

计算机网络 Computer Network

⚠ Warning

- 平时成绩 50% = 课程作业 7-8 次 + 课程实验 6 次 + 出勤+quiz 3-4 次
- 期末考试 50% (闭卷) 试卷为英文

💡 Tip

<https://zjucomp.net/docs/terms> 术语中英对照表

计算机网络 Computer Network

1. 引言

1.1 初识互联网

1.2 网络实例

1.2.1 网络分类 (按地域规模)

1.2.2 网络组成

 1.2.2.1 网络边缘

 1.2.2.2 网络核心

 1.2.2.2.1 接入网

 1.2.2.2.2 物理媒介

 1.2.2.2.3 网络核心两大功能

1.2.3 分组交换和电路交换

 1.2.3.1 分组交换 (也称包交换 packet switching)

 1.2.3.2 电路交换 (circuit switching)

 1.2.3.3 存储转发的报文交换 (Message switching)

1.3 协议和分层结构

1.3.1 协议设计目的

 1.3.1.1 统一标准

 1.3.1.2 模块独立

 1.3.1.3 分层结构

1.3.2 服务原语

1.3.3 服务与协议的关系

1.4 参考模型

1.4.1 OSI 参考模型

 1.4.1.1 物理层 (Physical Layer)

 1.4.1.2 数据链路层 (Data Link Layer)

 1.4.1.3 网络层 (Network Layer)

 1.4.1.4 传输层 (Transport Layer)

 1.4.1.5 会话层 (Session Layer)

 1.4.1.6 表示层 (Presentation Layer)

 1.4.1.7 应用层 (Application Layer)

1.4.2 TCP/IP 参考模型

 1.4.2.1 链路层/网络接口层 (Link Layer)

 1.4.2.2 互联网层/网际层 (Internet Layer)

 1.4.2.3 传输层 (Transport Layer)

 1.4.2.4 应用层 (Application Layer)

1.4.3 OSI 模型 TCP/IP 模型对比

1.4.4 模型与网络实例

1.5 计算机网络度量单位

1.5.1 速率/比特率(bit rate)

1.5.2 波特率(baud rate)

1.5.3 带宽(bandwidth)

1.5.4 包转发率(PPS)

1.5.5 时延(Delay)

- 1.5.6 往返时延 RTT(Round-Trip Time)
 - 1.5.7 时延带宽积
 - 1.6 网络安全威胁
 - 1.6.0.1 拒绝服务攻击(Denial-of-Service (DoS) attack)
 - 1.6.0.2 僵尸网络 (Botnet)
 - 1.6.0.3 如何防御
 - 1.7 标准化组织
 - 1.8 互联网发展史与启示
 - 1.9 总结
2. 物理层
- 2.1 物理层基本概念
 - 2.1.1 物理层功能
 - 2.1.2 物理层特性
 - 2.1.2.1 物理层机械特性
 - 2.1.2.2 物理层电气特性
 - 2.1.2.3 物理层功能特性
 - 2.1.2.4 物理层过程特性
 - 2.1.3 物理层标准及示例
 - 2.2 数据通信基础
 - 2.2.1 数据通信基础理论
 - 2.2.1.1 傅里叶分析
 - 2.2.1.2 有限带宽信号
 - 2.2.2 信道的最大数据传输速率
 - 2.2.2.1 奈魁斯特定理 Nyquist
 - 2.2.2.2 信噪比
 - 2.2.2.3 香农定理 Shannon
 - 2.2.2.4 信息量
 - 2.2.3 数据通信系统模型
 - 2.2.3.1 传输方式
 - 2.2.3.2 数据编码技术/调制 digital modulation
 - 2.2.4 频带传输
 - 2.3 传输介质
 - 2.3.1 分类
 - 2.3.1.1 导引性介质 guided transmission media
 - 2.3.1.1.1 磁介质 magnetic tape
 - 2.3.1.1.2 双绞线 twisted pair (网线)
 - 2.3.1.1.3 同轴电缆 coaxial cable
 - 2.3.1.1.4 光纤 fiber
 - 2.3.1.1.5 电力载波
 - 2.3.1.2 非导引性介质
 - 2.3.1.2.1 短波传输 (无线电波)
 - 2.3.1.2.2 微波
 - 2.3.1.2.3 光波
 - 2.3.1.2.4 红外线
 - 2.4 无线与卫星通信
 - 2.4.1 无线
 - 2.4.2 卫星通信
 - 2.5 多路复用技术
 - 2.5.1 频分复用(FDM)
 - 2.5.2 正交频分复用(OFDM)
 - 2.5.3 时分复用(TDM)与统计时分复用(STDM)
 - 2.5.4 波分复用(WDM)
 - 2.5.5 码分复用(CDMA)
 - 2.5.6 空分复用
 - 2.6 其他

3. 数据链路层

3.1 数据链路层的设计问题

- 3.1.1 数据链路层在协议栈中的位置
- 3.1.2 数据链路层的功能
- 3.1.3 数据链路层提供的服务
- 3.1.4 成帧 framing
- 3.1.5 差错控制
- 3.1.6 流量控制

3.2 差错检测和纠正

- 3.2.1 检错码 (error-detecting code)
- 3.2.2 纠错码 (error-correcting code)
- 3.2.3 海明距离 (Hamming distance)
- 3.2.4 典型检错码
 - 3.2.4.1 奇偶校验
 - 3.2.4.2 校验和
 - 3.2.4.3 循环冗余校验 CRC
- 3.2.5 海明码 hamming code
- 3.2.6 卷积码 Convolutional Code
- 3.2.7 Trellis diagram

3.3 基本的数据链路层协议

- 3.3.1 关键假设
- 3.3.2 基本的协议定义
- 3.3.3 乌托邦式单工协议
- 3.3.4 无错信道单工停止-等待协议
- 3.3.5 有错信道单工停止-等待协议
- 3.3.6 停等协议的性能问题

3.4 滑动窗口协议

- 3.4.1 滑动窗口协议
- 3.4.2 停止-等待协议
- 3.4.3 回退 N 协议(GBN) (协议 5)
 - 3.4.3.1 协议原理分析
 - 3.4.3.2 协议的实现分析
- 3.4.4 选择重传协议 select and repeat(SR) (协议 6)
 - 3.4.4.1 基本原理
 - 3.4.4.2 协议实现分析

3.5 数据链路协议实例

- 3.5.1 点到点链路层协议 PPP
 - 3.5.1.1 PPP 协议实现的功能
 - 3.5.1.2 PPP 协议未实现的功能
 - 3.5.1.3 PPP 协议的构成
 - 3.5.1.4 PPP 协议的帧格式
- 3.5.2 PPPoE

4. 介质访问子层

4.1 信道分配问题

- 4.1.1 子信道的平均延迟
- 4.1.2 静态分配的特点

4.2 多路访问协议

- 4.2.1 随机访问协议
 - 4.2.1.1 ALOHA 协议
 - 4.2.1.1.1 纯 ALOHA
 - 4.2.1.1.2 分隙 ALOHA(slotted ALOHA)
 - 4.2.1.2 载波侦听多路访问协议(CSMA)
 - 4.2.1.2.1 非持续式 CSMA
 - 4.2.1.2.2 持续式 CSMA
 - 4.2.1.2.3 p-持续式 CSMA
 - 4.2.1.3 CSMA/CD (1-持续)
 - 4.2.1.4 CSMA/CD (续)

4.2.2 受控访问协议

- 4.2.2.1 位图协议（预留协议）
 - 4.2.2.1.1 信道利用率分析

4.2.2.2 令牌

- 4.2.2.3 二进制倒计数协议
 - 4.2.2.3.1 信道效率分析

4.2.3 有限竞争协议

- 4.2.3.1 自适应树搜索协议(Adaptive Tree Walk Protocol)

4.2.4 无线 LAN 协议

4.3 以太网

4.3.1 经典以太网

- 4.3.1.1 经典以太网的物理层

- 4.3.1.2 MAC 子层协议

4.3.2 交换式以太网

4.3.3 快速以太网

4.3.4 千兆以太网

4.3.5 万兆以太网

4.3.6 以太网的未来

4.4 数据链路层交换

4.4.1 数据链路层交换原理

- 4.4.1.1 MAC 地址表的构建-逆向学习源地地址

- 4.4.1.2 网桥对于入境帧的处理过程 (forwarding、filtering、flooding)

- 4.4.1.2.1 Forwarding (转发)

- 4.4.1.2.2 Filtering (过滤)

- 4.4.1.2.3 Flooding (泛洪)

- 4.4.1.3 总结

4.4.2 链路层交换机

- 4.4.2.1 交换方式

- 4.4.2.1.1 从带宽的角度

- 4.4.2.1.2 从转发时机的角度

4.4.3 生成树协议

- 4.4.3.1 物理环路引发的问题

- 4.4.3.2 生成树网桥

- 4.4.3.2.1 生成树的三个选举过程

- 4.4.3.2.2 重构生成树

4.4.4 虚拟局域网

- 4.4.4.1 VLAN 类型

- 4.4.4.1.1 基于端口的 VLAN (最常见)

- 4.4.4.1.2 基于 MAC 地址的 VLAN

- 4.4.4.1.3 基于协议的 VLAN

- 4.4.4.1.4 基于子网的 VLAN

- 4.4.4.1.5 VLAN 优点

- 4.4.4.2 Access 链路类型端口

- 4.4.4.3 Trunk 链路类型端口与 Trunk 链路

4.5 无线局域网

4.5.1 无线局域网概述

4.5.2 无线局域网组网模式

- 4.5.2.1 无线局域网体系结构

- 4.5.2.2 802.11 物理层

4.5.3 802.11 介质访问控制

- 4.5.3.1 CSMA/CA

- 4.5.3.2 RTS-CTS 机制 (可选机制)

4.5.4 802.11 帧结构

4.5.5 无线局域网的构建与管理

4.5.6 Wi-Fi6 核心技术概览

5. 网络层

5.1 网络层服务

- 5.1.1 网络层服务概述
- 5.1.2 网络层的关键功能
 - 5.1.2.1 路由 (控制面)
 - 5.1.2.3 提供给传输层的服务
 - 5.1.3.1 无连接服务
 - 5.1.3.2 连接服务
 - 5.1.4 虚电路和数据报网络的性能比较
 - 5.1.4.1 性能角度
 - 5.1.4.2 效率角度

5.2 Internet 网际协议

- 5.2.1 IPv4 报文格式
 - 5.2.1.1 数据包分片
 - 5.2.1.2 IP 地址
 - 5.2.1.3 最长前缀匹配
 - 5.2.1.4 IPv4 地址如何获取
- 5.2.2 DHCP 动态主机配置协议
 - 5.2.2.1 IP 与 MAC 地址
- 5.2.3 ARP 地址解析协议 (Address Resolution Protocol)
 - 5.2.3.1 IP 包转发
- 5.2.4 NAT
 - 5.2.4.1 NAT 工作机制
 - 5.2.4.2 NAT 优缺点
- 5.2.5 Internet 控制报文协议
 - 5.2.5.1 PING (Packet InterNet Groper)
 - 5.2.5.2 Traceroute 和 ICMP

5.3 路由算法

- 5.3.1 优化原则
- 5.3.2 最短路径算法
- 5.3.3 距离向量路由
- 5.3.4 链路状态路由
- 5.3.5 层次路由
- 5.3.6 广播路由 Broadcasting
- 5.3.7 组播路由 multicasting
- 5.3.8 选播路由 Anycast

5.4 Internet 路由协议

- 5.4.1 OSPF-内部网关路由协议
 - 5.4.1.1 OSPF-链路状态
 - 5.4.1.2 OSPF-区域的概念
 - 5.4.1.3 小结
- 5.4.2 RIP-内部网关路由协议
 - 5.4.2.1 小结
- 5.4.3 BGP-外部网关路由协议
 - 5.4.3.1 BGP 协议的特点
 - 5.4.3.2 BGP 报文
 - 5.4.3.3 BGP 路由策略
- 5.4.4 标签交换和 MPLS
 - 5.4.4.1 工作过程

5.5 路由器工作原理

- 5.5.1 控制层
- 5.5.2 数据层

5.6 拥塞控制算法

- 5.6.1 拥塞控制概述
- 5.6.2 流量感知路由
- 5.6.3 流量调节
- 5.6.4 随机早期检测

5.7 服务质量
5.8 三层交换与 VPN
 5.8.1 三层交换
 5.8.2 VPN 技术
 5.8.2.1 VPN 的原理

5.9 IPv6 技术
 5.9.1 IPv6 头部
 5.9.2 IPv6 扩展头
 5.9.3 邻居发现
 5.9.4 IPv6 地址及配置
 5.9.5 IPv4 到 IPv6 迁移及过渡技术
 5.9.5.1 隧道技术
 5.9.5.2 翻译技术

6. 传输层
 6.1 概述和传输层服务
 6.1.1 传输层的位置
 6.1.2 不同终端上的进程如何通信?
 6.1.3 传输层提供什么服务?

 6.2 套接字编程
 6.2.1 网络应用
 6.2.2 进程如何标识自己
 6.2.3 应用编程接口: socket API
 6.2.4 创建套接字: socket()
 6.2.5 套接字描述符
 6.2.6 使用 UDP 套接字实现回音服务
 6.2.6.1 通信流程
 6.2.7 使用 TCP 套接字实现回音服务
 6.2.7.1 服务器使用多个套接字服务客户
 6.2.7.2 基于 TCP 的套接字通信流程
 6.2.8 套接字标识与端口号
 6.2.8.1 套接字端口号的分配

 6.3 传输层复用和分用

 6.3.1 UDP 套接字

 6.3.2 TCP 套接字

 6.4 无连接传输: UDP

 6.4.1 UDP 报文段结构

 6.4.2 校验和计算

 6.4.3 为什么需要 UDP?

 6.5 面向连接的传输: TCP

 6.5.1 TCP 概述

 6.5.2 TCP 报文段结构

 6.5.3 TCP 可靠数据传输

 6.5.3.1 高度简化的 TCP 协议

 6.5.3.2 TCP 发送方要处理的事件

 6.5.3.3 如何设置超时值

 6.5.3.4 TCP 确认的二义性

 6.5.3.5 小结

 6.5.3.6 TCP 使用 GBN 还是 SR

 6.5.3.7 Crash Recovery

 6.5.4 TCP 流量控制

 6.5.4.1 TCP 如何进行流量控制

 6.5.5 TCP 连接管理

 6.5.5.1 TCP 两次次握手建立连接

 6.5.5.2 How to realize the assumptions?

 6.5.5.3 TCP 起始序号的选择

 6.5.5.4 Forbidden region of seqno

- 6.5.5.5 TCP 三次握手建立连接
- 6.5.5.6 关闭 TCP 连接
- 6.5.5.7 客户/服务器经历的 TCP 状态序列
- 6.5.5.8 SYN 洪泛攻击
- 6.5.5.9 TCP 端口扫描
- 6.6 理解网络拥塞
 - 6.6.1 网络拥塞的后果
 - 6.6.2 拥塞控制的常用方法
- 6.7 TCP 拥塞控制
 - 6.7.1 拥塞检测和速率限制
 - 6.7.2 拥塞窗口的调节策略: AIMD
 - 6.7.3 TCP 慢启动
 - 6.7.4 区分不同的丢包事件
 - 6.7.5 TCP 拥塞控制的实现
 - 6.7.6 TCP 发送端的事件与动作
 - 6.7.7 Wireless Issues
 - 6.7.8 TCP 连接的吞吐量
 - 6.7.9 TCP 的公平性
- 6.8 拥塞控制的发展
 - 6.8.1 TCP CUBIC
 - 6.8.1.1 经典 TCP 拥塞控制的性能问题
 - 6.8.1.2 TCP-BIC
 - 6.8.1.3 TCP CUBIC
 - 6.8.2 Google BBR
 - 6.8.2.1 拥塞与瓶颈链路带宽
 - 6.8.2.2 优化点的近似观测
 - 6.8.2.3 BDP 检测阶段
 - 6.8.3 Data Center TCP
 - 6.8.3.1 数据中心的性能问题
 - 6.8.3.2 需要什么样的传输层协议
 - 6.8.3.3 DCTCP 核心思想
 - 6.8.4 DCCP
- 6.9 传输层协议的发展
 - 6.9.1 MPTCP
 - 6.9.1.1 MPTCP 的优势
 - 6.9.1.2 MPTCP 在网络体系结构中的位置
 - 6.9.2 QUIC
 - 6.9.2.1 TCP 存在的问题
 - 6.9.2.2 QUIC 在网络体系结构中的位置

7. 应用层

- 7.1 应用层概述
 - 7.1.1 应用进程通信方式
 - 7.1.2 服务器进程工作方式
- 7.2 域名系统
 - 7.2.1 历史和概述
 - 7.2.2 域名系统名字空间和层次结构
 - 7.2.2.1 域名系统名字空间和层次结构
 - 7.2.3 域名服务器
 - 7.2.3.1 域名服务器分类
 - 7.2.3.2 权威名字服务器
 - 7.2.3.3 递归解析器/递归服务器
 - 7.2.4 域名解析过程
 - 7.2.5 域名系统查询和响应(选讲)
 - 7.2.5.1 DNSSEC
 - 7.2.6 域名系统高速缓存
 - 7.2.7 域名系统隐私(选讲)

7.3 电子邮件

7.3.1 电子邮件系统体系结构

7.3.2 邮件格式

7.3.3 最终交付协议

 7.3.3.1 POP3 协议

 7.3.3.2 IMAP

7.4 WWW

7.4.1 WWW 体系结构概述

7.4.2 静态 Web (对象)

7.4.3 动态 Web 和 Web 应用

 7.4.3.1 动态 Web 页面

 7.4.3.2 通用网关接口 CGI

 7.4.3.3 脚本语言+数据库技术

 7.4.3.4 Client-side dynamic web page generation

 7.4.3.5 典型的动态 Web 技术: AJAX

7.4.4 HTTP 协议

 7.4.4.1 HTTP 发展现状

 7.4.4.2 思考题

7.4.5 Web 缓存技术与 Web 代理

 7.4.5.1 浏览器缓存

 7.4.5.2 Web 代理服务器缓存

 7.4.5.3 Web 代理服务器缓存性能分析

7.4.6 Web 安全与隐私

 7.4.6.1 Web 访问安全

 7.4.6.2 Cookie

7.5 流式音频和视频

7.5.1 流媒体概述

7.5.2 数字音视频与编码

7.5.3 直播与实时音视频

7.6 内容分发

7.6.1 内容和 Internet 流量

7.6.2 服务器群和 Web 代理

7.6.3 内容分发网络 CDN

7.6.4 P2P 网络实现内容分发

7.7 其它应用层协议

7.7.1 Telnet

7.7.2 FTP

7.7.3 SNMP

8. 网络安全

8.1 网络安全的基本属性

8.2 网络攻击与威胁

8.2.1 窃听

8.2.2 嗅探

8.2.3 ARP 协议回顾

 8.2.3.1 ARP 欺骗 - “毒化”

8.2.4 拒绝服务攻击 (Deny of Service)

 8.2.4.1 网络各层中的拒绝服务攻击

 8.2.4.2 SYN Floods

8.2.5 放大攻击

8.2.6 MAC Flooding

8.3 安全机制与手段

8.3.1 密码学

 8.3.1.1 Substitution Ciphers 替换密码

 8.3.1.2 Transposition Ciphers 换位密码

 8.3.1.3 One-Time Pads

8.3.2 现代加密算法

1. 引言

1.1 初识互联网

网络的基本功能：传递信息

服务用什么区分？

- 功能、延迟、带宽、丢失率
- 端节点数目、服务接口
- 可靠性、实时/非实时等外特性

1.2 网络实例

计算机网络拓扑结构主要取决于它的通信子网

1.2.1 网络分类（按地域规模）

- 个域网 PAN (Personal Area Network) : 能在便携式消费电器与通信设备之间进行短距离通信的网络；覆盖范围一般在 10 米半径以内，如蓝牙耳机等
- 局域网 LAN (Local Area Network) : 局部地区形成的区域网络，如企业网络；分布地区范围有限，可大可小，大到一栋建筑、小到办公室内的组网；电脑 WLAN 接入，打印机共享等等
- 城域网 MAN (Metropolitan Area Network) : 范围覆盖一个城市的网络
- 广域网 WAN (Wide Area Network) : 覆盖很大地理区域，乃至覆盖地区和国家

互联网 (Internet)	互连网 (internet)
相似之处	
网络的网络 (这种类型的一个具体实例)	网络的网络 (泛指这种类型)
不同之处	
特指遵循 TCP/IP 标准、利用路由器将各种计算机网络互连起来而形成的、覆盖全球的、特定的互连网	泛指由多个不同类型计算机网络互连而成的网络
使用 TCP/IP	除 TCP/IP 外，还可以使用其他协议
是一个专用名词	是一个通用名词

Figure 1

ISP: Internet 服务商

IXP 或 IX: 互联网交换点

1.2.2 网络组成

① Note

计算机网络从逻辑功能上可划为通信子网和资源子网

1.2.2.1 网络边缘

端系统：位于互联网边缘与互联网相连的计算机和其他设备

- 端系统由各类主机(host)构成：桌面计算机、移动计算机、服务器、其他智能终端设备

1.2.2.2 网络核心

由互联端系统的分组交换设备和通信链路构成的网状网络（网络+路由器）

1.2.2.2.1 接入网

- 接入网的目标是将主机连接到边缘路由器上
 - 边缘路由器是端系统 Host 去往任何其他远程端系统的路径上的第一台路由器
- 各种异构网络通过边缘路由器接入
- 光纤到户 FTTH: Fiber to the home
 - 数字用户线 DSL: Digital Subscriber Line
 - 同轴电缆: Cable
 - 无线接入
 - 无线局域网 WLAN
 - 广域蜂窝接入网

传输单位：bit

⚠ Warning

存储常用字节 Byte K/M/G 层级为 2^{10} 进制

传输常用比特 BitK /M/G 层级为 10^3 进制

$1B = 8b$ (注意大小写)

1.2.2.2.2 物理媒介

- 是指发射机和接收机之间的具体链路介质
- 引导型介质：信号在固体介质中传播，例如铜、光纤、同轴电缆
- 非引导型介质：信号自由传播，包括无线电（陆地无线电、卫星无线电通道）、无线链路类型（无线局域网（WiFi）、广域（如 3/4/5G 蜂窝）蓝牙、地面微波、卫星）

1.2.2.2.3 网络核心两大功能

➤ 功能1：路由

- 全局操作：确定数据分组从源到目标所使用的路径
- 需要路由协议和路由算法，产生路由表

➤ 功能2：转发

- 本地操作：路由器或交换机将接收到的数据分组转发出去（即移动到该设备的某个输出接口）
- 确定转发出去的接口/链路：根据从“入接口”收到分组头中的目的地址，查找本地路由表，确定“出接口”



Figure 2

1.2.3 分组交换和电路交换

1.2.3.1 分组交换（也称包交换 packet switching）

- 主机将数据分成分组，发送到网络，通信双方以分组为单位、使用存储-转发机制，实现数据交互的通信方式
- 网络将数据分组从一个路由器转发到下一个路由器，通过从源到目标的路径上的链路，逐跳传输抵达目的地
- 每个分组在互联网中独立的选择传输路径
- 支持灵活的统计多路复用 multiplexing
- 到达包的顺序和发送顺序可能不一致
- 附加信息开销大

1.2.3.2 电路交换（circuit switching）

- 通信、传输时延小
- 资源预留，不会被抢占 -> 有序、性能有保障
- 容错性低（无法应对互联网中广泛存在的“突发”（Burst）流量）
- 资源利用不够充分

1.2.3.3 存储转发的报文交换（Message switching）

- 路由器需要接收到完整的整个数据报文后，才能开始向下一跳发送
- 有传输延迟
- 同一报文的不同分组可以经过不同的传输路径通过通信子网

三种交换的比较

- 电路交换需要建立连接并预留资源，难以实现灵活复用
- 报文交换(Message Switching)和分组交换较灵活，抗毁性高，在传送突发数据时可提高网络利用率
- 由于分组长度小于报文长度，分组交换比报文交换的时延小，也具有更好的灵活性

