的开始执行时间不同，由于周期性还是会出现震荡。

eg: OSPF 开放式最短路径优先 (Open Shortest Path First)

距离向量路由选择算法(DV)算法

[Top]

迭代的，异步的，自我终止的，分布式的算法]

Bellman-Ford 方程：

$$d_x = \min_v \{c(x, v) + d_v(y)\}, \text{ 其中 } v \text{ 为 } x \text{ 的每一个邻接节点}$$

简单来说，是通过得到邻居到指定点的链路开销来不断更新源点到目的点的链路开销(这个过程是异步的、迭代的，且由于它不是全局获取即分布式的)，最终能收敛到目的点的最小链路开销。

eg: RIP协议(Routing Information Protocol,路由信息协议)

链路开销改变与链路故障

P252，当链路开销增大后，更新的迭代次数将会变得很大，甚至趋近于"无穷计数"。

毒性逆转：可用于解决两个直连的邻居节点出现上述问题的现象

因特网上的距离向量算法-RIP协议

[Top]

- 链路上的费用定义为1，即多少个路由器或多少跳
- 一条路径的最大费用限制为15
- 选路更新消息每30s在邻居之间以RIP响应报文(RIP通告)的形式进行交换
- 180s心跳检测，检测邻居是否离线
- 运行在UDP上的应用层协议(端口520)

因特网中自治系统内部的路由选择：OSPF

[Top]

之前的将网络只看作一个互联路由器的集合太过于简单化，主要原因有两个：

- 规模：路由器数目过大，迭代算法可能永远无法收敛
- 自治：每个ISP有自己的路由器网络，也会有自己对应的路由选择算法，且对外隐藏网络内部组织细节。

解决方案：自治系统(AS)

开放最短路优先(OSPF)

[Top]

OSPF是一种链路状态协议，使用洪泛链路状态信息和Dijkstra最低开销路径算法，能层次化地将AS配置成多个区域，其每个区域运行自己的OSPF。

- 每个区域有32bit区域标识符，一般一个区域的路由器个数不超过200
- 主干区域用于连通AS中其它区域，即为其提供路由选择，标识符为0.0.0.0

AS间的路由选择：BGP(Broder Gateway Protocol)边界网关协议

[Top]

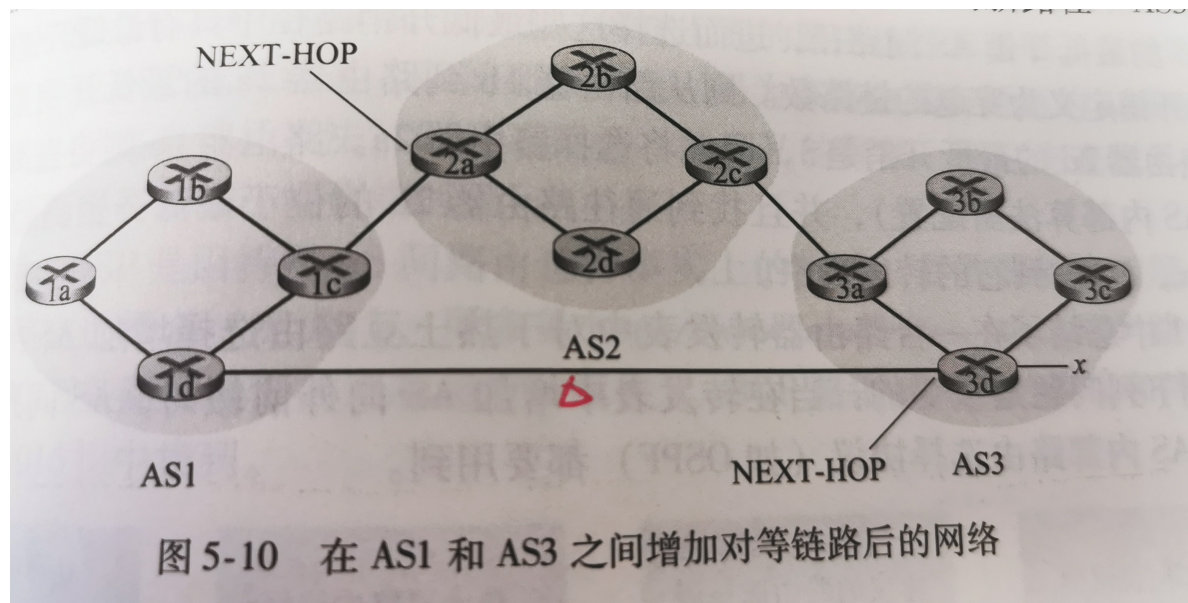
BGP中，分组是路由到CIDR化的前缀，例如138.16.68/22共1024个IP地址，路由器的转发表将具有形式为 (x, I) 的表项， x 是类似上述的前缀， I 是路由器的某一接口号。