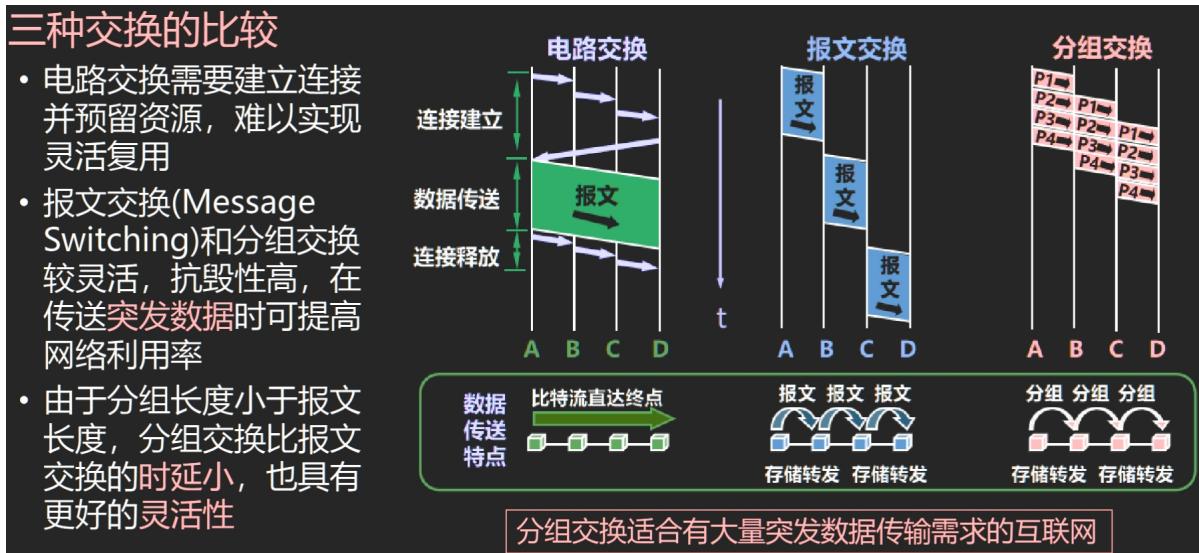


Figure 3

总结：分组交换在实际应用率最高

1.3 协议和分层结构

分层协议 layered protocol

协议由语法、语义、同步三方面组成

协议是指在 **不同结点对等实体** 之间进行通信的规则或约定

1.3.1 协议设计目的

- 可靠性、资源分配、拥塞问题、自适应性、安全问题

1.3.1.1 统一标准

- 明晰简化，便于分析学习
- 各层独立，加速技术演进
- 统一接口，确保技术互通 (interoperable)

1.3.1.2 模块独立

1.3.1.3 分层结构

不同机器上的同一层称为对等层，同一层的实体称为对等实体

➤ 层次栈 (a stack of layers)

- 为降低网络设计的复杂性，网络使用层次结构的协议栈，每一层都使用其下一层所提供的服务，并为上层提供自己的服务

➤ 对等实体 (peers)

- 不同机器上构成相应层次的实体成为对等实体

➤ 接口 (interface)

- 在每一对相邻层次之间的是接口；接口定义了下层向上层提供哪些服务原语

➤ 网络体系结构 (network architecture)

- 层和协议的集合为网络体系结构，一个特定的系统所使用的一组协议，即每层的协议，称为协议栈

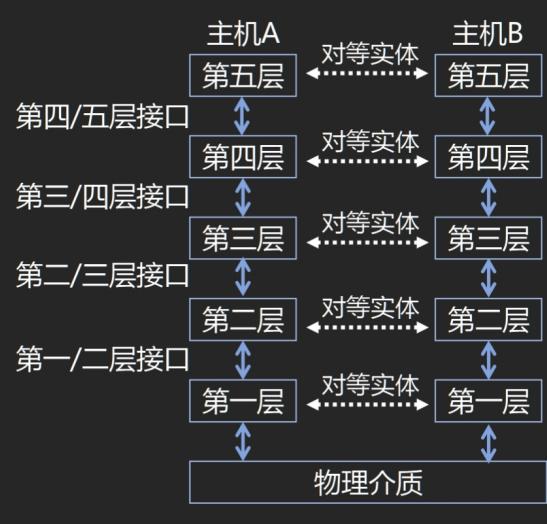


Figure 4

发送端：层层封装；接收端：层层解封装

不同层对应协议数据单元 (PDU Protocol Data Unit)

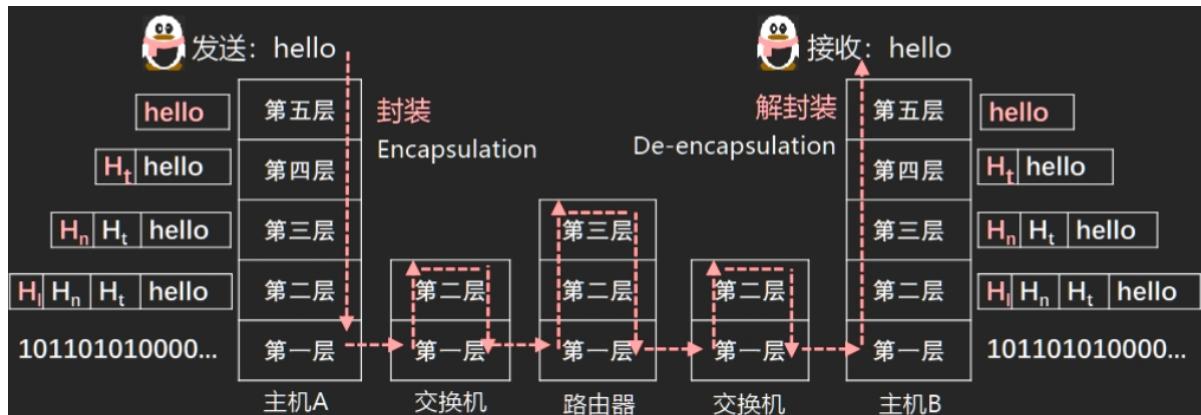


Figure 5

从顶层到底层，每一层 Header 往前加，后面（前一层的 header 等）都是数据

1.3.2 服务原语

服务原语 Service Primitives：请求 request、指示 indication、响应 response、证实 confirmation

- 面向连接 (电话系统) / 无连接 (邮政系统)

➤ 六个核心服务原语 (以面向连接服务为例)



Figure 6

1.3.3 服务与协议的关系

协议是“水平”的，服务是“垂直”的

- > 实体使用协议来实现其定义的服务
- > 上层实体通过接口使用下层实体的服务

1.4 参考模型

1.4.1 OSI 参考模型

OSI: Open Systems Interconnection

7 层模型：从下到上，物理层、数据链路层、网络层、传输层、会话层、表示层、应用层

服务、接口、协议

1.4.1.1 物理层 (Physical Layer)

定义如何在信道上 传输比特 0、1：Bits on the wire

处理信号通过介质的传输

1.4.1.2 数据链路层 (Data Link Layer)

数据链路层的传输单位是帧；点到点通信

实现相邻 (Neighboring) 网络实体间的数据传输，保证数据正确的顺序与完整性

成帧 (Framing)：从物理层的比特流中提取出完整的帧

物理地址 (MAC address)：48 位，理论上唯一网络标识，烧录在网卡，不便更改

流量控制，避免“淹没” (overwhelming)：当快速的发送端遇上慢速的接收端，接收端缓存溢出

共享信道上的访问控制 (MAC)：同一个信道，同时传输信号。

数据链路层协议有：SDLC\HDLC\PPP\STP

1.4.1.3 网络层 (Network Layer)

网络层的传输单位是 数据报

将数据包跨越网络从源设备发送到目的设备 (host to host)

路由 (Routing)：在网络中选取从源端到目的端转发路径，常常会根据网络可达性动态选取最佳路径，也可以使用静态路由

服务质量 (QoS) 控制：处理网络拥塞、负载均衡、准入控制、保障延迟

- 提供面向连接的通信，支持无连接

网络层的控制主要面向运营商

Note

为什么除了 MAC 地址外，还有 IP 地址？不能用 MAC (网卡地址) 寻址，路由器不知道网卡在哪；？？面向运营商 (第一节课)

网络层的协议有：IP\IPX\ICMP\IGMP\ARP\RARP\RIP\OSPF 等

1.4.1.4 传输层 (Transport Layer)

将数据从源端口发送到目的端口 (进程到进程)

传输层为终端用户提供 **端到端的数据传输控制**

- 提供面向连接的通信

两类模式：可靠的传输模式，或不可靠传输模式

- 可靠传输：可靠的端到端数据传输，适合于对通信质量有要求的应用场景，如文件传输等
- 不可靠传输：更快捷、更轻量的端到端数据传输，适合于对通信质量要求不高，对通信响应速度要求高的应用场景，如语音对话、视频会议等

传输层的协议有：TCP\UDP

1.4.1.5 会话层 (Session Layer)

利用传输层提供的服务，在应用程序之间建立和维持会话，并能使会话获得同步

- 设置检验点，使通信双方在通信失效时可从检验点恢复通信

1.4.1.6 表示层 (Presentation Layer)

关注所传递信息的语法和语义，管理数据的表示方法，传输的数据结构

- 数据解密与加密、压缩、格式转换等

1.4.1.7 应用层 (Application Layer)

通过应用层协议，提供应用程序便捷的网络服务调用

提供用户和网络的接口

1.4.2 TCP/IP 参考模型

4 个层！

1.4.2.1 链路层/网络接口层 (Link Layer)

- 类似 OSI 的物理层和数据链路层，描述了为满足无连接的互联网络层需求，链路必须具备的功能

1.4.2.2 互联网层/网际层 (Internet Layer)

- 允许主机将数据包注入网络，让这些数据包独立的传输至目的地，并定义了数据包格式和协议 (IPv4 协议和 IPv6 协议)
- 无连接的通信
- 主要功能有：路由选择

1.4.2.3 传输层 (Transport Layer)

- 允许源主机与目标主机上的对等实体，进行 **端到端的数据传输**：TCP， UDP
 - TCP：可靠、面向连接的协议
 - UDP：无连接，不保证可靠

1.4.2.4 应用层 (Application Layer)

- 传输层之上的所有高层协议：DNS、HTTP、FTP、SMTP

Note

先有 TCP/IP 协议栈，然后有 TCP/IP 参考模型

参考模型只是用来描述协议栈的

ARPNET 最终采用 TCP 和 IP 为主要协议

1.4.3 OSI 模型 TCP/IP 模型对比

7 层模型与 4 层模型

- TCP/IP 模型的网络接口层定义主机与传输线路之间的接口，描述了链路为无连接的互联网层必须提供的基本功能
 - TCP/IP 模型的互联网层、传输层与 OSI 模型的网络层、传输层大致对应
 - TCP/IP 模型的应用层包含了 OSI 模型的表示层与会话层
- 基本设计思想：通用性与实用性
- OSI：先有模型后设计协议，不局限于特定协议，明确了服务、协议、接口等概念，更具通用性
 - TCP/IP 模型：仅仅是对已有协议的描述
- 无连接与面向连接
- OSI 模型网络层能够支持无连接和面向连接通信
 - TCP/IP 模型的网络层仅支持无连接通信（IP）



Figure 7

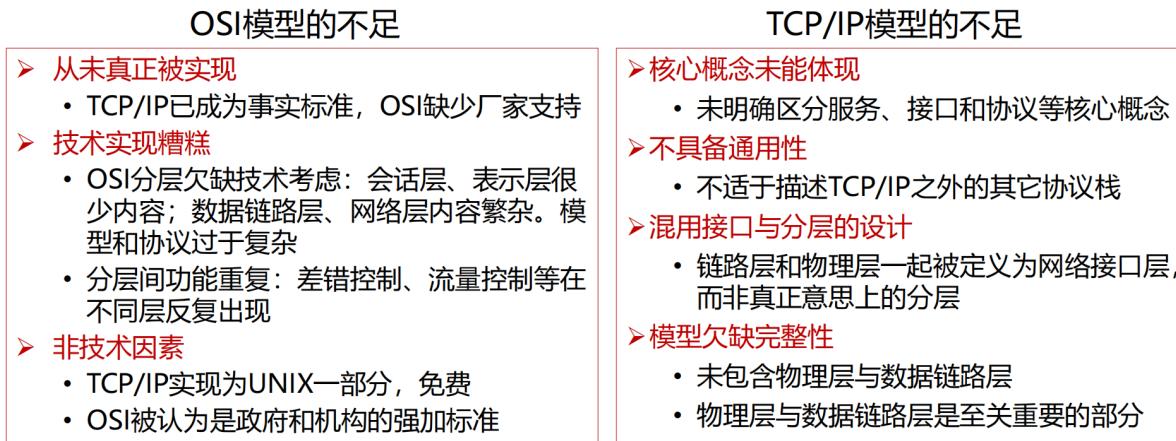


Figure 8

Important

本课程采用 5 层模型：物理层、数据链路层、网络层、传输层、应用层

1.4.4 模型与网络实例

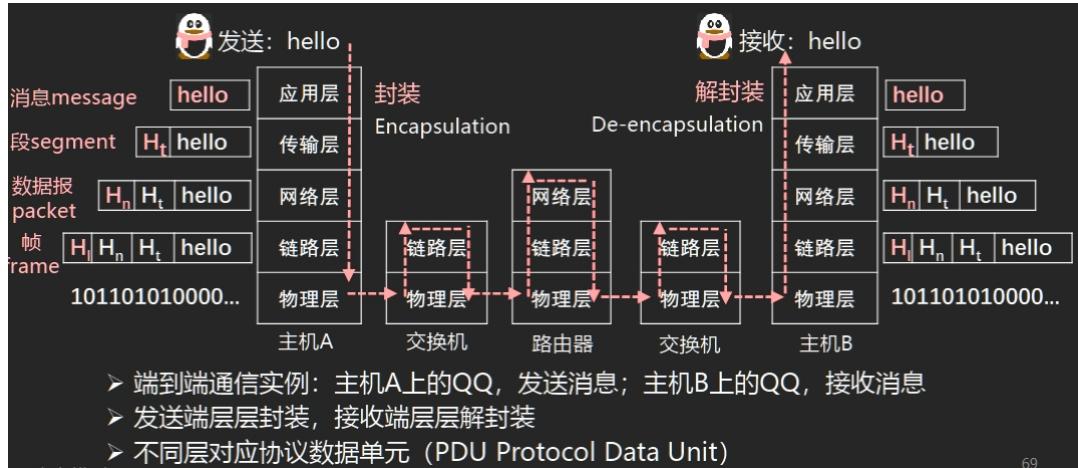


Figure 9

Host 上每个层都有

交换机不解析 IP 地址，只处理 MAC 地址

路由器处理网络地址 (IP 地址)

1.5 计算机网络度量单位

1.5.1 速率/比特率(bit rate)

- 主机在数字信道上传送数据的速率，也称数据率
- 比特率的单位是 b/s(比特每秒)，也可以写为 bps, (bit per second), 或 kbit/s、Mbit/s、Gbit/s 等

1.5.2 波特率(baud rate)

- 码元传输速率，表示单位时间内数字通信系统所传输的码元数/每秒可能发送的信号变化次数（也称调制速率或符号速率 Symbol rate）

- 单位是波特 Baud, 1 波特表示每秒传输 1 码元, 若 1 码元携带 n 比特信息量, 则波特率 M Baud 对应的比特率为 Mn b/s, 表示该码元所需的不同离散值为 $V = 2^n$ 个

1.5.3 带宽(bandwidth)

- 网络中某通道传送数据的能力, 即单位时间内网络中的某信道所能通过的“**最高数据率**”
- 单位是 bit/s, 即 “**比特每秒**”
- 带宽 = 发送数据大小 / 传播时延

1.5.4 包转发率(PPS)

- 全称是 Packet Per Second(包/秒), 表示交换机或路由器等网络设备以包为单位的转发速率
- 线速转发 (line speed) : 交换机端口在满负载的情况下, 对帧进行转发时能够达到该端口线路的最高速度

Note

在交换机上: 大包和小包, 哪个更容易实现线速?

- 小的

1.5.5 时延(Delay)

时延 (delay 或 latency) 是指数据 (一个报文或分组) 从网络 (或链路) 的一端传送到另一端所需的时间, 也称为延迟

- **传输时延/发送时延(transmission delay)**: 数据从结点进入到传输媒体所需要的时间

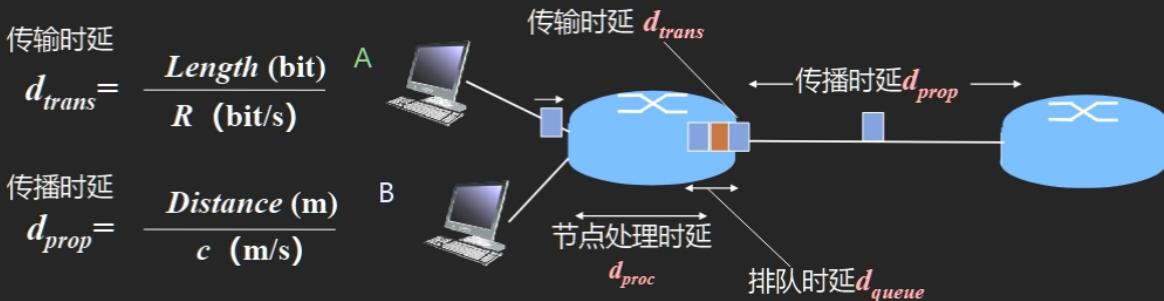
$$\text{发送时延} = \text{分组长度} / \text{发送速率}$$

- **传播时延(propagation delay)**: 电磁波在信道中需要传播一定距离而花费的时间

$$\text{传播时延} = \text{信道长度} / \text{电磁波在信道上的传播速率}$$

- **处理时延(processing delay)**: 主机或路由器在收到分组时, 为处理分组 (例如分析首部、提取数据、差错检验或查找路由) 所花费的时间
- **排队时延(queueing delay)**: 分组在路由器输入输出队列中排队等待处理所经历的时延 -> queue 不长不短最好

➤ 四种时延产生的地方



$$d_{total} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

Figure 10

1.5.6 往返时延 RTT(Round-Trip Time)

- 从发送方发送数据开始，到发送方收到来自接收方的确认，经历的总时间
- 可用于判断网络的通断性、测试网络时延、计算数据包丢失率等

1.5.7 时延带宽积

时延带宽积 = 传播时延 × 带宽，即按比特计数的链路长度

EG: 传播时延为 20ms, 带宽为 10Mb/s, 则: 时延带宽积 = $20/1000 \times 10 \times 10^6 = 2 \times 105 \text{ bit}$

吞吐量 throughput

有效吞吐量 goodput

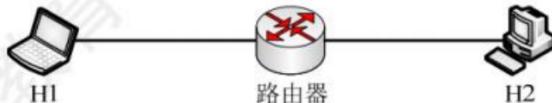
信道利用率 Channel Utilization

丢包率

传输 m 个分组所需时间: $T = (\text{路由器数量} + 1) \times r + (m - 1) \times r$, r = 分组大小/传输速率

EG1:

31. 【2023 统考真题】在下图所示的分组交换网络中，主机 H1 和 H2 通过路由器互连，2 段链路的带宽均为 100Mb/s，时延带宽积（即单向传播时延×带宽）均为 1000b。若 H1 向 H2 发送一个大小为 1MB 的文件，分组长度为 1000B，则从 H1 开始发送的时刻起到 H2 收到文件全部数据时刻止，所需的时间至少是（ ）。(注: $1M = 10^6$ 。)



- A. 80.02ms B. 80.08ms C. 80.09ms D. 80.10ms

Figure 11

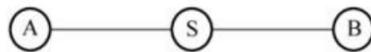
31. D

文件大小为 1MB，分组长度为 1000B，分组数量为 $1MB/1000B = 1000$ ，一个分组从 H1 到 H2 所需的时间 = H1 的发送时延 t_1 + H1 到路由器的传播时延 t_2 + 路由器的发送时延 t_3 + 路由器到 H2 的传播时延 t_4 ，其中 $t_1 = t_3 = 1000B/100Mb/s = 0.08ms$, $t_2 = t_4 = 1000b/100Mb/s = 0.01ms$ 。因此，一个分组从 H1 到 H2 所需的时间为 $(0.08 + 0.01) \times 2 = 0.18ms$ ，H1 发送前 999 个分组所需的时间为 $999 \times t_1 = 79.92ms$ ，总时间等于发送前 999 个分组的时间加上最后一个分组从 H1 到 H2 的时间，即所需的时间至少为 $79.92 + 0.18 = 80.10ms$ 。

Figure 12

EG2:

01. 如下图所示，主机 A 和 B 都通过 10Mb/s 的链路连接到交换机 S。



每条链路上的传播时延都是 $20\mu s$ 。S 是一个存储转发设备，它在接收完一个分组 $35\mu s$ 后开始转发收到的分组。试计算将 10000 比特从 A 发送到 B 所需的总时间。

1) 作为单个分组。

2) 作为两个 5000 比特的分组一个紧接着另一个发送。

Figure 13

01. 【解答】

1) 每条链路的发送时延是 $10000/(10\text{Mb/s}) = 1000\mu\text{s}$ 。

总传送时间等于 $2 \times 1000 + 2 \times 20 + 35 = 2075\mu\text{s}$ 。

解法二：此题属于分组交换各过程中时间不等长的情况，类似于不等长流水段的情况，为避免出错，建议画出对应的时空图。根据题意可分为 5 个流水段，各流水段的时间分别为 $500\mu\text{s}$ 、 $20\mu\text{s}$ 、 $35\mu\text{s}$ 、 $500\mu\text{s}$ 、 $20\mu\text{s}$ ，共有 2 个分组，注意不同分组的相同流水段不能重叠，画出的时空图如下图所示。本题只有 2 个分组，不用流水线的方法也可求得结果，但当分组数量更多时，采用流水线的方法并画出时空图得出计算规律，才不容易出错。

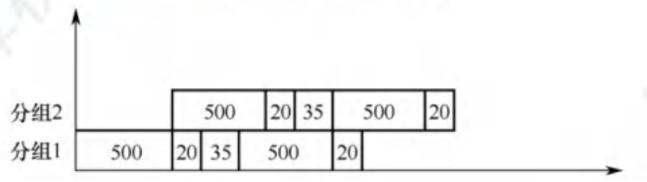


Figure 14

1.6 网络安全威胁

恶意软件、病毒、蠕虫

1.6.0.1 拒绝服务攻击(Denial-of-Service (DoS) attack)

攻击者通过制造大量虚假流量占用资源

使合法流量无法使用资源(服务、带宽)

1.6.0.2 僵尸网络 (Botnet)

采用一种或多种传播手段，将大量主机感染 bot 程序（僵尸程序）病毒，从而在控制者和被感染主机之间所形成的一个可一对多控制的网络

> 分布式 DoS 攻击的发起者，往往通过僵尸网络发起攻击，从而躲避追查

1.6.0.3 如何防御

- > 身份验证：证明你就是你！
- > 保密：加密技术
- > 完整性检查：数字签名检测/防止篡改
- > 访问限制：受密码保护的 VPN
- > 防火墙：接入网络和核心网络中的专用“安全卫士”

1.7 标准化组织

ISO、ITU、IEEE、WFA、IETF、IRTF(Internet research of)

- RFC(request for comments)标准

1.8 互联网发展史与启示

1983 ARPANET 采用 TCP/IP——标志互联网诞生

1.9 总结

主机 A 给主机 B 发送数据的理论最高速率取决于链路带宽、主机 A/B 网卡速率的最小者

2. 物理层

2.1 物理层基本概念

2.1.1 物理层功能

位置：物理层是网络体系结构中的最低层

⚠ Warning

不是连接计算机的具体物理设备，不是负责信号传输的具体物理媒体

功能：如何在连接各计算机的传输媒体上 *传输数据比特流*

作用：尽可能地 *屏蔽掉不同传输媒体和通信手段的差异*

2.1.2 物理层特性

2.1.2.1 物理层机械特性

涉及接口的物理结构，通常采用接线器来实现机械上的连接

定义 接线器的 **形状和尺寸、引线数目和排列、固定和锁定装置** 等

2.1.2.2 物理层电气特性

规定了多条信号线的电气连接及有关电路特性

- 发送器和接收器的电路特性、负载要求、传输速率和连接距离等
- 如发送信号电平、发送器和接收器的输出阻抗、平衡特性等

2.1.2.3 物理层功能特性

描述接口执行的功能，定义接线器的每一引脚(针， Pin)的作用

2.1.2.4 物理层过程特性

指明对于不同功能的各种可能事件的出现顺序

2.1.3 物理层标准及示例

点对点通信线路 用于直接连接两个结点

广播通信线路

2.2 数据通信基础

2.2.1 数据通信基础理论

2.2.1.1 傅里叶分析

任何一个周期为 T 的有理周期性函数 $g(t)$ 可分解为若干项（可能无限多项）正弦和余弦函数之和

$$g(t) = c + \sum_{n=1}^{\infty} a_n \sin(2\pi n f t) + \sum_{n=1}^{\infty} b_n \cos(2\pi n f t)$$

$f = 1/T$ 基本频率

a_n, b_n n 次谐波项的正弦和余弦振幅值

2.2.1.2 有限带宽信号

信号在信道上传输时的特性

- 对不同傅立叶分量的衰减不同，引起输出失真
- 信道有 截止频率 f_c ， $0 \sim f_c$ 的振幅衰减较弱， f_c 以上的振幅衰减厉害，这主要由信道的物理特性决定， $0 \sim f_c$ 是信道的有限带宽
- 实际使用时，可以接入滤波器，限制用户的带宽
- 通过信道的 谐波次数 N 越多，信号越逼真

对于比特率为 B bps 的信道，发送 8 位所需的时间为 $8/B$ 秒，若 8 位为一个周期 T ，则一次谐波的频率是： $f_1 = B/8$ Hz

EG: 能通过信道的最高次谐波数目为： $N = f_c / f_1$

- 音频线路的截止频率为 3000Hz
- $N = f_c / f_1 = 3000/(B/8) = 24000/B$

Bps	T (msec)	First harmonic (Hz)	# Harmonics sent
300	26.67	37.5	80
600	13.33	75	40
1200	6.67	150	20
2400	3.33	300	10
4800	1.67	600	5
9600	0.83	1200	2
19200	0.42	2400	1
38400	0.21	4800	0

Fig. 2-2. Relation between data rate and harmonics.

- 结论：即使对于完善的信道，有限的带宽限制了数据的传输速率

2.2.2 信道的最大数据传输速率

2.2.2.1 奈奎斯特定理 Nyquist

无噪声 有限带宽信道的最大数据传输率公式

- 最大数据传输率 = $2H\log_2 V$ (bps)
- 任意信号通过一个 带宽为 H 的低通滤波器，则 每秒采样 $2H$ 次 (频率 Hz) 就能完整地重现该信号 (无损！)，信号电平分为 V 级

EG: 无噪声理想信道带宽 4MHz，QAM 调制，信道最大数据传输速率为 48Mb/s，则 QAM 调制方案是？

$$2H\log_2 V = 48Mb/s, H = 4MHz, \text{算得 } V = 64 ==> \text{QAM-64}$$

① Note

Symbol rate vs. Data rate

symbol 代表若干个比特，symbol rate 又叫波特率 baud rate

data rate = symbol rate * bits per symbol 比特率 = 波特率 × 每码元所含的比特数

The modulation technique determines the number of bits/symbol

调制技术决定了比特/符号的数量，提高传输速率 => 每个码元携带更多 bit => 多元调制

2.2.2.2 信噪比

随机噪声出现的大小用 信噪比 (信号功率 S 与 噪声功率 N 之比) 来衡量

- 信噪比 $SNR = 10\log_{10} S/N$ ，单位：分贝 db
- 或 信噪比 $SNR = S/N$ 没有单位

2.2.2.3 香农定理 Shannon

带宽为 H 赫兹，信噪比为 S/N 的任意信道的 **最大数据传输率** 为： $H \log_2(1 + S/N)$ (bps)

- 此式是利用信息论得出的，具有普遍意义
- 与信号电平级数、采样速度无关
- 此式仅是上限，难以达到

2.2.2.4 信息量

一条消息包含信息的多少称为信息量

- 信息量的大小与消息所描述事件的出现概率有关
- 一条消息所负载的 **信息量** 等于它所表示的事件发生的 **概率 p** 的倒数的对数

$$I = \log_a \frac{1}{p} = -\log_a p$$

- a 表示进制， $a = 2$ 时， I 的单位为比特； $a =$ 自然数 e ， I 的单位为奈特，通常用 **比特** 作为信息量的单位

2.2.3 数据通信系统模型

信号是消息的载体

2.2.3.1 传输方式

- 串行、并行
- 点到点/点到多点
- 交互方式：
 - 单工/单向 (sender—> receiver, 方向指定)，需要一个信道
 - 半双工 (双向但不能同时)，需要两个信道，每个方向一个
 - 全双工 (双向可同时)，需要两个信道，每个方向一个
- 基带传输、频带传输

2.2.3.2 数据编码技术/调制 digital modulation

1. 不归零制码 (NRZ: Non-Return to Zero)

- 原理：用两种不同的电平分别表示二进制信息“0”和“1”，低电平表示“0”，高电平表示“1”
- 缺点：难以分辨一位的结束和另一位的开始；发送方和接收方必须有时钟同步；若信号中“0”或“1”连续出现，信号直流分量将累加
- 结论：容易产生传播错误

2. 曼彻斯特码 (Manchester)，也称相位编码

- 原理：每一位中间都有一个跳变，从低跳到高表示“0”，从高跳到低表示“1”
- 优点：克服了 NRZ 码的不足。每位中间的跳变即可作为数据，又可作为时钟，能够自同步
- 每个比特需要两个信号周期，信号率（波特率）是数据率的两倍，编码效率是 50%

3. 差分曼彻斯特码 (Differential Manchester)

- 原理：每一位中间都有一个跳变，每位开始时有跳变表示“0”，无跳变表示“1”。位中间跳变表示时钟，位前跳变表示数据
- 优点：时钟、数据分离，便于提取

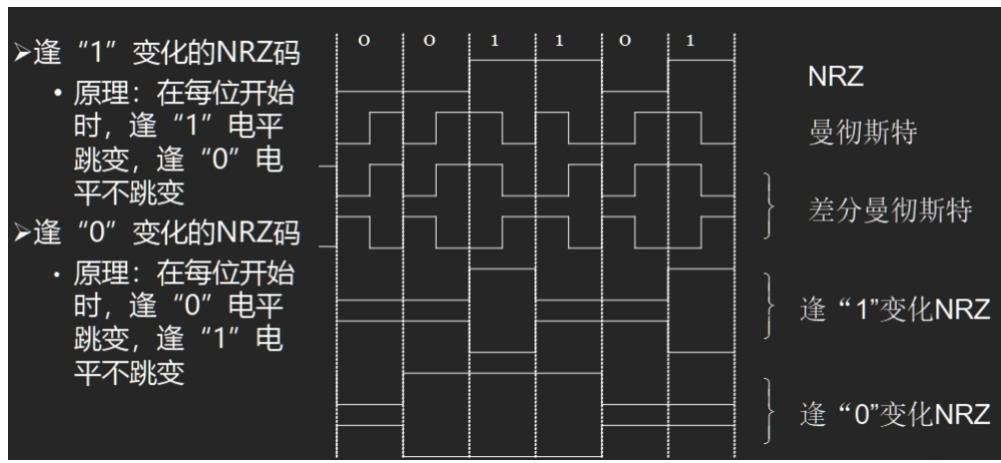


Figure 15

- 曼彻斯特编码和差分曼彻斯特编码在每个码元的中间都发生电平跳变，相当于将一个码元一分为二，编码速率是码元速率的 2 倍，二者所占的频带宽度是原始基带宽度的 2 倍。
- 标准以太网使用的就是曼彻斯特编码，而差分曼彻斯特编码则被广泛用于宽带高速网中

2.2.4 频带传输

三种调制技术：载波 $A \sin(\omega t + \phi)$ 的三个特性 **幅度、频率、相位**

调制是把数字数据变成模拟信号

- 幅移键控法（调幅） Amplitude-shift keying (ASK)，幅移就是把频率、相位作为常量，而把振幅作为变量
- 频移键控法（调频） Frequency-shift keying (FSK)，频移就是把振幅、相位作为常量，而把频率作为变量
- 相移键控法（调相） Phase-shift keying (PSK)，相移就是把振幅、频率作为常量，而把相位作为变量

PSK 又称 BPSK：使用了基准的正弦波和相位反转的波浪，使一方为 0，另一方为 1，从而可以同时传送接受 2 值(1 比特)的信息

QPSK：一种四进制相位调制，采用移相方式 QPSK，每个点 2 比特信息，具有良好的抗噪特性和频带利用率通信业

QAM (Quadrature Amplitude Modulation)：正交幅度调制是一种数字调制方式，产生的方法有正交调幅法和复合相移，16QAM 是指包含 16 种符号的 QAM 调制方式（用 4 个 symbol）

- 在频率相同的前提下，将 AM 与 PM 结合起来，形成叠加信号。
- 设波特率为 B ，采用 m 个相位，每个相位有 n 种振幅，则该 QAM 的数据传输速率 R 为

$$R = B \log_2(mn)$$

(单位为 b / s)

2.3 传输介质

2.3.1 分类

2.3.1.1 导引性介质 guided transmission media

指电磁波被导向沿着某一媒体传播，包括双绞线、同轴电缆、电力线和光纤等

2.3.1.1.1 磁介质 magnetic tape

- 顺序读写
- Bandwidth 带宽 = $800\text{GB/tape} \times 1000 \text{ tapes/box} / (24 \times 60 \times 60\text{s}) = 70+\text{Gbps}$
- Cost = $(800\text{GB/tape} \times 1000\text{tapes/box}) / 5000\$ = 160\text{GB/\$}$

Never underestimate the bandwidth of station wagon full of tapes hurtling down the highway. 带宽很大，但延迟多

2.3.1.1.2 双绞线 twisted pair (网线)

- 绞合密度、扭绞方向和绝缘材料等因素影响双绞线的特性阻抗、衰减和近端串扰
- 便宜、常见
- 局域网多采用双绞线作为传输介质

2.3.1.1.3 同轴电缆 coaxial cable

- 寿命长、容量大、传输稳定、外界干扰小、维护方便等优点
- 同轴电缆带宽高得益于它的高屏蔽性相比于双绞线

2.3.1.1.4 光纤 fiber

- 双层或多层同心圆柱体（类似同轴电缆），由纤芯、包层和护套组成
- 三种实用光纤
 - 单模 纤芯直径很小，折射率分布属于突变型；带宽极大，适用于大容量远距离通信
 - 若光纤的直径减小到光一个波长的大小，则光沿直线传播
 - 多模突变 带宽较窄，适用于小容量短距离通信
 - 多模渐变 纤带宽较宽，适用于中容量中距离通信
 - 多模光纤传输信号的原理：光的全发射特性
- 光纤损耗和波长有关系，三个窗口
- 光纤损耗分为
 - 固有损耗指光纤材料的性质和微观结构引起的吸收损耗和瑞利散射损耗
 - 非固有损耗指杂质吸收、结构不规则引起的散射和弯曲幅射损耗等
- 优点：通信容量大、传输损耗小、抗干扰性好、保密性好、体积小、重量轻

2.3.1.1.5 电力载波

电力载波是电力系统特有的通信方式

- 利用现有电力线，通过载波方式将模拟或数字信号进行高速传输
- 交流电信号+信号 = 调制信号

优点：投资少、连接方便、传输速率高、安全性好和使用范围广

缺点：无法提供高质量的数据传输业务，如家庭电器产生的电磁波干扰等

2.3.1.2 非导引性介质

指电磁波在大气层、外层空间或海洋中进行的无线传播，包括短波传输、地面微波、卫星微波、光波传输等

2.3.1.2.1 短波传输（无线电波）

100m ~ 10m 波长长-> 绕射能力强-> 传播距离远

多径传播：短波电波通过若干条路径或者不同的传播模式由发信点到达收信点的长度不同，而引起由发信点到达收信点的时间不同的现象

多径时散：指不同路径的时延差；与路径长度、工作频率、昼夜、季节等因素有关

- 多径时散对数据通信的影响主要体现在码间干扰
- 为了保证传输质量，往往采用限制数据传输速率的措施

散射传输：利用大气层中传输媒体的不均匀性对无线电波的散射作用进行的超视距通信

2.3.1.2.2 微波

1m ~ 1mm

多路复用、射频工作和中继接力是地面微波传输的三个最基本的工作特点

波长 m * 频率 Hz = 速度 m/s

2.3.1.2.3 光波

$3 \times 10^2 \sim 60 \times 10^4 \mu\text{m}$

分类

- 按照光源特性的不同，分为激光通信和非激光通信
- 按照传输媒体的不同，分为大气激光通信和光纤通信
- 按照传输波段的不同，光波通信分为可见光通信、红外线（光）通信和紫外线（光）通信

大气激光通信可传输语音、数据、图像等信息

- 具有抗干扰性好、设备轻便、保密性强、机动性
- 但使用时收、发天线相互对准较为困难，通信距离限于视距范围
- 易受气候影响，尤其在恶劣气候条件下会造成通信中断

2.3.1.2.4 红外线

不可见光，其波长范围 $760\text{nm} \sim 4.0 \times 10^5 \text{nm}$

2.4 无线与卫星通信

2.4.1 无线

无线传输：可以在自由空间利用电磁波发送和接收信号进行通信

无线电波：指在自由空间（包括空气和真空）传播的射频频段的电磁波

频段名称	频率范围	波段名称	波长范围
特高频 (UHF)	300~3000MHz	微波	1~0.1m (1~10 ⁻¹ m)
超高频 (SHF)	3~30GHz		厘米波 10~1cm (10 ⁻¹ ~10 ⁻² m)
极高频 (EHF)	30~300GHz		毫米波 10~1mm (10 ⁻² ~10 ⁻³ m)
至高频 (THF)	300~3000GHz		亚毫米波 1~0.1mm (10 ⁻³ ~10 ⁻⁴ m)
		光波	3×10^{-3} ~ 3×10^{-5} mm (3×10^{-6} ~ 3×10^{-8} m)

Figure 16

2.4.2 卫星通信

利用人造地球卫星作为中继站，转发或反射无线电波，在两个或多个地球站之间进行的通信特点

- 传播时延长，从一个地球站经卫星到另一个地球站的电波传播时间约需 240 ~ 280ms (可取 270ms)
- 传播损耗大，达 200dB 左右
- 受大气层的影响大
- “面覆盖”式的传播信道

卫星通信的缺点

- 传输时延大：500 毫秒~800 毫秒的时延
- 高纬度地区难以实现卫星通信
- 为了避免各卫星通信系统之间的相互干扰，同步轨道的星位是有一点限度的，不能无限制地增加卫星数量
- 太空中的日凌现象和星食现象会中断和影响卫星通信

移动卫星通信

静止轨道~

中轨道~

低轨道~(LEO)

- 信号传输时延小
- 地面终端设备简单、造价低，是手持式终端的最佳方式
- 卫星造价低、发射容易

2.5 多路复用技术

复用 (multiplexing) 技术的目的是：允许用户使用一个共享信道进行通信，避免相互干扰，降低成本，提高利用率。

2.5.1 频分复用(FDM)

频分复用 (Frequency-division multiplexing) , 是一种将多路基带信号调制到不同频率载波上, 再进行叠加形成一个复合信号的多路复用技术

频分复用的所有用户在同样的时间 占用不同的带宽资源 (请注意, 这里的“带宽”是 **频率带宽** 而不是数据的 **发送速率**)

2.5.2 正交频分复用(OFDM)

正交频分复用 (Orthogonal Frequency Division Multiplexing, OFDM)

- 将信道分成若干正交子信道
- 将高速数据信号转换成并行的低速子数据流, 调制到在每个子信道上进行传输

问题:

Wideband (e.g., 10MHz) ==> short symbol time (e.g., 0.1μs)

- Delay spread caused by multipath transmission: about 1μs ==> 0.1μs << 1μs ==> severe inter-symbol interference (ISI)

解决方法:

Divide the wideband into many narrow bands, each with a carrier (i.e., subcarrier)宽带变窄带

For example: 10MHz --> 100 subcarriers --> symbol time 10μs --> 10μs >> delay spread (1μs) --> smaller ISI

- Multi-carrier modulation

2.5.3 时分复用(TDM)与统计时分复用(STDM)

时分复用 (Time Division Multiplexing) 是将时间划分为一段段等长的时分复用帧

- 每一个时分复用的用户在每一个 TDM 帧中占用固定序号的时隙
- 每一个用户所占用的时隙是周期性地出现 (其周期就是 TDM 帧的长度) 的。
- TDM 信号也称为等时 (isochronous) 信号。
- 时分复用的所有用户在 **不同的时间** 占用 **同样的频带宽度**
- **存在的不足:** 使用时分复用系统传送计算机数据时, 由于计算机数据的突发性质, 用户对分配到的子信道的利用率一般是不高的

统计时分复用 (statistical time division multiplexing) 是指动态地按需分配共用信道的时隙, 只将需要传送数据的终端接入共用信道, 以提高信道利用率的多路复用技术

2.5.4 波分复用(WDM)

波分复用 (Wavelength Division Multiplexing, WDM) 是利用 多个激光器在 **单条光纤** 上同时发送 **多束不同波长激光** 的技术, 频分复用的一种形式

2.5.5 码分复用(CDMA)

又称 **码分多址** (Code Division Multiple Access, CDMA)是指利用码序列相关性实现的多址通信, 基本思想是靠 **不同的地址码** 来区分的地址

- 各用户使用经过特殊挑选的不同码型, 因此彼此不会造成干扰
- 这种系统发送的信号有很强的抗干扰能力, 其频谱类似于白噪声, 不易被敌人发现

码片序列 (chip sequence)

- 每一个比特时间划分为 m 个短的间隔，称为码片 (chip)。
- 每个站被指派一个唯一的 m bit 码片序列。
 1. 如发送比特 1，则发送自己的 m bit 码片序列。
 2. 如发送比特 0，则发送该码片序列的二进制反码。
- 例如，S 站的 8 bit 码片序列是 00011011。
 1. 发送比特 1 时，就发送序列 00011011，
 2. 发送比特 0 时，就发送序列 11100100。

Figure 17

码片序列的正交关系

正交就是向量 S 和 T 的规格化内积 (inner product) 等于 0

➤ 共有四个站进行码分多址CDMA通信。四个站的码片分别为

- A: (-1 -1 -1 +1 +1 -1 +1 +1) B: (-1 -1 +1 -1 +1 +1 +1 -1)
- C: (-1 +1 -1 +1 +1 +1 -1 -1) D: (-1 +1 -1 -1 -1 -1 +1 -1)

➤ 问题

- 现收到这样的码片序列： $M = (-1 +1 -3 +1 -1 -3 +1 +1)$
- 问哪个站发送数据了？
- 发送数据的站发送的1还是0？

$$\mathbf{S} \bullet \mathbf{T} \equiv \frac{1}{m} \sum_{i=1}^m S_i T_i = 0$$

➤ 求解

- $A^*M = 1/8 * (1 -1 +3 +1 -1 +3 +1 +1) = 1$ 因此A发送了1
- 同理， $B^*M = -1$, $C^*M = 0$, $D^*M = 1$
- 即A、D发送了1，B发送了0，C未发数据

Figure 18

码片序列实现了扩频(spread spectrum)

假定 S 站要发送信息的数据率为 b bit/s。由于每一个比特要转换成 m 个比特的码片，因此 S 站实际上发送的数据率提高到 mb bit/s，同时 S 站所占用的频带宽度也提高到原来数值的 m 倍

2.5.6 空分复用

2.6 其他

两个网段在物理层进行互连时要求：数据传输速率必须相同

中继器可以用来连接不同介质的局域网（速率、协议要相同）

中继器原理：信号再生

集线器 Hub 是多个端口的中继器；半双工，所有端口属于一个冲突域

3. 数据链路层

Important

- (一) 数据链路层的功能
- (二) 组帧

(三) 差错控制

- 检错编码；纠错编码

(四) 流量控制与可靠传输机制

- 流量控制、可靠传输与滑动窗口机制；停止 - 等待协议；
- 后退 N 帧协议 (GBN)；选择重传协议 (SR)

(五) 介质访问控制

1. 划分：频分复用、时分复用、波分复用、码分复用
2. 随机访问：ALOHA 协议；CSMA 协议；CSMA / CD 协议；CSMA / CA 协议
3. 轮询访问：令牌传递协议

(六) 局域网

- 局域网的基本概念与体系结构；
- 以太网与 IEEE 802.3; IEEE 802.11 无线局域网；
- VLAN 的基本概念与基本原理

(七) 广域网

- 广域网的基本概念；PPP 协议

(八) 数据链路层设备

- 以太网交换机及其工作原理

3.1 数据链路层的设计问题

3.1.1 数据链路层在协议栈中的位置

向下：利用物理层提供的位流服务

向上：向网络层提供明确的 (well-defined) 服务接口

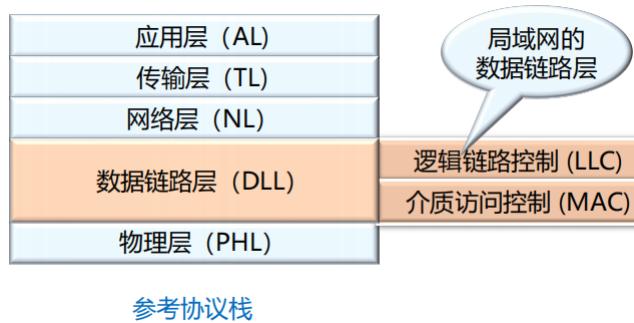


Figure 19

3.1.2 数据链路层的功能

成帧 (Framing)

- 将比特流划分成“帧”的主要目的是为了检测和纠正物理层在比特传输中可能出现的错误，数据链路层功能需借助“帧”的各个域来实现

差错控制 (Error Control)

- 处理传输中出现的差错，如位错误、丢失等

流量控制 (Flow Control) (重点前两个)

- 确保发送方的发送速率，不大于接收方的处理速率
- 避免接收缓冲区溢出

3.1.3 数据链路层提供的服务

无确认无连接 (Unacknowledged connectionless)

- 接收方不对收到的帧进行确认
- 适用场景：误码率低的可靠信道；实时通信（延迟低）
- 网络实例：以太网

有确认无连接 (Acknowledged connectionless)

- 每一帧都得到单独的确认
- 适用场景：不可靠的信道（无线通信）
- 网络实例：802.11

有确认有连接 (Acknowledged connection-oriented)

- 适用场景：长延迟的不可靠信道，例如卫星通信

Note

有连接就一定要有确认，即不存在无确认有连接的服务

3.1.4 成帧 framing

在数据前后都添加首部和尾部，构成帧，帧是数据链路层的传输单元

➤ 分组 (packet) 与 帧(frame) 的关系

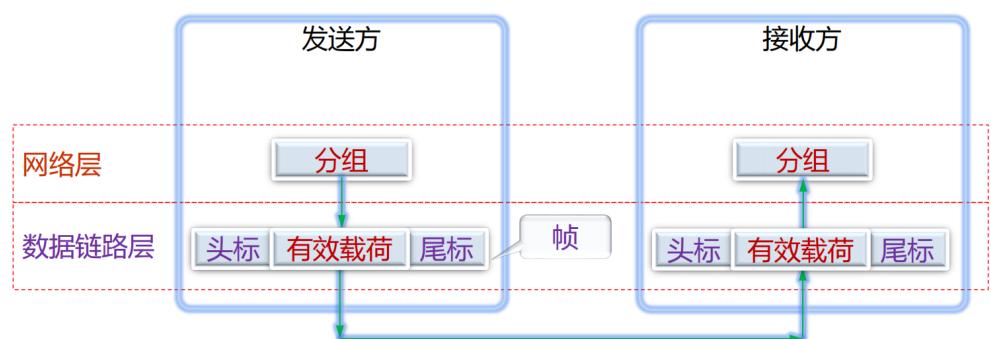


Figure 20

界定符 标识一个帧的开始

- **字节计数法** (Byte count) : 无差错传输的情形，现实中很少用（多米诺骨牌式错误

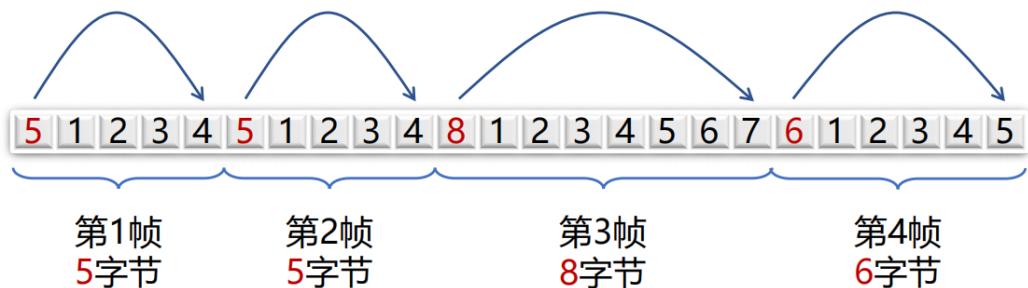
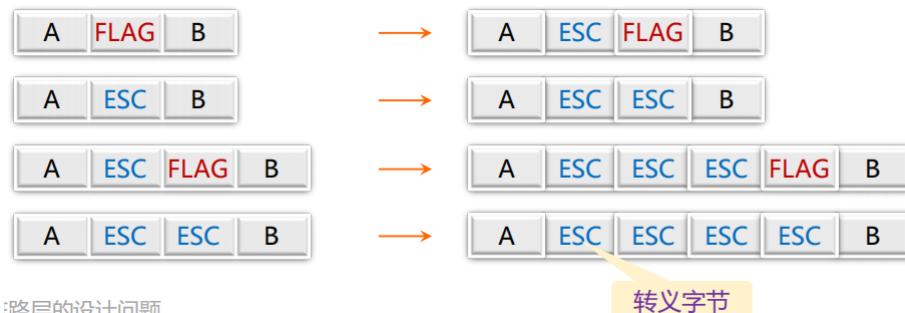


Figure 21

- 带字节填充的定界符法 (Flag bytes with byte stuffing)

- 定界符：一个特殊的字节，比如 01111110，即 0x7E，用于区分前后两个不同的帧
- 发送方将数据（有效载荷）中加入 转义字符 加以区分



- 接收方逐个检查收到的每一个字节
 - 收到 ESC，则后一字节无条件成为有效载荷，不予检查
 - 收到 FLAG，则为帧的边界

不足：overhead 开销大、效率低

- 带比特填充的定界符法 (Flag bits with bit stuffing)

- 定界符：两个 0 比特之间，连续 6 个 1 比特，即 01111110，0x7E
- 发送方在数据（有效载荷）中出现连续 5 个 1 比特，则直接插入 1 个 0 比特 --> 数据中不会出现定界符
- 接收方若出现连续 5 个 1 比特
 - 若下一比特为 0，则为有效载荷，直接丢弃 0 比特
 - 若下一比特为 1，则连同后一比特的 0，构成定界符，一帧结束

- 物理层编码违例 (Physical layer coding violations)

- 核心思想：选择的定界符不会在数据部分出现

3.1.5 差错控制

链路层存在的一个问题：信道的噪声导致数据传输问题

- 差错 (incorrect) : 数据发生错误
- 丢失 (lost) : 接收方未收到
- 乱序 (out of order) : 先发后到，后发先到
- 重复 (repeatedly delivery) : 一次发送，多次接收

💡 解决方案：差错检测与纠正、确认重传

- 确认：接收方校验数据（差错校验），并给发送方应答，防止差错

- 定时器：发送方启动定时器，防止丢失
- 顺序号：接收方检查序号，防止乱序递交、重复递交

3.1.6 流量控制

链路层存在的另一个问题：接收方的处理速率

- 接收方的接收缓冲区溢出

解决方案

- 基于反馈 (feedback-based) 的流量控制
- 接收方反馈，发送方调整发送速率
- 基于速率 (rate-based) 的流量控制
- 发送方根据内建机制，自行限速

3.2 差错检测和纠正

通常采用增加冗余信息（或称校验信息）的策略

- 示例：每个比特传三份，如果每比特的三份中有一位出错，可以纠正

3.2.1 检错码 (error-detecting code)

在被发送的数据块中，包含一些冗余信息，但这些信息只能使接收方推断是否发生错误，但不能推断哪位发生错误，接收方可以请求发送方重传数据

- 主要用在 **高可靠、误码率较低** 的信道上，例如光纤链路
- 偶尔发生的差错，可以通过重传解决差错问题

3.2.2 纠错码 (error-correcting code)

发送方在每个数据块中加入足够的冗余信息，使得接收方能够判断接收到的数据是否有错，并能纠正错误（定位出错的位置）

- 主要用于 **错误发生比较频繁** 的信道上，如无线链路
- 也经常用于物理层，以及更高层（例如，实时流媒体应用和内容分发）
- 使用纠错码的技术通常称为 **前向纠错** (FEC, Forward Error Correction)

码字 (code word)：一个包含 m 个数据位和 r 个校验位的 n 位单元，描述为 (n, m) 码， $n = m+r$

码率 (code rate)：码字中不含冗余部分所占的比例，可以用 m/n 表示

3.2.3 海明距离 (Hamming distance)

给定 m 的条件下，纠正单个错误所需校验位数的下限： $m + r + 1 \leq 2^r$

- m 为消息位个数， r 为校验位个数

两个码字的海明距离：两个码字之间不同对应比特的数目

- 例：0000000000 与 0000011111 的海明距离为 5
- 如果两个码字的海明距离为 d ，则需要 d 个单比特错就可以把一个码字转换成另一个码字

一种编码方案 (code) 的海明距离

- 该编码方案中 **任意** 两个 **合法** 码字的 **最小** 海明距离

- **Ex1:** the Hamming distance of 2 codewords (10001001 and 10110001) is 3
- **Ex2:** consider a coding scheme:
 - 00 => 00000 00000
 - 01 => 00000 11111
 - 10 => 11111 00000
 - 11 => 11111 11111
 - Its Hamming distance is ? 5

Figure 22

The error-detection and error-correcting properties of a code depend on its Hamming distance. 代码的错误检测和纠错特性取决于其海明距离

To detect d errors:

- you need $d+1$ Hamming distance code. (Because with such a code there is no way that d single-bit errors can change a valid code into another valid codeword).
- **Ex2:** can detect 4 bit errors.