BGP为每台路由器提供了一种完成以下任务的手段：

- 从邻居AS获得前缀的可达性信息
- 确定到该前缀的"最好的"路由

AS中每一台路由器要么是位于AS边缘的**网关路由器**，要么是**内部路由器**。

在BGP中，每对路由器通过使用179端口的半永久TCP连接交换路由选择信息，跨越两个ASD的BGP连接成为eBSP(外部BGP)，在相同AS中的两台路由器之间的BGP会话称为iBGP(内部BGP)。



考虑AS3向AS1发送BGP报文，则为"AS2 AS3 x"和"AS x"两条。

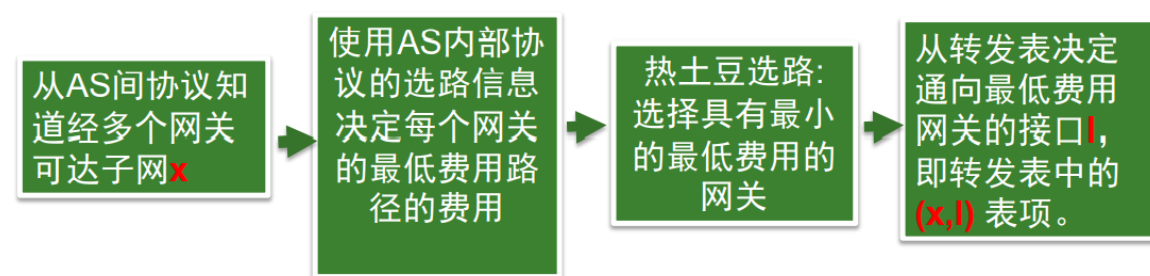
确定最好的路由

BGP术语：前缀及其属性称为**路由**。

BGP属性：

- **AS-PATH**：包含已经通过的AS的列表，前缀每经过一个前缀即将对应的ASN加入到现有列表起始的位置
- **NEXT-HOP**：AS-PATH 起始的路由器接口的IP地址
- 前缀

热土豆路由选择：



即对于路由器，尽可能减小在自己AS中的开销，而忽略AS之外的端到端开销的其他部分

路由器选择算法：

对于两条及以上的路由，采用以下**消除规则**：

1. **本地偏好值**(完全取决于AS的网络管理员)最高的先选
2. 从余下的路由中使用距离向量算法算则**AS跳数最小**的路由
3. 从余下的路由中使用**热土豆算法**
4. 仍留下多条路由，则通过**BGP标识符**选择

IP任播

[Top]

被DNS系统广泛使用用于将DNS请求指向最近的根DNS服务器

路由选择策略:

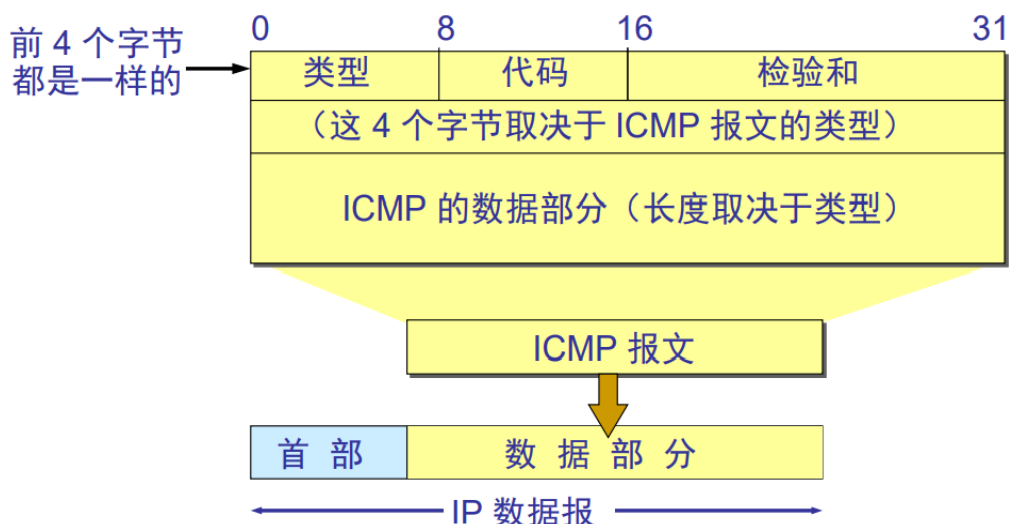
[Top]

P262 客户/提供商网络

ICMP 因特网控制报文协议

[Top]

报文格式:



一般用于:

1. 差错报告
2. 指示是否分片及分片大小: 设置DF位
3. Traceroute: 通过设定TTL值

6.链路层和局域网

[Top]

链路层概述

[Top]

节点: 主机和路由器

提供的服务:

[Top]

- 成帧: 用链路层帧封装网络层数据报, 将数据加上头部和尾部, 帧头部用MAC地址标识源和目的
- 链路接入: **媒体访问控制 (MAC)** 协议规定帧在链路上传输的规则
- 可靠交付
- 差错检测和纠正
- 流量控制
- 全双工和半双工

链路层的主体部分在**网络适配器** (网卡, 半自主单元) 中实现。

差错检测和纠正技术

[Top]

差错检测和纠正技术越复杂, 出现未检出比特错误的概率就越小, 但导致的开销就越大, 是一个trade-off。

奇偶校验

[Top]

单个奇偶校验位=>二维奇偶校验：可以通过行列索引发生错误的比特并纠正，能检测(但不能纠正)一个分组中的两个比特错误的任何组合。

检验和(checksum)方法

[Top]

将数据的字节作为16比特整数来求和并取反码作为checksum字段，用户计算接收数据(包括checksum)的和，通过结果是否为全1来检测检验和。（3.3节有讲

链路层用专用的**硬件**执行更复杂的**CRC操作**。

循环冗余检测(Cyclic Redundancy Check CRC)

[Top]

建议看计组ppt复习

- 增加冗余码（校验位） r 位
有效信息位 k (书中标识为 D)位，则满足 $N = k + r \leq 2^r - 1$ ，取满足条件的最大的 r 值
- 发送方和接收方约定位数为 $(r + 1)$ 位的生成多项式 $G(X)$

$G(X)$ 满足的条件：

- 最高位最低位不为0
- CRC码任何一位发生错误后，被生成多项式做除的余数不为0
- 不同位发生错误，模2除运算后余数不同，因此可以循环校验
- 对不为0的余数继续进行模2除运算使余数循环

CRC-32(即32位多项式)用于以太网协议。每个CRC标准都能检测小于 $(r + 1)$ 比特的突发差错。

多路访问链路和协议

[Top]

- 点对点链路
- 广播链路

多路访问问题：协调多个发送和接收节点对一个共享广播信道的访问

1. 信道划分协议

[Top]

时分复用(TDM)和频分复用(FDM)：

比较简单

消除了碰撞，保证了公平，但是对每个节点有限制，前者是 $R/Nbps$ ，后者是 R/N 的带宽。

码分多址(CDMA)：每个节点分配一种不同的编码，每个节点就用它唯一的编码来对它发送的数据进行编码以达到不同的节点能够同时传输。用于蜂窝电话

2. 随机访问协议

[Top]