Figure 23

To use Hamming distance to correct d errors: you need $2d+1$ Hamming distance code. 用海明距离纠正 d 个错误需要 $2d+1$ 个距离码

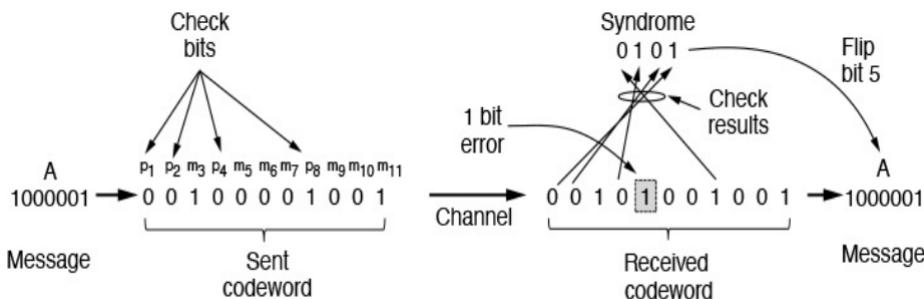


Figure 3-6

Figure 24

图 3-6 纠正 1 位错误的海明码：海明距离为 3，可以纠正 1 个错误或者检测两个错误

3.2.4 典型检错码

常用的检错码包括：

- 奇偶检验 (Parity Check): 1 位奇偶校验是最简单、最基础的检错码
- 校验和 (Checksum): 主要用于 TCP/IP 体系中的网络层和传输层
- 循环冗余校验 (Cyclic Redundancy Check, CRC): 数据链路层广泛使用的校验方法, crc 只能检错不能纠错

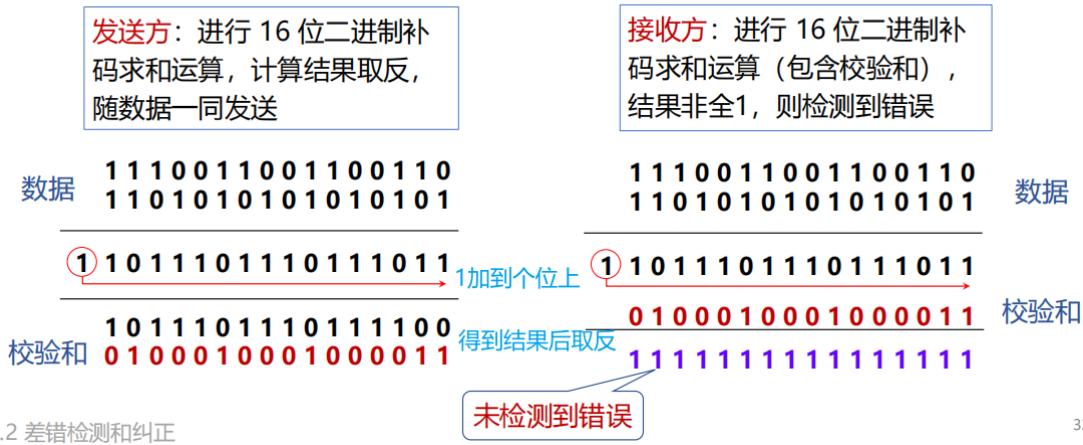
3.2.4.1 奇偶校验

1 位奇偶校验：增加 1 位校验位，**可以检查奇数位错误** --> 1 个 bit 的错误

- $n-1$ 个信息位 + 1 位校验位
- 偶校验：保证 n 个码字中 1 的个数为偶数个
- 奇校验：保证 n 个码字中 1 的个数为奇数个

3.2.4.2 校验和

➤ TCP/IP体系中主要采用的校验方法



33

Figure 25

3.2.4.3 循环冗余校验 CRC

cyclic redundancy code

CRC校验码计算方法

- 设原始数据D为k位二进制位模式
- 如果要产生n位CRC校验码, 事先选定一个n+1位二进制位模式G (称为生成多项式, 收发双方提前商定), G的最高位为1
- 将原始数据D乘以 2^n (相当于在D后面添加 n 个0), 产生k+n位二进制位模式, 用G对该位模式做模2除, 得到余数R (n位, 不足n位前面用0补齐) 即为CRC校验码

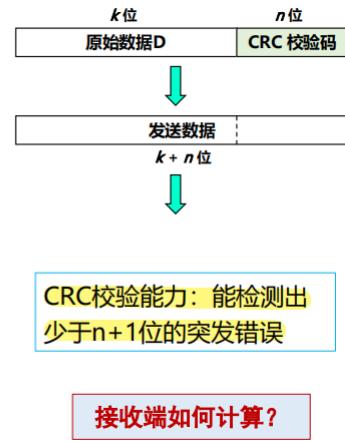


Figure 26

CRC校验码计算示例

- $D = 1010001101$
- $n = 5$
- $G = 110101$ 或 $G = x^5 + x^4 + x^2 + 1$
- $R = 01110$
- 实际传输数据: **101000110101110**

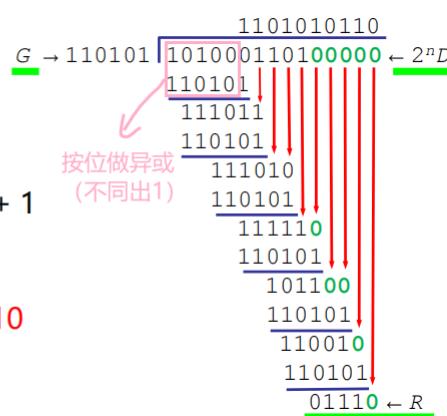


Figure 27

余数R, 作为FCS

CRC有纠错功能, 但是数据链路层只用了它的检错功能

① Note

➤ 设计纠错码

- 要求: m 个信息位, r 个校验位, 纠正单比特错
- 对 2^m 个有效信息中任何一个, 有 n 个与其距离为1的无效码字:
 - 可以考虑将该信息对应的合法码字的 n 位逐个取反, 得到 n 个距离为1的非法码字, 需要 $n+1$ 个位模式来标识
- 因此有: $(n + 1) \cdot 2^m \leq 2^n$
- 利用 $n = m + r$, 得到 $(m + r + 1) \leq 2^r$
- 在给定 m 的情况下, 利用该式可以得出纠正单比特错误校验位数的下界

Figure 28

3.2.5 海明码 hamming code

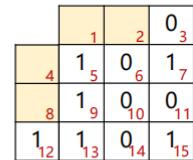
有效信息 n 位, 检验位 k 位, 满足: $n + k \leq 2^k - 1$

纠错1位

➤ 目标: 以奇偶校验为基础, 如何找到出错位置, 提供1位纠错能力

➤ 理解海明码编码过程, 以(15, 11)海明码为例

- 例如: 11比特的数据01011001101
- 11比特数据按顺序放入数据位
- 校验位: 2的幂次方位 (记为 p_1, p_2, p_4, p_8)
- 每个校验位对数据位的子集做校验, 缩小定位错误的范围
- 问题: 每个校验位如何计算?



右下角数字为码字中位序号

Figure 29

子集的选择与校验位计算

- 海明码缺省为偶校验 (也可以使用奇校验) ■ 每组的数据位 ■ 每组的校验位

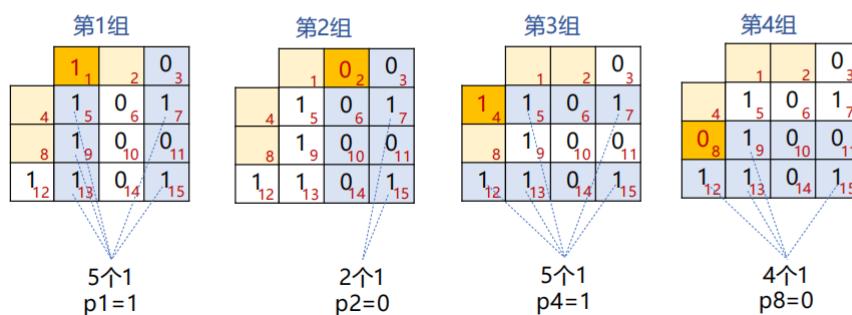
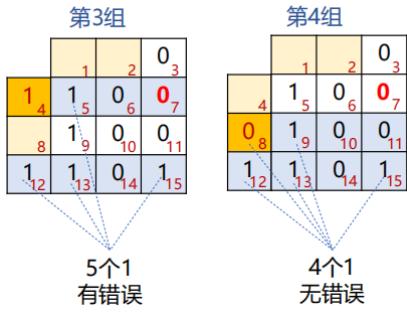


Figure 30

纠正



定位错误与纠正

出错位	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
组1: 0001	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗
组2: 0010	✓	✗	✗	✓	✓	✗	✗	✓	✓	✗	✗	✓	✓	✗	✗
组3: 0100	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗
组4: 1000	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗

- 观察出错的“位置”和出错的“校验组”的关系
- $7: 0111 = 0100 + 0010 + 0001$
- **出错的位置编号正好是检验出错的组的位置编号之和!**

Figure 31

海明码纠正的实现过程

- 每个码字到来前，接收方计数器清零
- 接收方检查每个校验位 k ($k = 1, 2, 4 \dots$) 的奇偶值是否正确（每组运算）
- **若 pk 奇偶值不对，计数器加 k**
- 所有校验位检查完后，若计数器值为 0，则码字有效；若计数器值为 j ，则第 j 位出错。例：若校验位 $p1, p2, p8$ 出错，则第 11 位变反

使用海明码纠正突发错误

- 可采用 k 个码字 ($n = m + r$) 组成 $k \times n$ 矩阵，按列发送，接收方恢复成 $k \times n$ 矩阵
- kr 个校验位， km 个数据位，可纠正最多为 k 个的突发性连续比特错

Figure 32

按列发，每行最多错一位，可以纠正

[最通俗易懂的海明码校验纠错讲解 海明码纠错-CSDN 博客](#)

3.2.6 卷积码 Convolutional Code

3.2.7 Trellis diagram

输入数据: $u = [1 \ 1 \ 0 \ 1 \ 1]$

- 初始状态: 00

编码: $V = [11 \ 01 \ 01 \ 00 \ 01]$

正确接收: $R = [11 \ 01 \ 01 \ 00 \ 01]$

错误接收: $R = [11 \ 01 \ 01 \ 10 \ 00]$

—> 0

---> 1

1/11

1/01

0/01

1/00

0/00

1/10

0/10

1/01

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

1/11

0/11

1/00

0/00

1/10

0/01

3.3.2 基本的协议定义

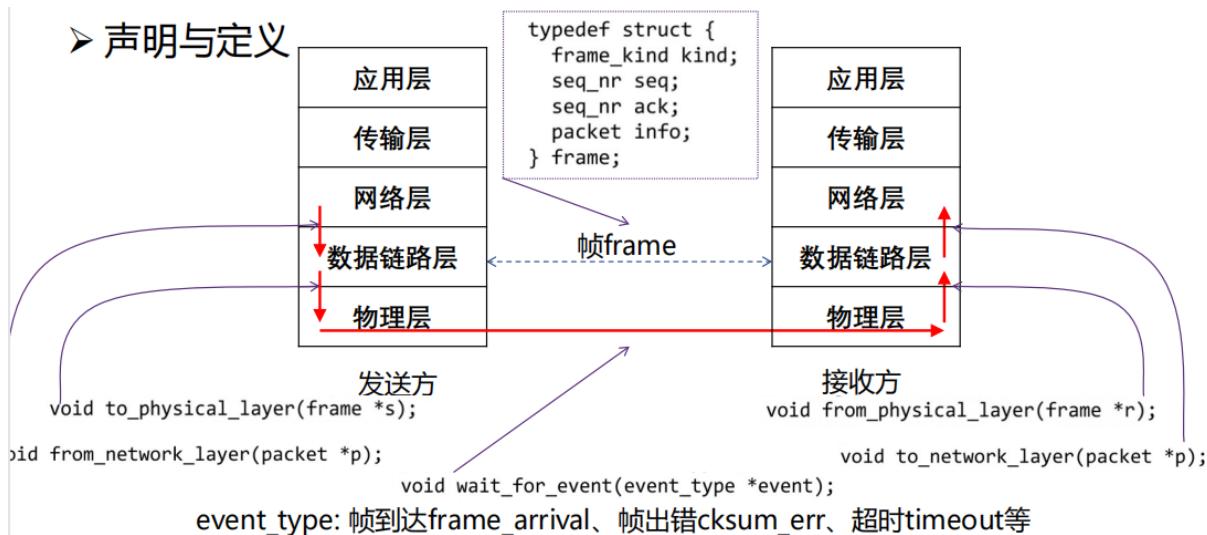


Figure 35

3.3.3 乌托邦式单工协议

假设

- 单工 (Simplex) 协议：数据单向传输
- 完美信道：帧不会丢失或受损
- 始终就绪：发送方/接收方的网络层始终处于就绪状态
- 瞬间完成：发送方/接收方能够生成/处理无穷多的数据

👉 乌托邦：完美但不现实的协议

- 不处理任何流量控制或纠错工作
- 接近于无确认的无连接服务，必须依赖更高层次解决上述问题

3.3.4 无错信道单工停止-等待协议

不再假设

- 接收方能够处理以无限高速进来的数据
- 发送方以高于接收方能处理到达帧的速度发送帧，导致接收方被“淹”

没"(overwhelming)

👉 仍然假设

- 通信信道不会出错 (Error-Free)
- 数据传输保持单向，但是需要双向传输链路 (半双工物理信道)

➤ 停-等式协议 (stop-and-wait)

- 发送方发送一帧后暂停，等待确认 (Acknowledgement) 到达后发送下一帧
- 接收方完成接收后，回复确认接收。
- 确认帧的内容是不重要的：哑帧 (dummy frame)

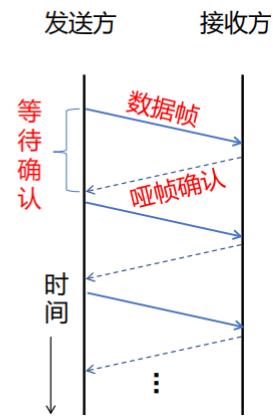


Figure 36

3.3.5 有错信道单工停止-等待协议

➡ 假设

- 通信信道可能会出错，导致：
 - 帧在传输过程中可能会被损坏，接收方能够检测出来
 - 帧在传输过程中可能会丢失，永远不可能到达接收方

➡ 一个简单的解决方案

- 发送方增加一个 计时器(timer)，如果经过一段时间没有收到确认，发送方将超时，于是再次发送该帧

stop and wait

序号 (SEQ: sequence number)

- 接收方需要区分到达的帧是第一次发来的新帧
- 让发送方在发送的帧的头部放一个序号，接收方可以检查它所收到的帧序号，由此判断这是个新帧还是应该被丢弃的重复帧。

序号所需要的最小位数 (bits)

- 在这个协议中，唯一不明确的地方出现在当前帧 (序号 m) 和它的直接后续帧 (序号m+1)
- 1 bit序号(0或1)就足以满足要求。

自动重复请求，或带有重传的肯定确认

- ARQ(Automatic Repeat reQuest)
- PAR(Positive Acknowledgement with Retransmission)

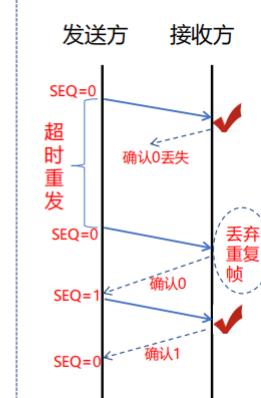
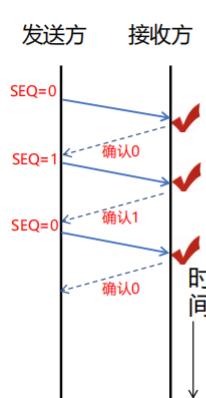


Figure 37

➤ 发送方

- 初始化帧序号0，发送帧
- 等待：正确的确认/错误的确认/超时
- 正确确认：发送下一帧
- 超时/错误确认：重发

```
next_frame_to_send = 0;
from_network_layer(&buffer);
while (true) {
    s.info = buffer;
    to_physical_layer(&s);
    start_timer(s.seq);
    wait_for_event(&event);
    if (event == frame_arrival){
        from_physical_layer(&s);
        if (s.ack == next_frame_to_send){
            from_network_layer(&buffer);
            inc(next_frame_to_send);
        }
    }
}
```

➤ 接收方

- 初始化期待0号帧
- 等待帧达到
- 正确帧：交给网络层，并发送该帧确认
- 错误帧：发送上一个成功接收帧的确认

```
frame_expected = 0;
while (true) {
    wait_for_event(&event);
    if (event == frame_arrival){
        from_physical_layer(&r);
        if (r.seq == frame_expected){
            to_network_layer(&r.info);
            inc(frame_expected);
        }
        s.ack = 1 - frame_expected;
        to_physical_layer(&s);
    }
}
```

Figure 38

效率的评估

F = frame size (bits)

R = channel capacity (Bandwidth in bits/second)

I = propagation delay + processor service time (second)

- 每帧发送时间 (Time to transmit a single frame) = F/R
- 总延迟 (Total Delay) = D = 2I
- 停止等待协议的发送工作时间是 F/C，空闲时间是 D

当 $F < D$ 时：信道利用率 (line utilization)= $\frac{F}{F+R\cdot D} < 50\%$

3.3.6 停等协议的性能问题

👉 停止-等待机制降低了信道利用率

- 设数据速率记为 R，帧长度记为 F，往返延迟记为 D，则采用停-等协议的线路效率为： $F/(F+R\cdot D)$
- 假如将链路看成是一根管道，数据是管道中流动的水，那么在传输延迟较长的信道上，停-等协议无法使数据充满管道，因而信道利用率很低

👉 解决办法

- 流水线协议或管道协议：允许发送方在没收到确认前连续发送多个帧

3.4 滑动窗口协议

用于流量控制

3.4.1 滑动窗口协议

窗口机制

- 发送方和接收方都具有一定容量的缓冲区（即窗口），发送端在收到确认之前可以发送多个帧

目的：

- 对可以连续发出的最多帧数（已发出但未确认的帧）作限制

👉 序号使用

- 循环重复使用有限的帧序号

👉 流量控制：接收窗口驱动发送窗口的转动

- 发送窗口：其大小记作 W_T ，表示在收到对方确认的信息之前，可以连续发出的最多数据帧数
 - 接收窗口：其大小记作 W_R ，为可以连续接收的最多数据帧数
- 👉 累计确认：不必对收到的分组逐个发送确认，而是对按序到达的最后一个分组发送确认

发送窗口与接收窗口(窗口大小7)

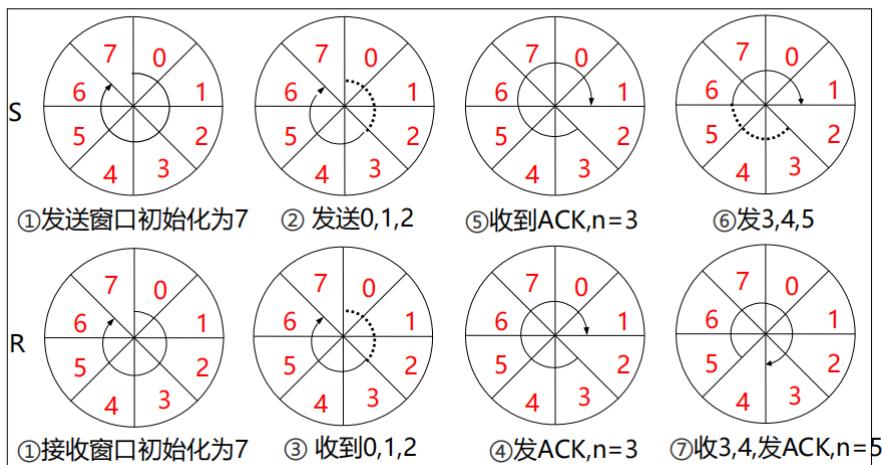


Figure 39

收到，前窗口后移，后窗口不动；发出确认后，后窗口才后移

3.4.2 停止-等待协议

发送方每次只能发一个帧，当发送方收到接收方的确认帧之后，才能发生下一个帧

从滑动窗口角度看，停止-等待协议的发送窗口和接收窗口大小均为1

超时重传

信道利用率U 很低

$$U = \frac{T_D}{T_D + RTT + T_A}$$

发送时延 T_D ，接收时延 T_A ，往返时延 RTT

3.4.3 回退 N 协议(GBN) (协议 5)

3.4.3.1 协议原理分析

出错全部重发

- 当接收端收到一个出错帧或乱序帧时，丢弃所有的后继帧，并且不为这些帧发送确认
- 发送端超时后，重传所有未被确认的帧

👉 适用场景

- 该策略对应接收窗口为 1 的情况，即只能**按顺序接收帧**

👉 优缺点

- 优点：连续发送提高了信道利用率
- 缺点：按序接收，出错后即便有正确帧到达也丢弃重传

➤ 基本原理

- 当发送方发送了N个帧后，若发现该N帧的前一个帧在计时器超时后仍未返回其确认信息，则该帧被判为出错或丢失，此时发送方就重新发送出错帧及其后的N帧。

➤ 滑动窗口长度

- 出错全部重发时，若帧序号为n位，接收窗口 $W_R=1$ ，发送窗口 $W_T \leq 2^n - 1$

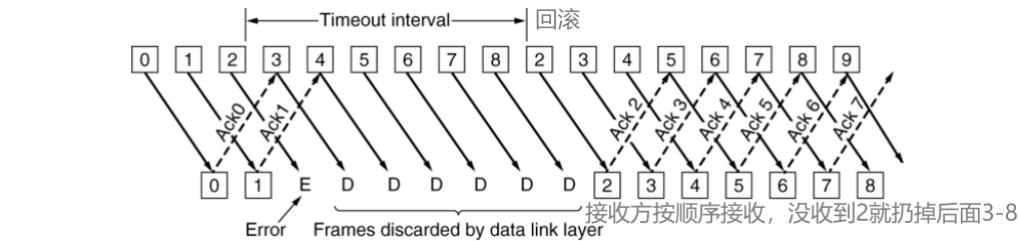


Figure 40

3.4.3.2 协议的实现分析

发送方：

- 窗口尺寸： $1 < W_T \leq 2^n - 1$ ，最多连续发送窗口中的 W_T 个PDU
- 窗口滑动：收到期望的ACK(k)：窗口底部移到PDU(k)，窗口顶部向前移动，始终保持窗口里有 W_T 个PDU未确认。
- 窗口滑动后，发送新进入窗口的PDU
- 超时重发：超过T未收到期望的ACK，重发窗口中的PDU（回退整个窗口）
- 超次数失败：超过最大重发次数 N_{max} 仍无正确应答

Figure 41

3.4.4 选择重传协议 select and repeat(SR) (协议 6)

若发送方发出连续的若干帧后，收到对其中某一帧的否认帧，或某一帧的定时器超时，则 **只重传该出错帧或计时器超时的数据帧**

该策略对应接收窗口大于 1 的情况，即暂存接收窗口中序号在出错帧之后的数据帧

- 优点：避免重传已正确传送的帧
- 缺点：在接收端需要占用一定容量的缓存

3.4.4.1 基本原理

- 在发送过程中，如果一个数据帧计时器 **超时**，就认为该帧丢失或者被破坏；**接收端只把出错的帧丢弃，其后面的数据帧保存在缓存中**，并向发送端回复 NAK；发送端接收到 NAK 时，**只重传出错的帧**
- 如果 **落在窗口内的帧** 从未接受过，那么存储起来，等比它序列号小的所有帧都正确接收后，按次序交付给网络层
- 接收端收到的数据包的顺序可能和发送的数据包顺序不一样，因此在数据包里必须含有 **顺序号** 来帮助接收端进行排序。
- 没有累计确认的机制

滑动窗口长度

- 发送窗口的尺寸： $W^T \leq 2^{n-1}$ ，发送窗口应等于或小于序号空间的一半
- 接收窗口小于等于发送窗口 $W^R \leq 2^{n-1}$

3.4.4.2 协议实现分析

发送方必须响应的三件事

- 上层的调用：检测有没有可以使用的序号，如果有就发送
- 收到 ACK：如果收到的是最小序号的 ACK，窗口滑动。如果收到其他序号的 ACK，进行标记
- 超时事件：每个 PDU 都有定时器，哪个超时重传哪个

发送方：

1. 窗口尺寸： $1 < W_T \leq 2^{n-1}$ ，最多连续发送窗口中的 W_T 个 PDU
2. 窗口滑动：与回退 N 帧协议相同
3. 选择重发：收到 NAK (k)，重发 PDU (k)
4. 超时重发：超过 T 未收到期望的 ACK，重发当前超时未应答的 PDU
5. 超次数失败：超过最大重发次数 N_{max} 仍无正确应答，报告上层失败

Figure 42

接收方：

1. 窗口尺寸： $1 < W_R \leq 2^{n-1}$
2. 窗口滑动：窗口底部数据上交，窗口向前滑动一步
3. 窗口内接收：窗口内的 PDU 全部接收，存储出错的后续 PDU，按序交付；
窗口外的 PDU 一律丢弃
4. 确认策略：按序到达的 PDU 可立即确认，也可延迟确认(收到多帧后一起确认)ACK (k)；出错用否定性确认 NAK(k) (期望重发 k 号 PDU)

Figure 43

3.5 数据链路协议实例

3.5.1 点到点链路层协议 PPP

PPP 协议是目前使用最多的数据链路层协议之一

能够在不同的链路上运行；能够承载不同的网络层分组

特点：简单、灵活

3.5.1.1 PPP 协议实现的功能

3.5.1.2 PPP 协议未实现的功能

帧数据的纠错功能

- 数据链路层的 PPP 协议 **只进行检错**，PPP 协议是不可靠传输协议

流量控制功能

- PPP 协议未实现点到点的流量控制

可靠传输功能

- PPP 为 **不可靠协议**，不使用帧的序号。不可靠网络中可能使用有序号的工作方式。

多点连接功能

- PPP 协议不支持多点线路，**只支持点对点** 的链路通信。

单工和半双工链路

- PPP 协议支持全双工链路

3.5.1.3 PPP 协议的构成

封装 (Encapsulation)

- 提供在同一链路上支持不同的网络层协议
- PPP 既支持异步链路（无奇偶检验的 8 比特数据），也支持面向比特的同步链路
- IP 数据包在 PPP 帧中是其信息部分，其长度受到 MTU 的限制

链路控制协议 LCP (Link Control Protocol)。

- 用来建立、配置和测试数据链路的链路控制协议，通信双方可协商一些选项

网络控制协议 NCP (Network Control Protocol)。

- 其中每个协议支持一种不同的网络层协议，如 IP、OSI 的网络层、DECnet、AppleTalk 等

3.5.1.4 PPP 协议的帧格式

1.字节填充

避免在信息字段中出现和 **标志字段** 一样的比特组合 (FLAG 为 0X7E)

当 PPP 使用 **异步传输** 时，定义 **转义字符** 0X7D，并使用字节填充 (ESC 为 0X7D)

- 将信息字段中出现的 0x7E 字节转变成为 2 字节序列(0x7D, 0x5E)

2.零比特填充

PPP 协议用在 SONET/SDH 链路时，使用 **同步传输**（一连串的比特连续传送），不是异步传输（逐个字符地传送）

异步传输是面向字符的传输，而同步传输是面向比特的传输

3.5.2 PPPoE

Ethernet 优点：原理简单，应用非常广，设备成本低

Ethernet 缺点：安全性较低、不宜管理：使用广播信道，造成了安全性较低，无认证功能

PPP 优点：

- 原理简单
- 安全性高：点对点信道，提供认证机制
- 提供良好的访问控制和计费功能

==> 结合变成 PPPoE(Point-to-Point Protocol over Ethernet)

- 提供在以太网链路上的 PPP 连接
- 实现了传统以太网不能提供的身份验证、加密，以及压缩等功能
- 实现基于用户的访问控制、计费、业务类型分类等，运营商广泛支持
- PPPoE 使用 Client/Server 模型，服务器通常是接入服务器

➤ PPPoE报文格式及Ethernet帧封装

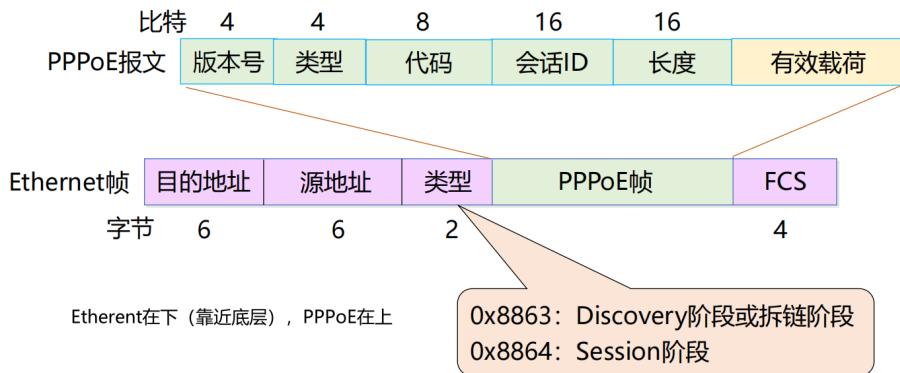


Figure 44

4. 介质访问子层

介质访问子层在数据链路层的下面

4.1 信道分配问题

4.1.1 子信道的平均延迟

信道 N 等分后每个子信道平均延迟时间会增加 (有害, 增加 N 倍)

4.1.2 静态分配的特点

问题：资源分配不合理

4.2 多路访问协议

4.2.1 随机访问协议

4.2.1.1 ALOHA 协议

4.2.1.1.1 纯 ALOHA

原理：想发就发

特点：随时可能冲突

帧时：发送一个标准帧需要的时间

S 吞吐量：在发送时间 T 内发送成功的平均帧数， $0 < S < 1$

生成帧均值： $2G$ (G : 网络负载)

冲突危险期： $2D$ (D : 单向传播延迟)

P_0 成功传输概率：连续两个时间 T 内都没有生成其它帧的概率

$$P_0 = e^{-2G}$$

纯 ALOHA 协议的性能：(吞吐率最大) 18.4%

4.2.1.1.2 分隙 ALOHA(slotted ALOHA)

冲突危险期：D

分隙 ALOHA 协议的性能：（吞吐率最大）36.8%

4.2.1.2 载波侦听多路访问协议(CSMA)

CSMA：Carrier Sense Multiple Access

特点：“先听后发” -> 改进 ALOHA 的侦听/发送策略分类

4.2.1.2.1 非持续式 CSMA

特点

- ① 经侦听，如果介质空闲，开始发送
- ② 如果介质忙，则等待一个随机分布的时间，然后重复步骤 ①
 - 好处：等待一个随机时间可以减少再次碰撞冲突的可能性
 - 缺点：等待时间内介质上如果没有数据传送，这段时间是浪费的

4.2.1.2.2 持续式 CSMA

特点

- ① 经侦听，如介质空闲，则发送
- ② 如介质忙，持续侦听，一旦空闲立即发送
- ③ 如果发生冲突，等待一个随机分布的时间再重复步骤 ①
 - 好处：持续式的延迟时间要少于非持续式
 - 缺点：如果两个以上的站等待发送，一旦介质空闲就一定会发生冲突

4.2.1.2.3 p-持续式 CSMA

特点

- ① 经侦听，如介质空闲，那么以 p 的概率发送，以 $(1-p)$ 的概率 延迟一个时间单元发送
- ② 如介质忙，持续侦听，一旦空闲重复 ①
- ③ 如果发送已推迟一个时间单元，再重复步骤 ①

 **Note**

冲突窗口

4.2.1.3 CSMA/CD （1-持续）

4.2.1.4 CSMA/CD （续）

在发完之前能知道有冲突

4.2.2 受控访问协议

无冲突协议

4.2.2.1 位图协议（预留协议）

竞争期：在自己的时槽内发送竞争比特

- 举手示意
- 资源预留

传输期：按序发送

- 明确的使用权，避免了冲突

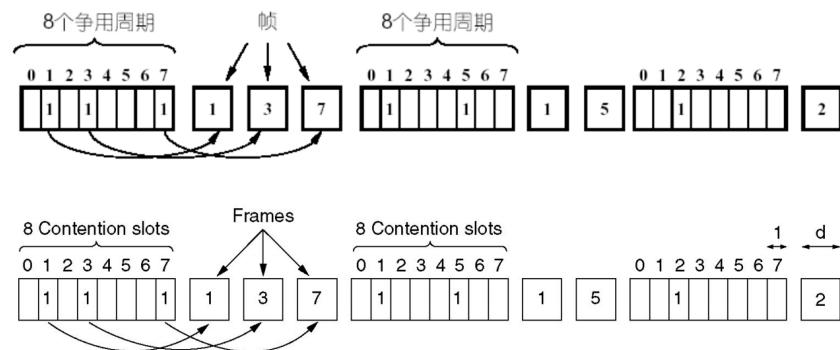


Figure 45

4.2.2.1.1 信道利用率分析

假设

- 有 N 个用户，需 N 个时隙，每帧 d 比特

信道利用率

- 在低负荷条件下： $d/(d+N)$ (N 越大，站点越多，利用率越低)
- 在高负荷条件下： $d/(d+1)$ ，接近 100%

缺点

- 位图协议无法考虑优先级

4.2.2.2 令牌

4.2.2.3 二进制倒计数协议

站点：编序号，序号长度相同

竞争期：有数据发送的站点从高序号到低序号排队，高者得到发送权

特点：**高序号站点优先**

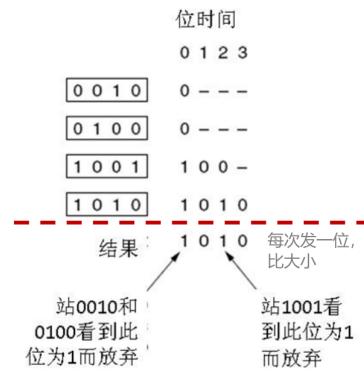


Figure 46

防止低序号站点一直抢不到发送权，可以怎样办？

换编号 overhead

4.2.2.3.1 信道效率分析

N 个站的二进制编码所需位数是 $\log_2 N$ 位

信道的利用率为: $d/(d + \log_2 N)$

如果规定每个帧的帧头为发送地址，即竞争的同时也在发送。则效率为 100%

4.2.3 有限竞争协议

4.2.3.1 自适应树搜索协议(Adaptive Tree Walk Protocol)

- 在一次成功传输后的第一个竞争时隙，**所有站点**同时竞争。
- 如果**只有一个站点**申请，则获得信道。
- 否则在下一竞争时隙，有一半站点参与竞争（递归），下一时隙由另一半站点参与竞争
- 即所有站点构成一棵**完全二叉树**。

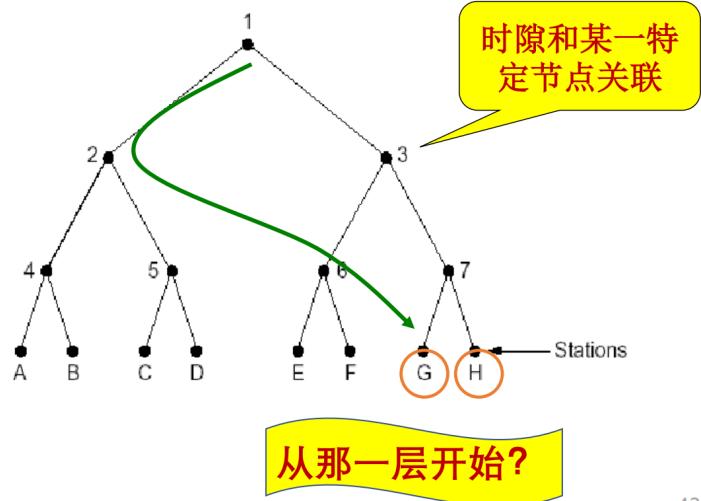


Figure 47

4.2.4 无线 LAN 协议

MACA

4.3 以太网

4.3.1 经典以太网

4.3.1.1 经典以太网的物理层

最高速率 10Mbps



任意两个收发器之间距离不得超过 2.5km (和 CSMA/CD 的 delay 有关)

且任意两个收发器之间经过的中继器不能超过 4 个以保证 MAC 协议正常工作。

4.3.1.2 MAC 子层协议

主机运行 CSMA/CD 协议

常用的以太网 MAC 帧格式有两种标准：

- DIX Ethernet V2 标准 (最常用的)
 - IEEE 的 802.3 标准

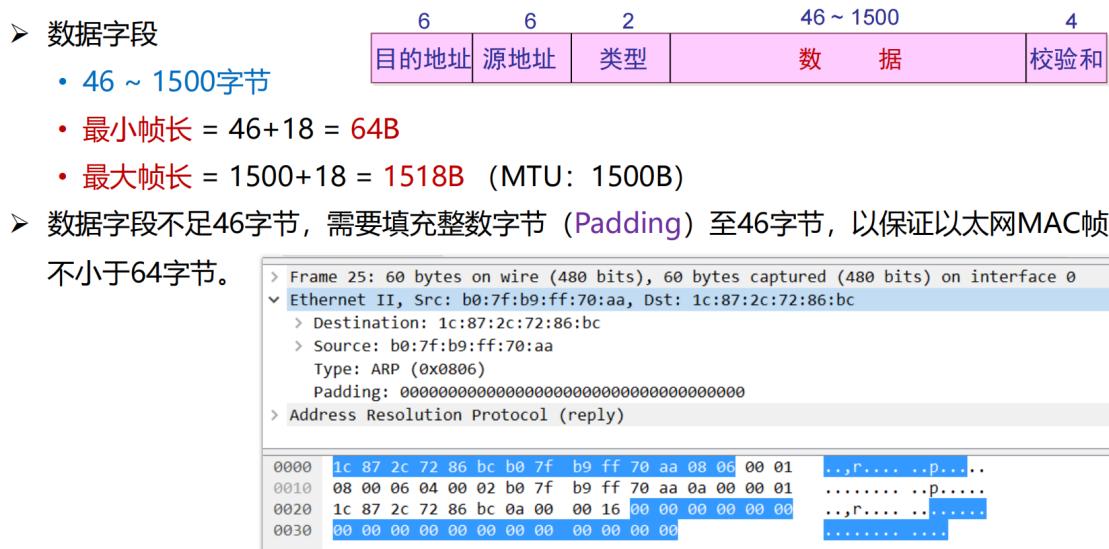


Figure 48

二进制指数后退(Binary exponential backoff)的 CSMA/CD

- 使用CSMA/CD的经典以太网检测到冲突后，会立即中止传输，并发出一个短冲突加强信号，在等待一段**随机时间**后重发。
- **二进制指数后退(Binary exponential backoff)**的CSMA/CD
 - 确定基本退避时间槽，其长度为以太介质上往返传播时间(2τ)，以太网中设为**512比特时间**
 - 定义重传次数 k , $k \leq 10$, 即

$$k = \min[\text{重传次数}, 10]$$
 - 从整数集合 $[0, 1, \dots, (2^k - 1)]$ 中随机地取出一个数，记为 r ；
 - 重传所需的时延就是 r 倍的时间槽 2τ ；
 - 当重传达 16 次仍不能成功时即丢弃该帧，并向高层报告。

Figure 49

- 数字 r 小的优先 (先发送)

$$P = F/B, \text{ F 为帧长, B 为带宽}$$

信道效率 = $\frac{P}{P + 2\tau/A}$, 传送一帧平均需要 P 秒，某个站获得信道的概率为 A , 2τ 为时间槽

信道效率 = $\frac{1}{1 + 2BLe/cF}$, L 为电缆长度, c 为信号传播速度；假设每帧 e 个竞争时间槽

电缆越长, τ 越大, 任何两个站之间的最大电缆距离会影响性能

在给定帧长的情况下, 增加带宽或距离会降低网络效率

? ?

4.3.2 交换式以太网

4.3.3 快速以太网

fast Ethernet(IEEE 802.3u, 1995)

带宽: 10Mbps -> 100Mbps

比特时间: 100ns -> 10ns (电缆的最大长度降低到十分之一)

保留原来的工作方式 (帧格式、接口、过程规则)

Note

名称中 100Base 意味着 100Mbps 的以太网

4.3.4 千兆以太网

gigabit Ethernet(IEEE 802.3ab, 1998)

100Mbps -> 1000Mbps(1Gbps)

保留原来的工作方式 (帧格式、接口、过程规则)

全双工和半双工两种方式工作

- 在半双工方式下使用 CSMA/CD (为了向后兼容)，增加 载波扩充 和 帧突发
- 全双工方式不需要使用 CSMA/CD (缺省方式)

4.3.5 万兆以太网

10-Gigabit Ethernet(IEEE 802.3ae, 2002)

1Gbps -> 10Gbps

只支持全双工，不再使用 CSMA/CD

保持兼容性

重点是超高速的物理层

4.3.6 以太网的未来

优势：灵活性、简单性、兼容性、廉价、可靠、易维护、易扩展

4.4 数据链路层交换

4.4.1 数据链路层交换原理

物理层设备扩充网络 => 扩大了冲突域，性能降低，安全隐患!

数据链路层设备扩充网络

- 网桥或交换机
- 分隔了冲突域

图中有几个冲突域？ 2个

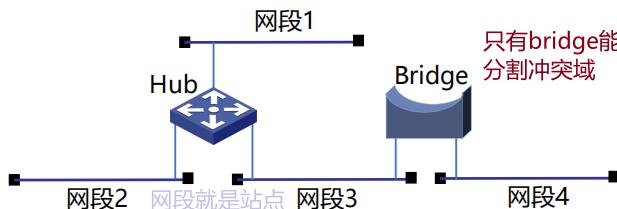


Figure 50

理想的网桥是透明的。

- 即插即用，无需任何配置
- 网络中的站点无需感知网桥的存在与否

4.4.1.1 MAC 地址表的构建-逆向学习源地地址

主机向外发送数据时，其 MAC 地址就会被学习

MAC 地址表的构建

- 增加表项：帧的源地址对应的项不在表中

- 删除表项：**老化时间**到期，默认300s
- 更新表项：帧的源地址在表中，更新时间戳

Note

MAC地址表会满而溢出吗？是不是存在安全隐患？

会，是，因为可以伪造MAC地址，真正有效的站点没办法被转发，被导向到别的站点

4.4.1.2 网桥对于入境帧的处理过程 (forwarding、filtering、flooding)

4.4.1.2.1 Forwarding (转发)

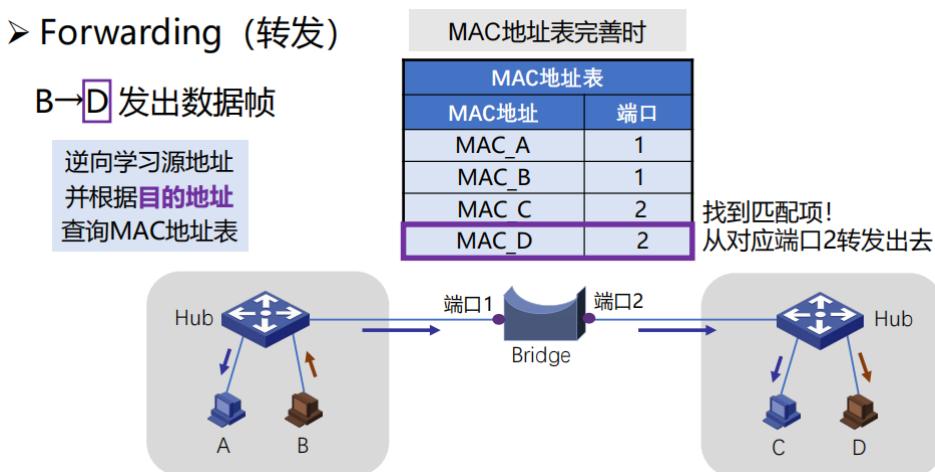


Figure 51

4.4.1.2.2 Filtering (过滤)

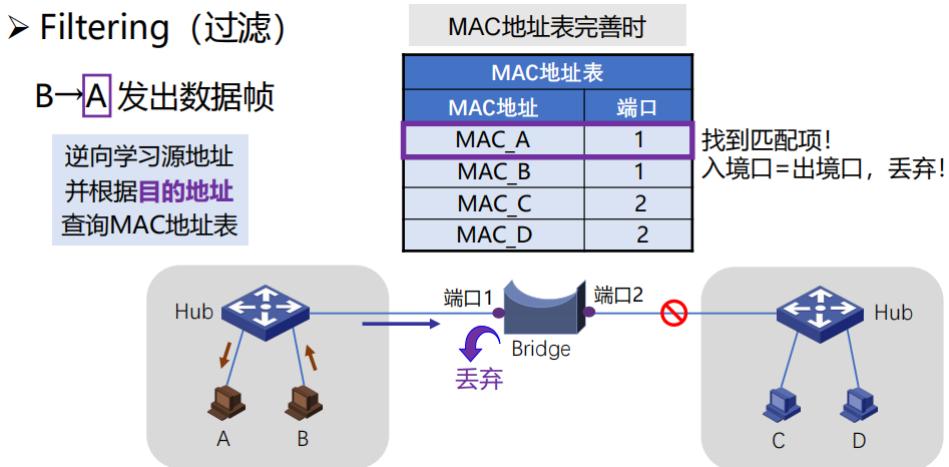
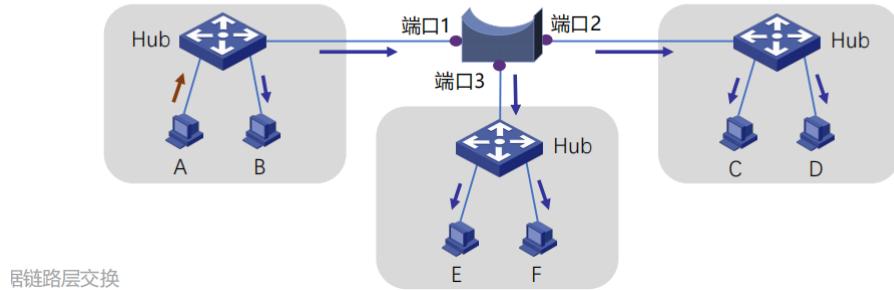


Figure 52

4.4.1.2.3 Flooding (泛洪)

➤ Flooding (泛洪)

A→B发出数据帧



找不到匹配表项！
从所有端口（除了入口）发送出去
一个网段的数据被发送到无关网段
存在安全隐患
浪费网络资源

86

Figure 53

两种目的地址的帧，需要泛洪：

- 广播帧：目的地址为 **FF-FF-FF-FF-FF-FF** 的数据帧
- 未知单播帧：目的地址不在 MAC 地址转发表中的单播数据帧

4.4.1.3 总结

逆向学习

- 根据帧的 **源地址** 在 MAC 地址表查找匹配表项，
- 如果没有，则 **增加** 一个新表项（源地址、入境端口、帧到达时间），
- 如果有，则 **更新** 原表项的帧到达时间，重置老化时间。

网桥对入境帧的转发过程（三选一），查帧的 **目的地址** 是否在 MAC 地址表中

- 如果有，且入境端口 ≠ 出境端口，则从对应的出境端口 **转发帧**；
- 如果有，且入境端口 = 出境端口，则丢弃帧（即 **过滤帧**）；
- 如果没有，则向除入境端口以外的其它所有端口 **泛洪帧**。