一个节点总是以信道的全部速率进行发送，但因此会发生碰撞。

ALOHA(随机接入协议)

[Top]

- 纯ALOHA

$P(\text{给定节点成功概率}) = p \times (1 - p)^{2(N-1)}$ ，取最合适 p ， n 趋于无穷，取极限得 $1/(2e) = 0.18$

- 时隙ALOHA
 - 所有帧大小相同
 - 时间被划分为长度为 L/Rs 的时隙

- 节点只会时隙开始传输
- 节点是同步的，即每个节点知道时隙何时开始
- 能检测到多个节点在一个时隙内传送的冲突

操作：

- 节点有新帧要发送时，等到下一个时隙传输
- 若没有碰撞则节点成功传输它的帧，且不需要重传
- 如果有碰撞，该节点在时隙结束之前检测到，并以概率 p 在后续每个时隙重传它的帧

特点：

- 单个节点能以全速 R 连续传输
- 高度分散
- 但是碰撞会引起时隙的浪费
- 节点需要在分组传送时间以内检测到碰撞

$P(\text{给定节点成功概率}) = p \times (1 - p)^{(N-1)}$, 取最合适 p , n 趋于无穷, 取极限得 $1/(e) = 0.37$

载波侦听多路访问(CSMA)

[\[Top\]](#)

- 载波侦听：传输前先监听信道是否空闲
- 碰撞检测：检测到另一个节点正在传输干扰帧则停止传输

分类：

- 非坚持CSMA/时隙非坚持CSMA
- 1坚持CSMA：持续侦听到信道空闲时，将数据发出。若发生冲突，站点就等待一个随机长的时间，然后重新开始
- P坚持CSMA：侦听到信道空闲时，以概率 p 将数据发出，以概率 $(1-p)$ 延迟一段时间 τ （网络中最远的端到端传播时延），从而在一定程度上避免空闲期多个节点同时发数据而产生的冲突。处理冲突方式同1坚持

实际网络中常选择1坚持

CSMA/CD

[\[Top\]](#)

- 强化碰撞：碰撞后除立即停止发送数据外，还要继续发送若干比特的人为干扰信号
- 争用期：最先发送数据帧的站，在发送之后至多经过 2τ 即可知道是否遭受碰撞

以太网CSMA/CD算法(适配器角度)：

1. 网卡从网络层获得数据报，创建数据帧
2. 如果网卡侦听到信道空闲，开始帧传输，否则等待，直到信道空闲
3. 如果网卡发送的整个帧没有探测到其它传输，完成帧发送
4. 检测到另一个传输，终止并发送干扰信号
5. 终止后，进入二进制指数回退，返回第二步

第 n 次碰撞，随机从 $0, 1, 2, \dots, 2^{n-1}$ 选择 K ，网卡实际等待 $K \times 512\text{bit}$ 时间

碰撞越多，间隙越大

$$\text{效率} = \frac{1}{1 + 5d_{prop}/d_{trans}} \quad (\text{不知道怎么算的} hh)$$

3. 轮流协议

[Top]

问题背景：

- 信道划分协议：只有一个活动节点，又只能分到1/N的带宽
- 随机访问协议：碰撞带来的开销

轮询协议

主节点轮询每个节点，告诉其能传输的帧的最多数量

缺点：轮询开销，等待时间，单点失效

令牌传递协议

令牌作为特殊帧以某种固定次序进行交换

缺点：节点故障，未释放需要恢复

交换局域网

[Top]

MAC地址：标识网络适配器

- 48bit，前24bit由IEEE分配管理，后24bit由厂商自行管理
- 平面地址，在不同的网络间迁移时不会改变(与IP的不同)