4.4.2 链路层交换机

4.4.2.1 交换方式

4.4.2.1.1 从带宽的角度

- 对称交换：出和入的带宽相同
- 非对称交换：出和入的带宽不同

4.4.2.1.2 从转发时机的角度

存储转发模式 (Store and Forward)

- 特点：转发前必须接收 **整个帧**、执行 CRC 校验
- 缺点：延迟大
- 优点：不转发出错帧、支持非对称交换

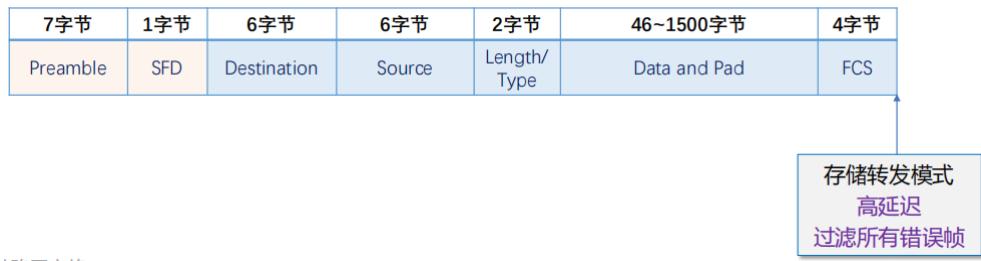


Figure 54

直通模式 (Cut-through)

- 特点：一旦接收到 **帧的目的地址**，就开始转发
- 缺点：可能转发错误帧、不支持非对称交换
- 优点：延迟非常小，可以边入边出

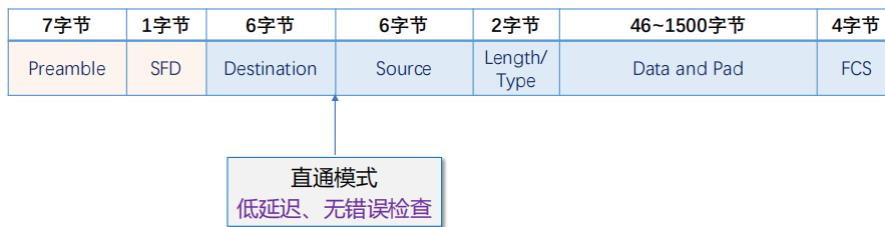


Figure 55

无碎片模式 (Fragment-free)

- 特点：接收到帧的 **前 64 字节**，即开始转发
- 缺点：仍可能转发错误帧，不支持非对称交换
- 优点：过滤了冲突碎片，延迟和转发错帧介于存储转发和直通交换之间



Figure 56

① Note

小于 64 字节的帧一定是以太网中冲突导致的

练习

在交换机的三种交换模式中，正常情况下(B)模式的转发延迟最大。

- 直通
- 存储转发
- 无碎片
- 都一样

Figure 57

4.4.3 生成树协议

可靠传输：冗余拓扑

付出的代价：导致物理环路

4.4.3.1 物理环路引发的问题

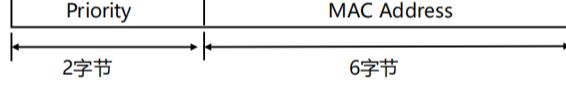
1. **广播风暴**：交换机（网桥）在物理环路上无休止地泛洪广播流量，无限循环，迅速消耗网络资源
2. **重复帧**：X发送到环路的单播帧，造成目的设备Y收到重复的帧（假设所有交换机的MAC地址表中均没有路由器Y的MAC地址）
3. **MAC地址表不稳定**：当一个帧的多个副本到达不同端口时，交换机会不断修改同一MAC地址对应的端口

4.4.3.2 生成树网桥

参与的交换机（网桥）：收发桥协议数据单元BPDU

桥协议数据单元BPDU包含的四个关键信息

- **根桥ID** (Root ID)：被选为根的桥ID。
 - 桥ID共8字节，由2字节的优先级和6字节的MAC地址组成的。



- **根路径开销** (Root Path Cost)：到根桥的最小路径开销。
- **指定桥ID** (Designated Bridge ID)：生成和转发BPDU的桥ID
- **指定端口ID** (Designated Port ID)：发送BPDU的端口ID。

Figure 58

4.4.3.2.1 生成树的三个选举过程

(1) 选举 **根桥**(Root Bridge)

- 生成树的选举过程1：选举根桥
广播域：broadcast或flooding能到的地方
- 同一广播域中的所有交换机均参与选举；
 - 桥ID最小的交换机（网桥）成为生成树的根；
 - 在给定广播域内只有一个根桥，其它均为非根桥。
 - 根桥的所有端口都处在**转发状态**。

可以接收和发送数据帧

Figure 59

① Note

如何比较桥 ID 大小？

- 首先比较优先级，优先级数值最小的交换机胜出成为根桥。
- 如果优先级数值相等，MAC地址最小的交换机成为根桥。

(2) 为每个非根桥选出一个 **根端口** (Root Port)

➤ 生成树的选举过程2：为每个非根桥选出一个根端口

- 每个非根桥，通过**比较**其每个端口到根桥的**根路径开销**，选出根端口；
- 具有**最小根路径开销**的端口被选作根端口；即 到达根最近的端口
- 如果多个端口的根路径开销相同，则**端口ID最小**的端口被选作根端口；
- 非根桥只能有一个根端口，根端口处于**转发状态**。

Figure 60

根路径开销

- **根桥**的根路径开销为**0**；
- **非根桥**的根路径开销为到根桥的路径上所有端口（链路）**开销之和**。
- 端口（链路）开销值由IEEE定义（如下表），也可通过手工配置改变

速率值	开销 (IEEE802.1D-1998)
10Mbps	100
100Mbps	19
1Gbps	4
10Gbps	2
>10Gbps	1

Figure 61

根端口选举实例

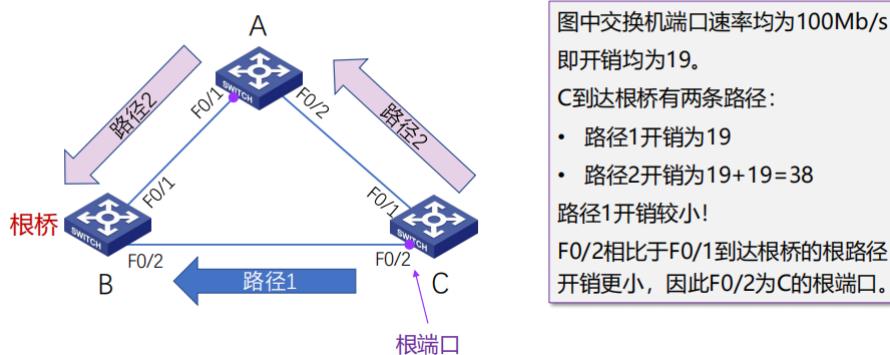


Figure 62

(3) 为每个网段确定一个 **指定端口** (Designated Port)

➤ 生成树的选举过程3：为每个网段确定一个指定端口

- 对于每一个网段，在所有连接到它的交换机（网桥）端口中进行选择；
- **一个具有最小根路径开销的端口**，作为该网段的**指定端口**；
- 指定端口处于**转发状态**，负责该网段的数据转发；
- 连接该网段的其他端口，若**既不是指定端口，也不是根端口**，则**阻塞**。

Figure 63

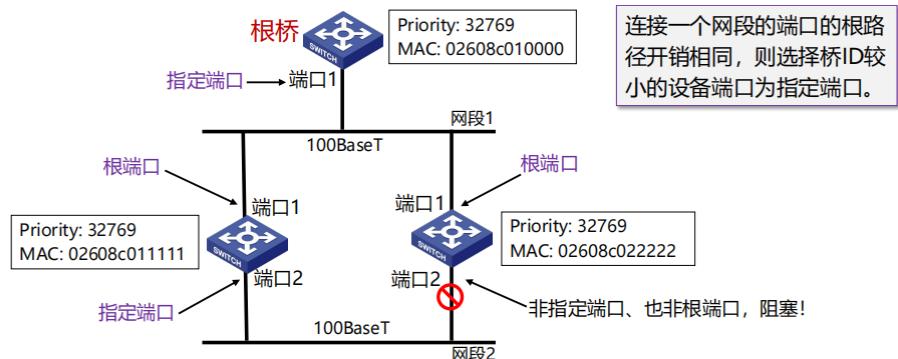


Figure 64

端口角色	英文名称	端口状态
指定端口	Designated port	Forwarding
根端口	Root port	Forwarding
非指定端口/根端口 (通常称为备用端口或冗余端口)	Alternate port	Blocking

练习：根桥的所有连接端口都是指定端口。 (判断题) 对

Figure 65

4.4.3.2.2 重构生成树

➤ 生成树的某“枝”断掉了，怎么办？

- 当由交换机（网桥）或链路故障导致网络拓扑改变时，重新构造生成树。

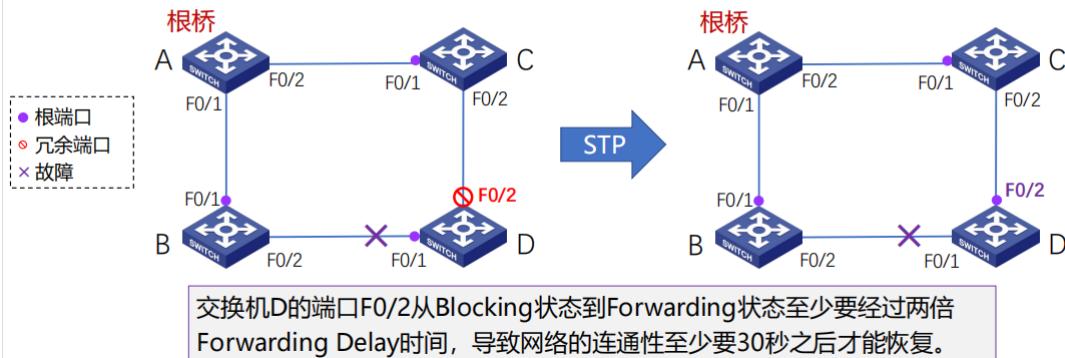


Figure 66

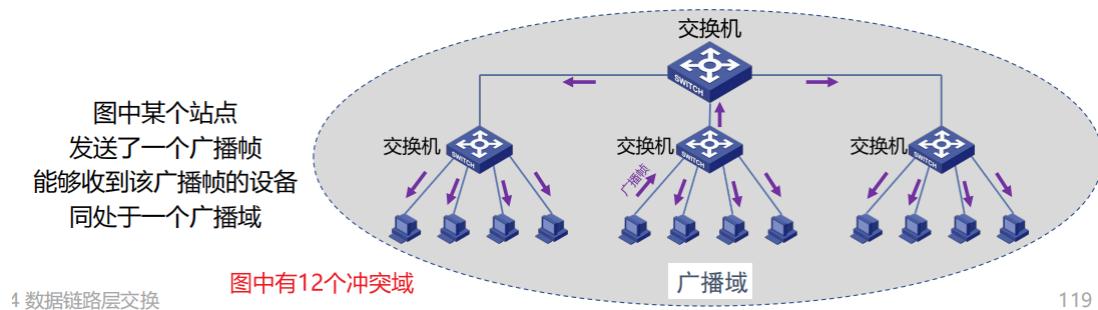
重新构建生成树太慢了，怎么办？

- 快速生成树协议 (Rapid Spanning Tree Protocol, RSTP)
- RSTP 是 STP 的优化版，在 IEEE802.1W 中定义；

4.4.4 虚拟局域网

广播域 (Broadcasting Domain)

- 广播域是广播帧能够到达的范围；
- 缺省情况下，交换机所有端口同属于一个广播域，无法隔离广播域；
- 广播帧在广播域中传播，占用资源，降低性能，且具有安全隐患



119

Figure 67

交换机可以分隔广播域吗？

- 可以！支持 VLAN 的交换机；
- 一个 VLAN (Virtual LAN) 是一个独立的广播域；

- 交换机通过划分 VLAN，来分隔广播域。

VLAN 是一个在物理网络上根据用途、工作组、应用等来 **逻辑划分** 的局域网络，与用户的物理位置没有关系。

- 不同 VLAN 的成员不能直接进行二层通信
- 不同 VLAN 的成员通信需要通过三层设备

4.4.4.1 VLAN 类型

4.4.4.1.1 基于端口的 VLAN (最常见)

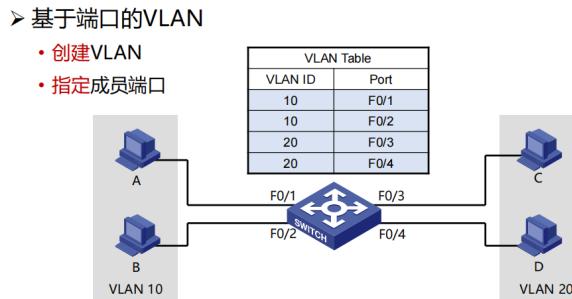


Figure 68

4.4.4.1.2 基于 MAC 地址的 VLAN

➤ 基于MAC地址的VLAN

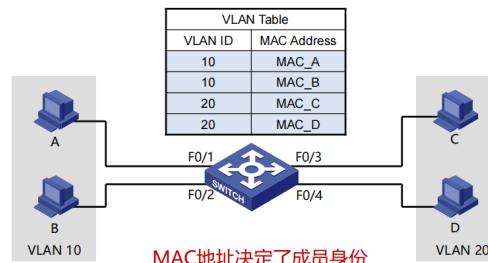


Figure 69

4.4.4.1.3 基于协议的 VLAN

通常需要服务器的参与

4.4.4.1.4 基于子网的 VLAN

一个子网就是一个 VLAN

如何区分不同 VLAN 的数据帧？

- 在数据帧中携带 VLAN 标记；
- VLAN 标记由交换机添加/剥除，对终端站点透明

标记帧 (Tagged Frame)

➤ 帧标记标准：IEEE802.1Q

- 带VLAN标记的帧称为**标记帧** (Tagged Frame)
- 不携带VLAN标记的普通以太网帧称为**无标记帧** (Untagged Frame)

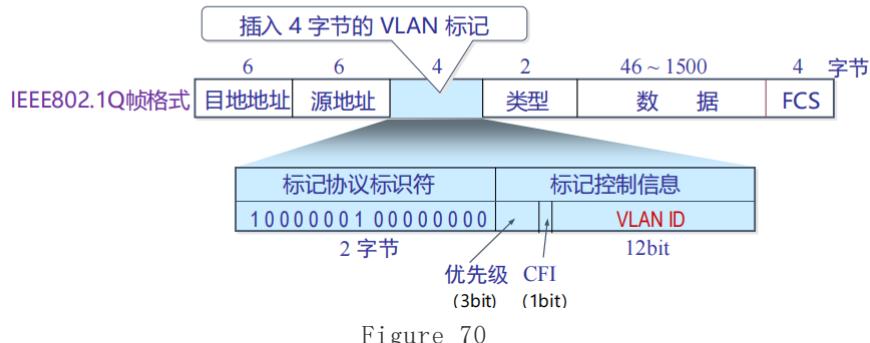


Figure 70

4.4.4.1.5 VLAN 优点

有效控制广播域范围：广播流量被限制在一个 VLAN 内；

增强网络的安全性：VLAN 间相互隔离，无法进行二层通信，不同 VLAN 需通过三层设备通信；

灵活构建虚拟工作组：同一工作组的用户不必局限于同一物理范围；

提高网络的可管理性：将不同的业务规划到不同 VLAN 便于管理

4.4.4.2 Access 链路类型端口

4.4.4.3 Trunk 链路类型端口与 Trunk 链路

4.5 无线局域网

4.5.1 无线局域网概述

无线局域网 (Wireless Local Area Network, WLAN)：指以无线信道作为传输介质的计算机局域网

4.5.2 无线局域网组网模式

基础架构模式

- 分布式系统 (DS)
- 访问点 (AP)
- 站点 (STA)
- **基本服务集 (BSS)**
- 扩展服务集 (ESS)
- 站点之间通信通过 AP 转发

自组织模式 (Ad hoc)

- 站点 (STA)
- 独立基本服务集 (IBSS)
- 站点之间直接通信
- 共享同一无线信道

4.5.2.1 无线局域网体系结构

4.5.2.2 802.11 物理层

物理层技术概览

- 频段: 2.4GHz、5GHz (ISM频段, 无需授权; 限制发送功率, 例如: ≤ 1瓦)
- 调制技术: DPSK → QPSK → CCK → 64-QAM → 256-QAM → 1024-QAM
- 直接序列扩频 (DSSS) → 正交频分多路复用 (OFDM) → 正交频分多址 (OFDMA)
- 单天线 → 单用户多入多出 (SU-MIMO) → 多用户多入多出 (MU-MIMO)
- 目标: 提升传输速率、增强可靠性、支持高密度接入

逻辑链路控制子层 (LLC)						
介质访问控制子层 (MAC)						
802.11 最高速率 2Mbps	802.11b 最高速率 11Mbps	802.11a 最高速率 54Mbps	802.11g 最高速率 54Mbps	802.11n 最高速率 600Mbps	802.11ac 最高速率 3.47Gbps	802.11ax 最高速率 9.6Mbps

Figure 71

4.5.3 802.11 介质访问控制

传输范围..?

➤ 隐藏终端问题

- 由于距离太远 (或障碍物) 导致站点无法检测到竞争对手的存在
- 隐藏站点不能侦听到发送端但能干扰接收端
- 假设: A正在向B传输数据, C也要向B发送数据

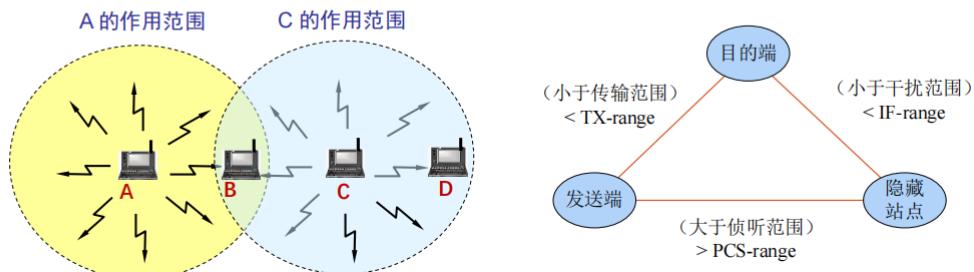


Figure 72

➤ 暴露终端问题

- 由于侦听到其他站点的发送而误以为信道忙导致不能发送
- 暴露站点能侦听到发送端但不会干扰接收端
- 假设：B正在向A传输数据，C要向D发送数据

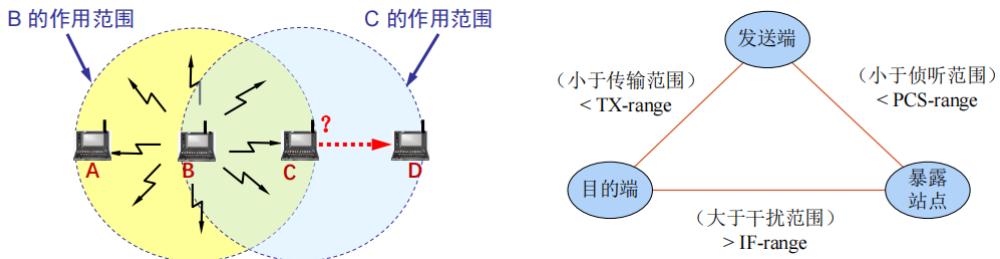


Figure 73

4.5.3.1 CSMA/CA

带有冲突避免的 CSMA Carrier Sense Multiple Access with Collision Avoid

- 当信道空闲时间大于 IFS (帧间隙)，立即传输
- 当信道忙时，延迟直到当前传输结束+IFS 时间
- 开始随机退后过程
 - 从 (0, CWindow) 中选择一个随机数作为退后计数器 (backoff counter)
 - 通过侦听确定每个时间槽是否活动
 - 如果没有活动，则减少退后时间
- **退后过程中如果信道忙，则挂起退后过程** (解决站点之间的公平问题)
- 在当前帧传输结束后恢复退后过程

Note

使用退后过程延迟发送的目的：避免多个站点同时传输引起的冲突

CWindow：竞争窗口

WIFI 采用停等机制

➤ 不同帧间隙控制优先级

- SIFS (Short IFS) : 最高优先级，用于Ack, CTS, 轮询响应等
- PIFS (PCF IFS) : 中等优先级 (SIFS+1槽口时间) , 轮询服务
- DIFS (DCF IFS) : 最低优先级 (SIFS+2槽口时间) , 异步数据服务

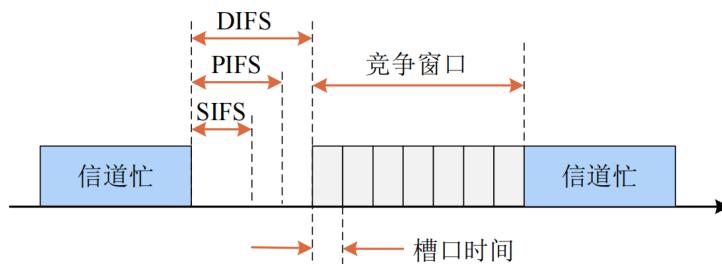


Figure 74

- 越短的帧间隙，越能抢到信道

4.5.3.2 RTS-CTS 机制（可选机制）

目的：通过信道预约，避免长帧冲突

- 发送端发送 RTS (request to send)
- 接收端回送 CTS (clear to send)
- RTS 和 CTS 中的持续时间 (Duration) 中指明传输所需时间 (数据+控制)
- 其他相关站点能够收到 RTS 或 (和) CTS，维护 NAV(Network Allocation Vector)，静默期
- 虚拟载波侦听 (Virtual Carrier Sense)
- RTS 和 CTS 帧很短，即使产生冲突，信道浪费较少

如何应对无线链路较高的出错率？

- 解决方法：采用较小的帧（将用户数据帧分段的机制对用户透明）

4.5.4 802.11 帧结构

➤ 802.11帧格式一般结构

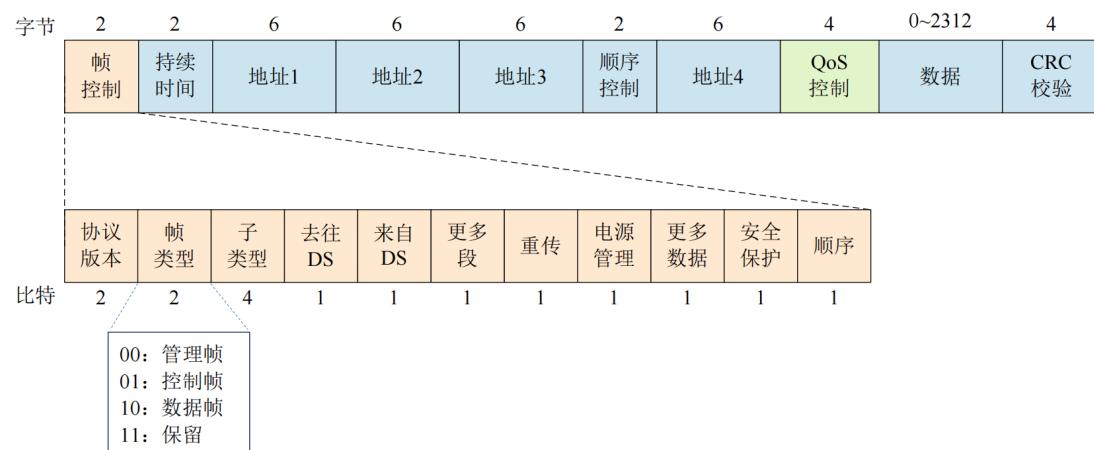


Figure 75

➤ 主要字段解释

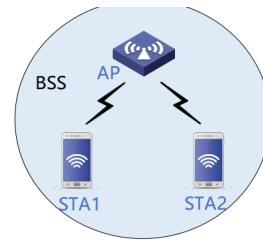
- 帧控制：具有多种用途
- 持续时间：下一个要发送帧可能持续的时间 (NAV) 或关联ID (AID)
- 地址1~地址4：每个地址的含义基于“去往DS”和“来自DS”域段确定
- 顺序控制：过滤掉重复帧，或用于分片组合
- QoS控制域段：存放数据流的QoS信息 (802.11e中扩展)
- 数据：包含任意长度的数据 (0-2312字节)
- CRC校验：802.11采用4个字节的校验码

Figure 76

4.5.5 无线局域网的构建与管理

➤ 基础架构模式

- 通过AP接入有线网络（互联网络）
- **关键：如何关联到AP？**
 - BSSID：AP的MAC地址，标识AP管理的基本服务集
 - SSID：32字节网名，标识一个扩展服务集（ESS），包含一个或多个基本服务集
- 关联到AP的三个阶段
 - 扫描（Scan）、认证（Authentication）、关联（Association）



BSSID: Basic Service Set Identifier

SSID: Service Set Identifier

胖AP: Fat AP, 功能全面

瘦AP: Fit AP, 配合无线交换机组网

Figure 77

被动扫描

- AP周期性发送Beacon帧，站点在每个可用的通道上扫描Beacon帧

主动扫描

- 站点依次在每个可用的通道上发出包含SSID的Probe Request帧，具有被请求SSID的AP返回Probe Response帧

➤ 认证过程

- 当站点找到与其有相同SSID的AP，在SSID匹配的AP中，根据收到的AP信号强度，选择一个信号最强的AP，然后进入认证阶段
- 主要认证方式包括：
 - 开放系统身份认证 (open-system authentication)
 - 共享密钥认证 (shared-key authentication)
 - WPA PSK认证 (pre-shared key)
 - 802.1X EAP认证

Figure 78

关联过程

- 身份认证获得通过后，进入关联阶段
- 站点向AP发送关联请求（Association Request）
 - 包含：Capability, Listen Interval, SSID, Supported Rates
- AP向站点返回关联响应（Association Response）
 - 包含：Capability, Status Code, Station ID, Supported Rates
- AP维护站点关联表，并记录站点的能力（如能够支持的速率等）

➤ 站点漫游

- 当前的AP的通道质量下降时，站点漫游到不同的AP
- 通过扫描功能发现通道质量更好的AP
 - 被动扫描
 - 主动扫描
- 站点向新的AP发送**重关联请求 (Reassociation Request)**
- 如果AP接受重关联请求
 - AP 向站点返回**重关联响应 (Reassociation Response)**
 - 如果重关联成功，则站点漫游到新的AP
 - 新的AP通过分布系统通知之前的AP

Figure 79

站点睡眠管理

- 目的：延长电池的续航时间

4.5.6 Wi-Fi6 核心技术概览

Wi-Fi 6 (802.11ax) 核心目标：解决网络容量和传输效率问题、降低传输时延，相对于 Wi-Fi 5，在高密部署场景中将用户平均吞吐量提升 4 倍以上，并发用户数提升 3 倍以上

5. 网络层

Important

目标：

5.1 网络层服务

5.1.1 网络层服务概述

Figure 80

网络层实现端系统间的**多跳传输**

网络层功能存在每台主机和路由器中

- 发送端：
- 接收端：
- 路由器：

5.1.2 网络层的关键功能

5.1.2.1 路由（控制面）

选择数据从源端到接收端

- 核心：

5.1.3 提供给传输层的服务

网络层提供可靠性，面向连接（虚电路）

端系统提供可靠性，面向无连接（数据报）

5.1.3.1 无连接服务

如寄信

数据报服务

- 4点

尽力而为交付

-

优点：造价大大降低

5.1.3.2 连接服务

如打电话

虚电路(virtual circuit)是逻辑链接

面向连接也不一定能保证数据传输的可靠性

Figure 81

5.1.4 虚电路和数据报网络的性能比较

5.1.4.1 1. 性能角度

5.1.4.2 2. 效率角度

5.2 Internet 网际协议

5.2.1 IPv4 报文格式

- **版本**: 4bit, 表示采用的IP协议版本
- **首部长度**: 4bit, 表示整个IP数据报首部的长度
- **区分服务**: 8bit, 该字段一般情况下不使用
- **总长度**: 16bit, 表示整个IP报文的长度,能表示的最大字节为 $2^{16}-1=65535$ 字节
- **标识**: 16bit, IP软件通过计数器自动产生, 每产生1个数据报计数器加1; 在ip分片以后, 用来标识同一片分片
- **标志**: 3bit, 目前只有两位有意义; MF, 置1表示后面还有分片, 置0表示这是数据报片的最后1个; DF, 不能分片标志, 置0时表示允许分片
- **片偏移**: 13bit, 表示IP分片后, 相应的IP片在总的IP片的相对位置



Figure 82

- **生存时间TTL(Time To Live)** : 8bit, 表示数据报在网络中的生命周期, 用通过路由器的数量来计量, 即跳数 (每经过一个路由器会减1)
- **协议**: 8bit, 标识上层协议 (TCP/UDP/ICMP...)
- **首部校验和**: 16bit , 对数据报首部进行校验, 不包括数据部分
- **源地址**: 32bit, 标识IP片的发送源IP地址
- **目的地址**: 32bit, 标识IP片的目的地IP地址
- **选项**: 可扩充部分, 具有可变长度, 定义了安全性、严格源路由、松散源路由、记录路由、时间戳等选项
- **填充**: 用全0的填充字段补齐为4字节的整数倍

Figure 83

5.2.1.1 数据包分片

MTU (Maximum Transmission Unit) , 最大传输单元

分片策略

- 允许途中分片: 根据下一跳链路的 MTU 实施分片
- 不允许途中分片: 发出的数据报长度小于路径 MTU (路径 MTU 发现机制)

重组策略

- 途中重组, 实施难度太大
- 目的端重组 (互联网采用的策略)
- 重组所需信息: 原始数据报编号、分片偏移量、是否收集所有分片

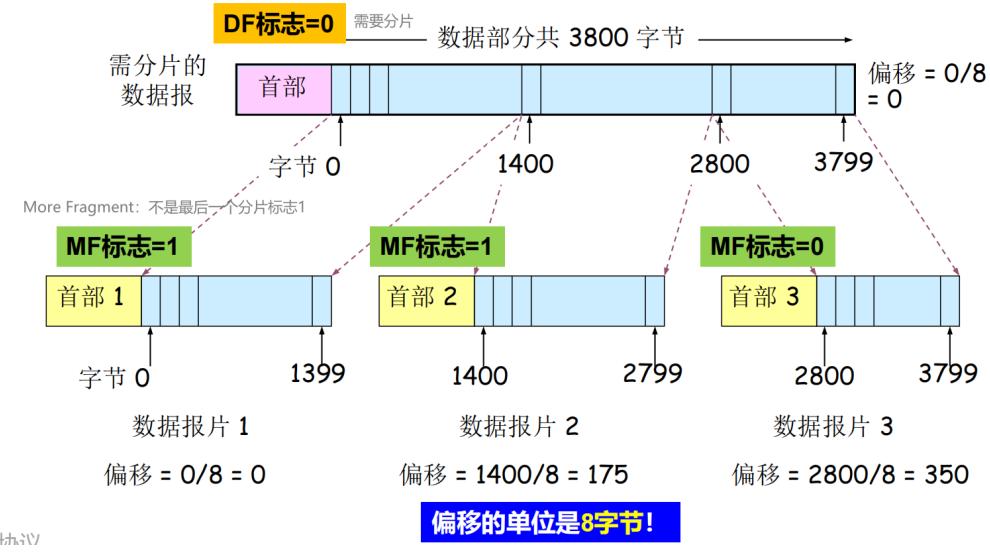


Figure 84

网络层基本功能

- 支持多跳寻路将IP数据报送达目的端：**目的IP地址**
- 表明发送端身份：**源IP地址**
- 根据IP头部协议类型，提交给不同上层协议处理：**协议**

其它相关问题

- 数据报长度大于传输链路的MTU的问题，通过分片机制解决：**标识、标志、片偏移**
- 防止循环转发浪费网络资源（路由错误、设备故障...），通过跳数限制解决：**生存时间TTL**
- IP报头错误导致无效传输，通过头部校验解决：**首部校验和**

Figure 85

5.2.1.2 IP 地址

IP 地址，网络上的每一台主机（或路由器）的每一个接口都会分配一个全球唯一的 32 位的标识符

- 👉 将 IP 地址划分为固定的类，每一类都由两个字段组成
- 👉 网络号相同的这块连续 IP 地址空间称为地址的前缀，或网络前缀

地址	用途
全0网络地址	只在系统启动时有效，用于启动时临时通信，又叫主机地址
网络127.0.0.0	指本地节点(一般为127.0.0.1)，用于测试网卡及TCP/IP软件，这样浪费了1700万个地址
全0主机地址	用于指定网络本身，称之为网络地址或者网络号
全1主机地址	用于广播，也称定向广播，需要指定目标网络
0.0.0.0	指任意地址
255.255.255.255	用于本地广播，也称有限/受限广播，无须知道本地网络地址

Figure 86

Q

A large number of consecutive IP addresses are available starting at 202.101.0.0. Suppose that five organizations, A, B, C, D, and E, request 1024, 2000, 2000, 4096 and 512 addresses (including special IPs), respectively, and in that order. Please assign the IP address and the mask in the w.x.y.z/s notation (following the order of allocating the minimum address segment firstly).

Figure 87

5.2.1.3 最长前缀匹配

最长前缀匹配 (Longest prefix match)

👉 CIDR 可变长子网掩码以及路由聚合，需要最长前缀匹配来实现最精确匹配

👉 IP 地址与 IP 前缀匹配时，总是选取 子网掩码最长的匹配项

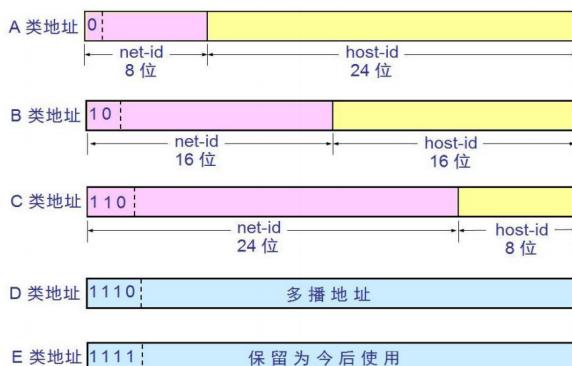
2.128.0.0/9	interface 1	2.1000 0000
2.192.0.0/10	interface 2	2.1100 0000
2.0.0.0/8	interface 3	2.0000 0000
2.2.3.0/24	interface 4	2.0000 0010
0.0.0.0/0	interface 5	适配任何

根据最长前缀匹配，下述目的IP将匹配哪个表项（出接口）？

2.5.1.2	Interface 3	2.200.1.2	Interface 2
2.0000 0101		2.1100 1000	
2.150.1.2	Interface 1	3.150.1.2	Interface 5
2.1001 0110			

Figure 88

- IP地址共分为A、B、C、D、E五类，A类、B类、C类为单播地址
- IP地址的书写采用**点分十进制**记法，其中每一段取值范围为0到255



请判断下列地址的类型

10.2.1.1	A类
128.63.2.100	B类
201.222.5.64	C类
256.241.201.10	不存在，超出范围

Figure 89

5.2.1.4 IPv4 地址如何获取

- 公有IP地址要求全球唯一

- ICANN (Internet Corporation for Assigned Names and Numbers) 即互联网名字与编号分配机构向ISP分配，ISP再向所属机构或组织逐级分配

- 静态设定

- 申请固定IP地址，手工设定，如**路由器、服务器**

- 动态获取

- 使用DHCP协议或其他动态配置协议
- 当**主机**加入IP网络，允许主机从DHCP服务器动态获取IP地址
- 可以有效利用IP地址，方便移动主机的地址获取

Figure 90

5.2.2 DHCP 动态主机配置协议

DHCP：动态主机配置协议

- 当主机加入 IP 网络，允许主机从 DHCP 服务器动态获取 IP 地址
- 可以有效利用 IP 地址，方便移动主机的地址获取

工作模式：客服/服务器模式（C/S）

- 基于 UDP 工作**，服务器运行在 67 号端口，客户端运行在 68 号端口

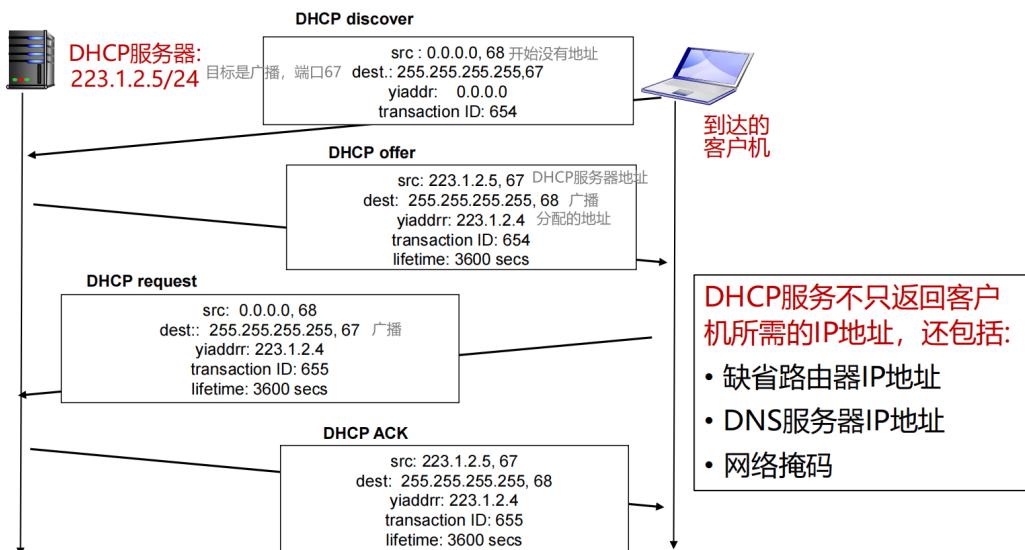


Figure 91

1. DHCP 客户从 UDP 端口 68 以广播形式向服务器发送发现报文 (DHCPDISCOVER)
2. DHCP 服务器单播发出提供报文 (DHCPOFFER)
3. DHCP 客户从多个 DHCP 服务器中选择一个，并向其以广播形式发送 DHCP 请求报文 (DHCPREQUEST)
4. 被选择的 DHCP 服务器单播发送确认报文 (DHCPACK)

阶段	源MAC	目标MAC	源IP	目标IP	传输形式
Discover	PC机的MAC	全FF	0.0.0.0	255.255.255.255	广播
Offer	DHCP服务器或者中继器路由的 MAC	DHCP客户机的MAC	DHCP服务器或者中继路由器的IP地址	准备分配的IP地址	单播
Request	PC机的MAC	全FF	0.0.0.0	255.255.255.255	广播
Ack	DHCP服务器或者中继器路由的 MAC	DHCP客户机的MAC	DHCP服务器或者中继路由器的IP地址	准备分配的IP地址	单播

Figure 92

5.2.2.1 IP 与 MAC 地址

5.2.3 ARP 地址解析协议 (Address Resolution Protocol)

5.2.3.1 IP 包转发

直接交付：与目的主机在同一个 IP 子网内

5.2.4 NAT

网络地址转换(NAT)用于解决 IPv4 地址不足的问题，是一种将私有（保留）地址转化为公有 IP 地址的转换技术

内网地址可以重复（在不同内网都是指不同的地址），到外网由 NAT 边界路由器进行 IP 地址转换

私有 IP 地址（内网地址）：

- A 类地址：10.0.0.0--10.255.255.255
- B 类地址：172.16.0.0--172.31.255.255
- C 类地址：192.168.0.0--192.168.255.255

5.2.4.1 NAT 工作机制

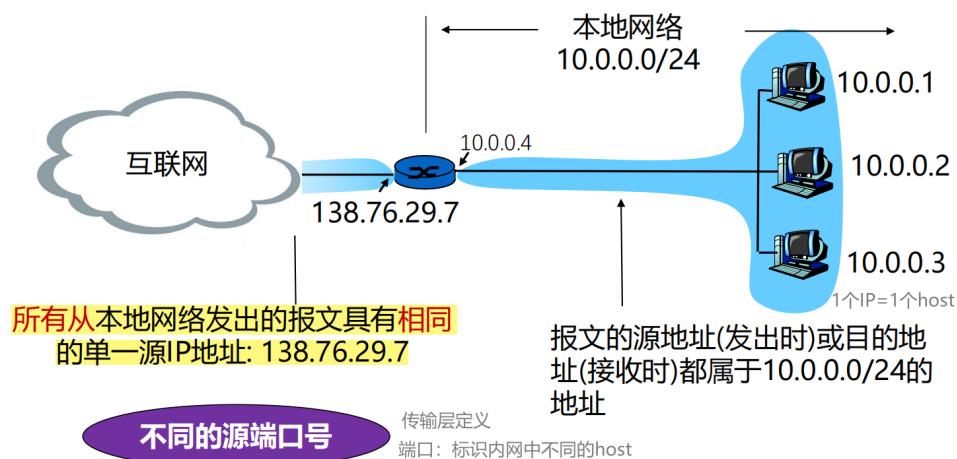


Figure 93

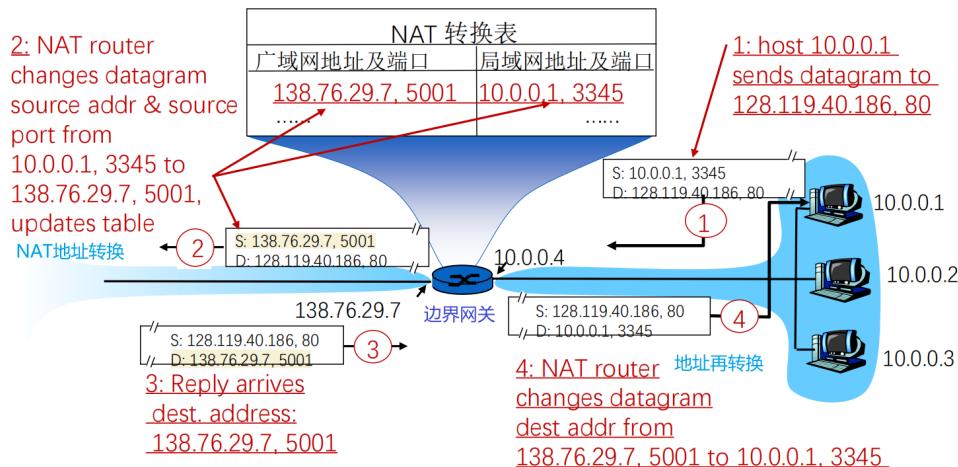


Figure 94

Note

同一主机不同应用，或者不同主机的同一端口，NAT 转换如何处理？换端口（区分到每台主机的不同应用）

出数据报：外出数据报用 NAT IP 地址(全局), 新 port # 替代 源 IP 地址(私有), port #

NAT 转换表：每个 (源 IP 地址, port #) 到(NAT IP 地址, 新 port #) 映射项

入数据报：对每个入数据报的地址字段用存储在 NAT 表中的(源 IP 地址, port #)替代对应的 (NAT IP 地址, 新 port #)

端口号由两个字节 (16bit) 表示

5.2.4.2 NAT 优缺点

NAT 的优势

- 节省合法地址，减少地址冲突
- 灵活连接 Internet
- 保护局域网的私密性

问题或缺点

- 违反了 IP 的结构模型，路由器处理传输层协议（三层的路由器看四层的端口）
- 违反了端到端的原则
- 违反了最基本的协议分层规则
- 不能处理 IP 报头加密
- 新型网络应用的设计者必须要考虑 NAT 场景，如 P2P 应用程序

外网的 client 要连内网的 host? relaying

➤ One possible solution: *relying* (used in Skype)

- NATed client establishes connection to relay
- External client connects to relay
- relay bridges packets between two connections

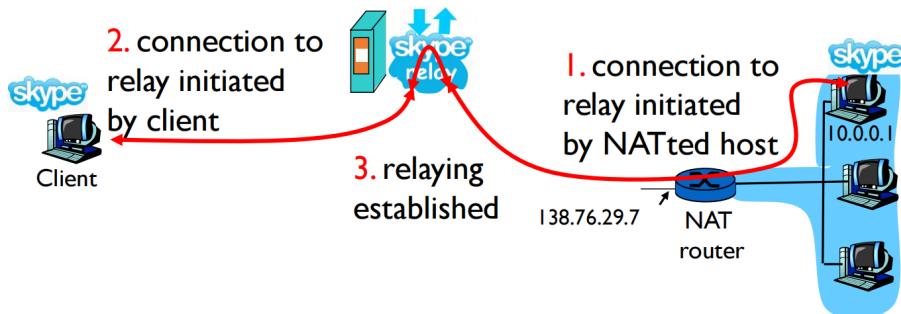


Figure 95

5.2.5 Internet 控制报文协议

ICMP: 互联网控制报文协议

ICMP 报文类型

- ICMP 差错报告报文
- 终点不可达：不可达主机、不可达网络，无效端口、协议

- ICMP 询问报文
- 回送请求/回答 (ping 使用)

5.2.5.1 PING (Packet InterNet Groper)

- PING 用来测试两个主机之间的连通性
- PING 使用了 ICMP 回送请求与回送回答报文
- 可以知道 连通性、往返时延、单向转发跳数

5.2.5.2 Traceroute 和 ICMP

如何知道整个路径上路由器的地址？使用 TraceRT 命令

这个方法不精确：互联网上的路由是独立的

5.3 路由算法

路由算法须满足的特性：

- 正确性、简单性、鲁棒性、稳定性、公平性、有效性

根据路由算法是否随网络的通信量或拓扑自适应划分

- 静态路由选择策略（非自适应路由选择）
- 动态路由选择策略（自适应路由选择）

5.3.1 优化原则

汇集树(Sink Tree)

- 所有的源节点到一个指定目标节点的最优路径的集合构成一棵以目标节点为根的树
- 一棵路由器 B 的汇集树（距离度量单位：步长数）

汇集树不是唯一的

5.3.2 最短路径算法

Dijkstra 算法

5.3.3 距离向量路由

Bellman-Ford 方程

假设 $Dx(y)$ 是从 x 到 y 最小代价路径的代价值；其中 m 为 x 的邻居, $c(x, m)$ 为 m 到 X 的距离

$$Dx(y) = \min c(x, m) + Dm(y)$$

距离向量 (Distance Vector) 算法基本思想：

- 每个节点周期性地向邻居发送它自己到某些节点的距离向量；
- 当节点 x 接收到来自邻居的新 DV 估计，它使用 B-F 方程更新其自己的 DV：

上述过程迭代执行， $Dx(y)$ 收敛为实际最小费用 $dx(y)$

距离向量算法 特点：迭代的、分布式的

缺点 drawback of DV:

- It reacts rapidly to good news, but leisurely to bad news. 对好消息反应快，坏消息反应慢

- It is known as the count-to-infinity problem

Attempts to solve **Count-to-infinity** problem —— **Poisoned reverse** (毒性逆转, RFC 1058) --> 但不能完全解决这个问题

5.3.4 链路状态路由

链路状态 (Link State) 路由可分为五个部分：

1. 发现邻居，了解他们的网络地址；
2. 设置到每个邻居的成本度量；
3. 构造一个分组，分组中包含刚收到的所有信息；
4. 将此分组发送给其他的路由器；
5. 计算到其他路由器的最短路径。

➤ 距离向量和链路状态算法比较：

- | | |
|--|--|
| <ul style="list-style-type: none"> • 网络状态信息交换的范围 <ul style="list-style-type: none"> • DV: 邻居间交换 • LS: 全网扩散 • 网络状态信息的可靠性 <ul style="list-style-type: none"> • DV: 部分道听途说 • LS: 自己测量 | <ul style="list-style-type: none"> • 健壮性: <ul style="list-style-type: none"> • DV: 计算结果传递，健壮性差 • LS: 各自计算，健壮性好 • 收敛速度: <ul style="list-style-type: none"> • DV: 慢，可能有计数到无穷问题 • LS: 快 |
|--|--|

Figure 96

5.3.5 层次路由

产生原因：

- 过于庞大的路由表存储、查找困难，路由信息交互开销高
- 为提高路由器查表速度，减少路由表存储空间，需要缩减路由条目
- 可以通过地址聚合进一步缩减路由条目

基本思路：

互联网由大量不同的网络互连，每个管理机构控制的网络是自治的

自治系统 (AS, Autonomous System)

- 一个管理机构控制之下的网络
- 一个 AS 内部通常使用相同的路由算法/路由协议，使用统一的路由度量（跳数、带宽、时延 ...）
- 不同的 AS 可以使用不同的路由算法/路由协议
- 每个 AS 有一个全球唯一的 ID 号：AS ID
- 自治系统内的还可以进一步划分层次：私有自治系统或区域

自治系统内部使用内部网关路由协议 Interior Gateway Protocols (IGP)

自治系统之间使用外部网关路由协议 Exterior Gateway Protocols (EGP)

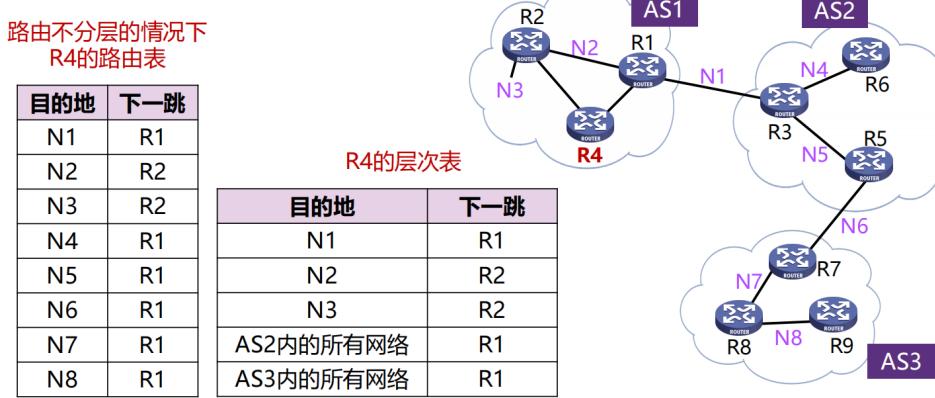


Figure 97

5.3.6 广播路由 Broadcasting

广播 (Broadcasting) : 源主机同时给全部目标地址发送同一个数据包

方法 3: 泛洪 (flooding)

- 一种将数据包发送到所有网络节点的简单方法
- 将每个进入数据包发送到除了进入线路外的每条出去线路
- 用途
 - 保证性: 一种有效广播手段, 可确保数据包被传送到网络中每个节点
 - 鲁棒性: 即使大量路由器被损坏, 也能找到一条路径 (如果存在)
 - 简单性: 仅需知道自己的邻居
- 无控制的泛洪: 环路可能导致广播风暴
- 解决方法: 受控制的泛洪 (每个路由器进行有选择的泛洪)
 - 序号控制泛洪 (sequence-number-controlled flooding)
 - 记录每个来源的广播序号

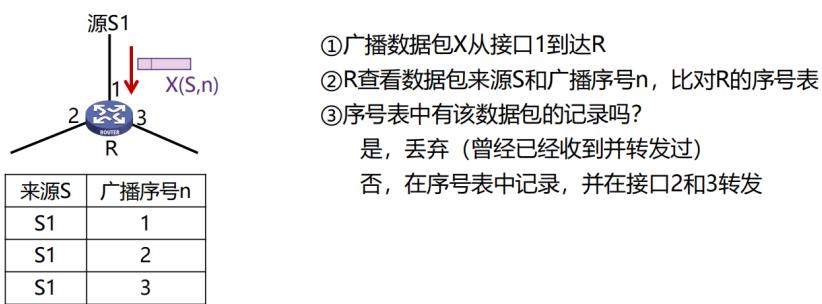


Figure 98

- 逆向路径转发 (reverse path forwarding, RPF)

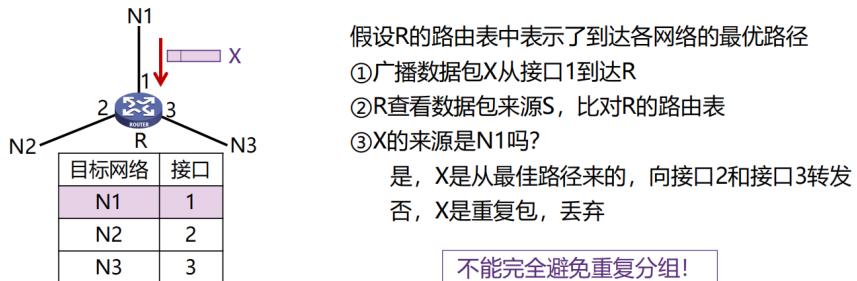
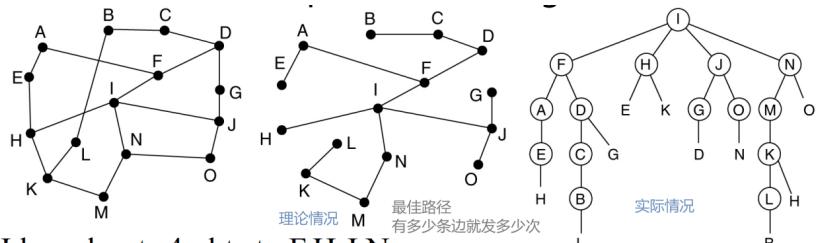


Figure 99



I broadcasts 4 pkts to F,H,J,N

All on preferred paths, ...