地址解析协议ARP：根据IP地址获取其MAC地址

ARP即插即用，ARP表自动建立

- 每一个IP节点(主机、路由器)都有ARP表
- IP/MAC映射 <IP;MAC;TTL>

将目的MAC地址设为广播地址FF-FF-FF-FF-FF-FF，获得子网中所有适配器的MAC地址并更新ARP表。

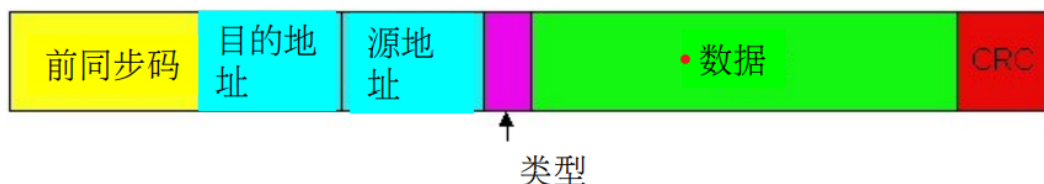
🔗 搞清楚不同子网间数据报传输的过程

以太网

[Top]

- 总线式以太网
- 交换式以太网：基于交换机

帧结构：



- 前同步码(8字节)：前7字节的值为10101010，用于“唤醒”接收的适配器，最后一个字节为10101011。用于同步接收方和发送方的时钟
- 数据字段(46~1500字节)：以太网的最大传输单元为1500字节，超过则需要分片，小于则需要填充，通过IP数据报首部的长度字段可去除填充部分
- 目的地址/源地址（6字节）
- 类型字段(2字节)：多路分解（以太网可用多种网络层协议）
- CRC(4字节)：用于检错

几个定义：

- 拥塞信号：确保传输者都能检测到碰撞而传输的信号；48比特长

- 比特时间

CSMA/CD协议的以太网不能进行全双工通信而只能进行双向交替通信

- 争用期长度

10Mb/s以太网取51.2us为争用期的长度，即发送64字节(最短有效帧长)的时间
在争用期内可发送512bit，如果前64字节没有发生碰撞则后续就不会发生碰撞

- 最短有效帧长

发生碰撞在前64字节之内

长度小于64字节的帧是由于冲突而异常终止的无效帧

传输以太网的物理层

10 BASE 5(T)

10: 传输速率为10Mbps

5: 最远传输距离为500m，为同轴电缆

T: 双绞线

信号编码

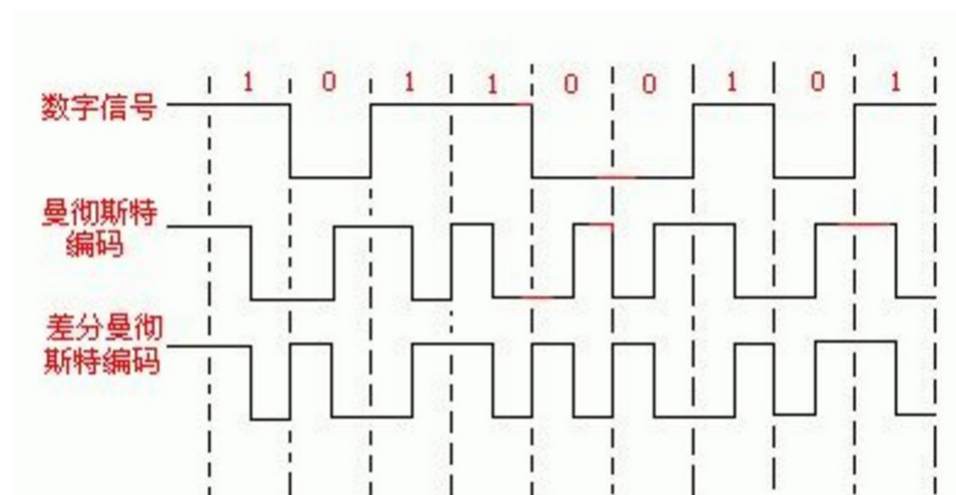
[\[Top\]](#)

曼彻斯特编码:

位中间 电平从高到低跳变表示"1";

位中间 电平从低到高跳变表示"0"。

差分曼彻斯特编码: 每位开始时**有无跳变**表示"0"或"1", 有跳变为"0", 无跳变为"1"



知乎 @桃子

针对第一位的画法（既看不到开始时的跳变）：

如果在最初信号的时候，即**第一个信号**时：

如果中间位电平从低到高，则表示0；

如果中间位电平从高到低，则表示1；

后面的（从第二个开始）就看每个信号位开始时**有没有跳变**来决定。

链路层设备

- 存储转发以太网帧
- 查看输入帧的MAC地址，通过CSMA/CD选择性地输出到一个或多个输出链路
- 对主机是**透明**的
- **即插即用，自学习**

交换机：多路同时传输

过滤：决定一个帧是否转发还是丢弃

转发：决定一个帧应该被导向哪个接口

1. 记录发送方的输入接口和MAC地址
2. 使用目标MAC地址检索交换表
3. 如果表中没有目标地址对应的表项，**广播该帧**，即向所有非来源接口转发数据帧
4. 如果检索到目标地址的接口**是来源接口**，**丢弃该帧**执行过滤功能
5. 检索到目标地址的接口且不是来源接口，说明该帧需要被转发到与该接口相连的局域网网段，交换机将该帧放到该接口的输出缓存中

自学习：

1. 交换表初始为空
2. 对每个接口接受到的每个入帧，该交换机在其表中存储<源MAC地址，接口，时间>
3. 在一段时间(老化期)后，交换机没有接受到该地址的帧，则在表中删除该地址
4. 接受到表中含有的源地址的帧，则更新时间

📺PPT多交换机互联和自学习动画示例

特点：

- 主机直接连接到交换机
- 交换机缓存数据报
- 每条链路都采用以太网协议 -> **消除碰撞，最大聚合宽带是该交换机所有接口速率之和**
- **异质的链路 -> 将链路彼此隔离**
- **管理**

补充了解：交换机毒化

网桥互联

存储转发设备，实现MAC层的LAN互联

需要了解相关概念，缺点，和集线器的对比。

存在**兜圈子**的问题：消耗网络资源

支撑树算法：即互连在一起的网桥在进行彼此通信后，就能找出原来的网络拓扑的一个**子集**。在这个子集里，整个连通的网络中**不存在回路**，即在任何两个站之间只有一条路径。

虚拟局域网VLAN

虚拟局域网（VLAN）是在局域网（LAN）的逻辑上划分成多个广播域，每一个广播域就是一个VLAN。

基于接口的VLAN：通过接口分组使得单一的交换机设备像多态交换机一样使用，由VLAN管理员配置哪些port对应哪个VLAN ID。

功能：

- 流量隔离
- 动态分组
- VLAN间数据转发

数据中心网络

[Top]

- 负载均衡：应用层路由，接收外部客户端请求，数据中心内处理负载
- 交换机、机架之间的丰富互联
 - 增加机架间的吞吐量
 - 增加冗余提高可靠性

回顾：Web页面请求的历程

[Top]

个人简单的归纳：

1. 接入到Internet，需要DHCP获得IP地址，DHCP请求报文的IP数据报由UDP承载被放入以太网帧中
2. 完成DHCP后，用户获得分配的IP地址，同时还有DNS服务器的IP地址
3. DNS和ARP：用户需要知道www.google.com对应的IP地址，因此需要生成DNS请求报文，使用ARP协议获得网关路由器的MAC地址
4. 通过域内路由(RIP,OSPF等)，域间路由BGP，多次转发最终IP数据报到达DNS服务器，后者响应生成DNS回答报文
5. 用户得到网页对应的IP地址，开始建立TCP连接，通过三次握手建立
6. 建立成功后发送HTTP请求报文，web server返回HTTP响应报文
7. 用户浏览器解析呈现Web页面

(细节需要看书，不过前面的知识有了，应该不难，只是整合到一块了)