Look at N:

Transmits to M and O

Look at M: M continues since N-M is preferred (in sink tree)

由 Look at O: O stops since N-O is not on preferred path

Figure 100

方法 4: 生成树 (spanning tree)

- 源节点向所有属于该生成树的特定链路发送分组
- 改进了逆向路径转发
- 没有环路
- 最佳使用带宽
- 最少副本，消除了冗余分组
- 一个路由器可以不必知道整颗树，只需要知道在一颗树中的邻居即可

5.3.7 组播路由 multicasting

组播 (multicasting) : 源主机给网络中的一部分目标用户发送数据包

组播实现的两个步骤:

- 确定组成员
- 生成树——路由器与路由器之间的协议
 - 最佳生成树的使用取决于组的密度分布
 - 密集分布，基于源点树 (source-based trees)

密集分布，基于源点树 (source-based trees)

- 链路状态路由算法：每个路由器针对组内的每个发送者构造一颗独立树，例如多播开放最短通路优先协议 (Multicast Open Shortest Path First, MOSPF)
- 距离向量路由算法：逆向路径转发，修剪没有组成员的路由器，例如距离向量多播路由协议 (Distance Vector Multicast Routing Protocol, DVMRP)、协议无关多播-稠密模式 (Protocol Independent Multicast - Dense Mode, PIM-DM)

协议无关指的是与单播路由协议无关，即PIM不需要维护专门的单播路由信息

Figure 101

■ 存在的问题：

- 大型网络中，组播源很多时，路由器需生成多棵大树，工作量巨大
- 路由器需要大量空间来存储多棵大树

- 稀疏分布，基于核心树 (core-based trees)

稀疏分布，基于核心树 (core-based trees)

- 多个组播源共享树
- 大大节省存储开销、消息发送和计算
- 每个路由器只需要保存一棵树
- 不属于共享树的路由器不需要为组做任何工作
- 例如协议无关多播-稀疏方式 (Protocol Independent Multicast - Sparse Mode, PIM-SM)，特定源组播 (Protocol Independent Multicast - source-specific multicast, PIM-SSM)

Figure 102

- 基于核心树 (core-based trees) 存在的问题：

- 可能无法达到最优
- 如果只有一个发送者，将发送者作为核心是最优的

IGMP (Internet Group Management Protocol): 路由器获悉该网段的组播组成员

常用组播地址段：224.0.0.0/24

局域网组播地址(一跳子网内使用)

- 224.0.0.1 LAN上所有设备
- 224.0.0.2 LAN上所有路由器
- 224.0.0.5 LAN上所有OSPF路由器
- 224.0.0.251 LAN上所有DNS服务器

Figure 103

5.3.8 选播路由 Anycast

选播 (Anycast)

- 将数据包传送给最近的一个组成员
- 在有多个服务器的情况下，用户希望快速获得正确信息，而不在乎从哪个服务器获得

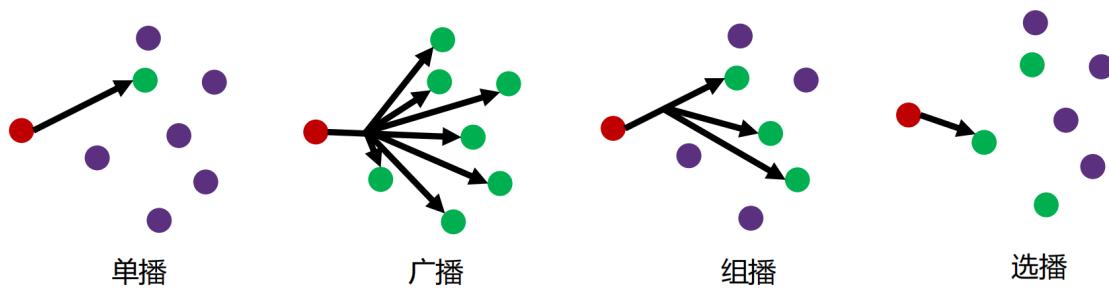


Figure 104

选播的典型应用：DNS

- 在没有指定 DNS 服务器的情况下，用户将始终连接到“最接近”(从路由协议角度来看) 服务器，可以减少延迟，并提供一定程度的负载平衡

5.4 Internet 路由协议

5.4.1 OSPF-内部网关路由协议

OSPF (Open Shortest Path First) 采用 **分布式的链路状态** 算法

OSPF 协议的基本思想

- 向本 **自治系统** 中所有路由器洪泛信息
- 发送的信息就是与本路由器相邻的所有路由器的链路状态
- 只有当 **链路状态** 发生变化时路由器才用洪泛法发送此信息

5.4.1.1 OSPF-链路状态

“链路状态”就是说明本路由器都和哪些路由器相邻，以及该链路的“度量”(metric)

- OSPF 度量值一般包括费用、距离、时延、带宽等

所有的路由器最终都能建立一个 **链路状态数据库 LSDB**

5.4.1.2 OSPF-区域的概念

OSPF 支持将一组网段组合在一起，称为一个区域

使用层次结构的区域划分，上层的区域叫做 **主干区域(backbone area)**，其他区域都必须与主干区域相连

非主干区域之间不允许直接发布区域间路由信息

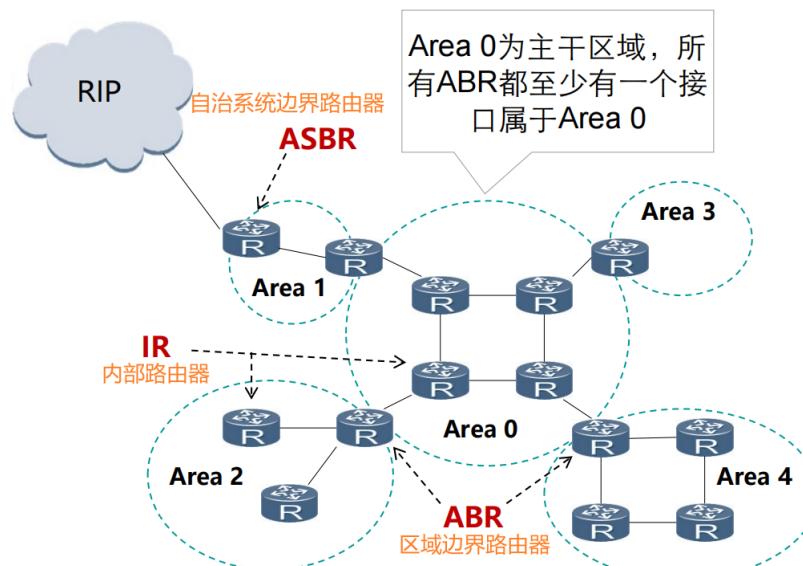


Figure 105

5.4.1.3 小结

➤ OSPF的特点

- 支持无类域间路由 (CIDR)
- 无路由自环
- 收敛速度快
- 使用IP组播收发协议数据
- 支持多条等值路由
- 支持协议报文的认证

➤ 请思考：

- 为什么OSPF协议适合大型网络？
- OSPF会不会存在“坏消息传播比较慢”的问题？不会

Figure 106

5.4.2 RIP-内部网关路由协议

路由选择协议 RIP (Routing Information Protocol) 是基于 [距离矢量算法](#) 的协议

使用 **跳数** 衡量到达目的网络的距离

- RIP 认为一个好的路由就是它通过的路由器的数目少，即“距离短”
- RIP 允许一条路径最多只能包含 15 个路由器

RIP 协议的基本思想

- 仅和相邻路由器交换信息（交换 DV）
- 路由器交换的内容是自己的路由表
- 周期性更新：30s

5.4.2.1 小结

➤ RIP协议的特点

- 算法简单，易于实现
- 收敛慢
- 需要交换的信息量较大

➤ RIP协议的防环路机制

- 触发更新
- 毒性反转
- 水平分割
- 其他

➤ RIP协议的适用场合

- 中小型网络 有最大跳数限制

Figure 107

5.4.3 BGP-外部网关路由协议

Note

路由协议：

内部网关协议 IGP：有 RIP 和、OSPF、ISIS 等多种具体的协议

外部网关协议 EGP：目前使用的协议就是 BGP

边界网关协议 BGP (Border Gateway Protocol)：目前互联网中唯一实际运行的自治域 AS 间的路由协议

BGP 功能

- eBGP: 从相邻的 AS 获得网络可达信息
- iBGP: 将网络可达信息传播给 AS 内的路由器
- 基于网络可达信息和策略决定到其他网络的“最优”路由

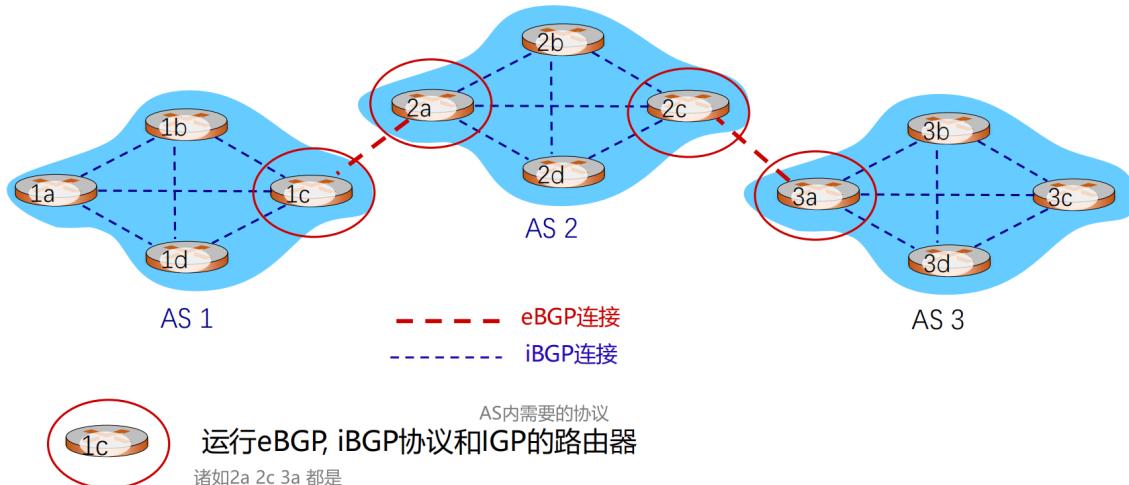


Figure 108

BGP 会话: 两个 BGP 路由器通过 TCP 连接交换 BGP 报文

- 通告到不同网络前缀的路径，即 **路径向量** 协议

BGP 路径通告: 完整路径通报给 AS

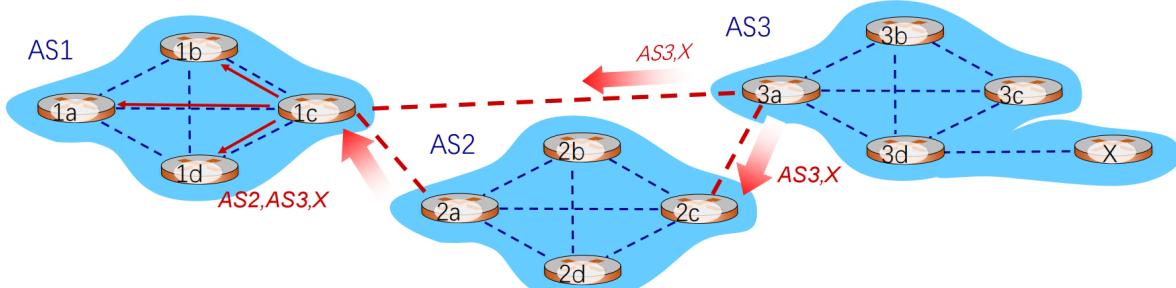


Figure 109

路由器可能会学到多条到目的网络的路径

由策略，AS1 路由器 1c 可能选择路径 AS3, X，并在 AS1 中通过 iBGP 通告路径

5.4.3.1 BGP 协议的特点

BGP 协议交换路由信息的结点数量级是自治系统数的量级

每一个自治系统边界路由器的数目是很少的

在 BGP 刚刚运行时，BGP 的邻站是交换整个的 BGP 路由表；以后只需要在发生变化时更新有变化的部分

5.4.3.2 BGP 报文

BGP 通过 TCP 的 179 端口交换报文

5.4.3.3 BGP 路由策略

路由器使用策略决定接受或拒绝接收到的路由通告

路由器也会基于策略决定是否向其他相邻 AS 通告路径信息

5.4.4 标签交换和 MPLS

MPLS (MultiProtocol Label Switching) 多协议标签交换

- 多协议表示在 MPLS 的上层可以采用多种协议，例如：IP, IPv6、IPX
- 标签是指每个分组被分配一个标签，路由器根据该标签对分组进行转发
- 交换是指标签的交换，MPLS 报文交换和转发是基于标签的

MPLS 设计初衷为了 提升查找速度

MPLS 主要有以下三个方面的应用

- 面向连接的服务质量管理
- 流量工程，平衡网络负载
- 虚拟专用网 VPN

标签交换路由器 LSR

- 支持 MPLS 的路由器
- 具备标签交换、路由选择两种功能

MPLS 域

- 所有相邻的支持 MPLS 技术的路由器构成的区域

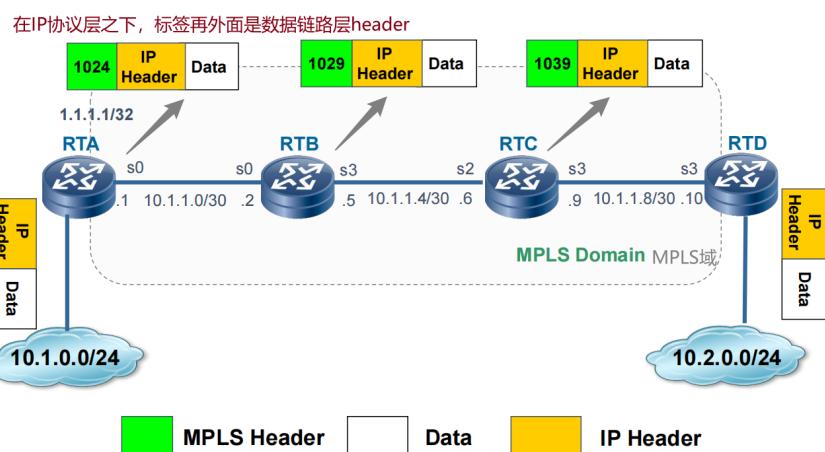
标签分配协议 LDP

- 用来在 LSR 之间建立 LDP 会话并交换 Label/FEC 映射信息

5.4.4.1 工作过程

➤ 加标签

- 在 **MPLS 域的入口处**，给每一个 IP 数据报加上 **标签**，然后对加上标记的 IP 数据报用 **硬件**进行转发



➤ 标签交换

- 采用 **硬件技术**对加上标记的 IP 数据报进行转发称为 **标签交换**

➤ 去标签

- 当分组离开 MPLS 域时，MPLS **出口路由器**把分组的**标签去除**。后续按照一般IP分组的转发方法进行转发

Figure 110

5.5 路由器工作原理

路由器是互联网最主要的网络设备，包含 2 个核心功能

- 控制层：运行各种路由协议：BGP、OSPF、RIP，学习去往不同目的的转发路径：路由表
- 数据层：根据上述路由表，将收到的 IP 分组转发到正确的下一跳链路

5.5.1 控制层

路由器可同时运行多个路由协议

路由器也可不运行任何路由协议，只使用静态路由和直连路由

路由管理根据路由优先级，选择最佳路由，形成核心路由表

控制层将核心路由表下发到数据层，形成转发表（FIB）

- 若存在多个“去往同一目的 IP 前缀”的不同类型路由，路由器根据优先级选择最佳路由
 - 优先级数值越小，优先级越高

路由种类	路由优先级
直连路由	0
静态路由	1
eBGP 路由	20
OSPF 路由	110
RIP 路由	120
iBGP 路由	200

◦

Figure 111

5.5.2 数据层

➤ 路由器中IP报文转发核心功能

- 链路层解封装，IP头部校验
- 获取报文目的IP地址
- 用目的IP地址，基于最长前缀匹配规则查询转发表
- 查询失败，丢弃报文
- 查询成功
 - 获取转发出接口和下一跳IP地址
 - IP头部“TTL”字段值减1，重新计算IP头部“校验和”
 - 重新进行链路层封装，发送报文

注：普通IP报文转发过程中，路由器不查看传输层及以上层协议的内容

➤ IP报文在路由器转发前后的变化

- 链路层封装更新，IP头部“TTL”减1，IP头部“校验和”更新

Figure 112

5.6 拥塞控制算法

💡 Tip

具体见下一章

5.6.1 拥塞控制概述

拥塞：网络中存在太多的数据包导致数据包传输延迟或丢失，从而导致网络吞吐量下降

拥塞控制（congestion control）：需要确保通信子网能够承载用户提交的通信量，是一个全局性问题，涉及主机、路由器等多种因素

产生拥塞的原因：

- 主机发送到网络的数据包数量过多，超过了网络的承载能力
- 突发的流量填满了路由器的缓冲区，造成某些数据包会被丢弃

拥塞可以在网络层解决，但实际上大多在传输层解决

5.6.2 流量感知路由

5.6.3 流量调节

显式拥塞通告（ECN，Explicit Congestion Notification）

5.6.4 随机早期检测

5.7 服务质量

💡 Tip

不是很重要

网络服务质量（QoS）

区分服务（DiffServ: Differentiated services）

5.8 三层交换与 VPN

5.8.1 三层交换

5.8.2 VPN 技术

虚拟专用网 VPN (Virtual Private Network)：利用公用网络架设专用网络的远程访问技术

- 专用网络的经济、可靠、灵活的解决方案
- 利用安全隧道技术将专用网络在公共网络上扩展

VPN 的设计原则

- 安全性、隧道与加密、数据验证、用户验证、防火墙与攻击检测

5.8.2.1 VPN 的原理

通过隧道技术在公共网络上模拟出一条点到点的逻辑专线，从而达到安全数据传输的目的

5.9 IPv6 技术

初始动机：应付“32-bit 地址空间耗尽”问题（CIDR 和 NAT 都无法从根本上解决地址短缺问题），增加地址空间

IPv6 地址

- 地址长度为128bit，是IPv4地址长度的4倍
- IPv6地址空间数量约为 3×10^{38}
- IPv6地址表示法，冒分十六进制， $x:x:x:x:x:x:x$
 - 简化方法：每个x前面的0可省略，可把连续的值为0的x表示为“::”，且“::”只能出现1次
 - 简化前地址，2001:0DA8:0000:0000:200C:0000:0000:00A5
 - 简化后地址，2001:**DA8**:0000:0000:200C:**A5**

Figure 113

5.9.1 IPV6 头部

- 版本：4bit，协议版本号，值为6
- 流量类型：8bit，区分数据包的服务类别或优先级
- 流标签：20bit，标识同一个数据流
- 有效载荷长度：16bit，IPv6报头之后载荷的字节数（含扩展头），最大值64K

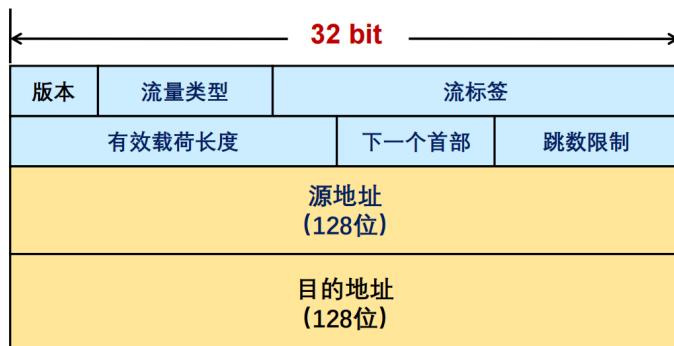


Figure 114

- 下一个首部：8bit，IPv6报头后的协议类型，可能是TCP/UDP/ICMP等，也可能是扩展头
- 跳数限制：8bit，类似IPv4的TTL，每次转发跳数减1，值为0时包将会被丢弃
- 源地址：128bit，标识该报文的源地址
- 目的地址：128bit，标识该报文的目的地址

Figure 115

IPv6 头部字段分析

- IPv6头部长度固定40字节，所有“选项”字段都在IPv6扩展头部分
- 与IPv4头部的比较
 - 去除“首部长度”（首部长度固定为40字节）
 - 去除“首部校验和”（传输层校验会涉及伪头部信息），提升转发速度
 - 去除分片字段：“标识”“标志”“片偏移”，移至扩展头（分段头）
- IPv6分片机制
 - IPv6分组不能在传输途中分片，只在源端进行分片
 - IPv6设计了专门的分片扩展头，分片字段不存在基本IPv6头部中
 - IPv6支持Path MTU发现机制

首部格式改变提升转发处理速度

Figure 116

5.9.2 IPv6 扩展头

- IPv6报文可承载多个扩展头
- 每个扩展头都包含“下一个首部”字段（IPv6首部固定字段也有）
 - 可指向下一个扩展头类型
 - 或指明传统上层协议类型（最后一个扩展头）：TCP/UDP/ICMP ...
- 如有多个扩展头，需按规定顺序出现
 - 逐跳选项头，转发路径上每个节点都需检查该扩展头的信息
 - 路由头，指明转发途中需经过哪些节点，类似于IPv4的源路由机制
 - 分段头，包含类似IPv4分片处理信息：片偏移、“更多段”标志、标识符
 - 目的地选项头，目的端系统需要确认的信息
 - ...

Figure 117

5.9.3 邻居发现

不展开

5.9.4 IPv6 地址及配置

- IPv6地址分类
 - 未指定地址（::/128），不能分配给任何节点
 - 回环地址（::1/128^{1个地址}），表示节点自己，不分配，类似IPv4中的127.0.0.1
 - 组播地址（FF00::/8）
 - 链路本地地址（FE80::/10），也称为Link-local地址，仅在本地链路上使用，网络设备根据接口MAC地址自动生成
 - 全局单播地址，其它地址
- IPv6地址配置方式
 - 手动配置
 - DHCPv6（IPv6动态主机配置协议）
 - 无状态地址自动配置，基于ND协议的RS报文的IPv6前缀信息，结合自己的链路层地址生成IPv6地址

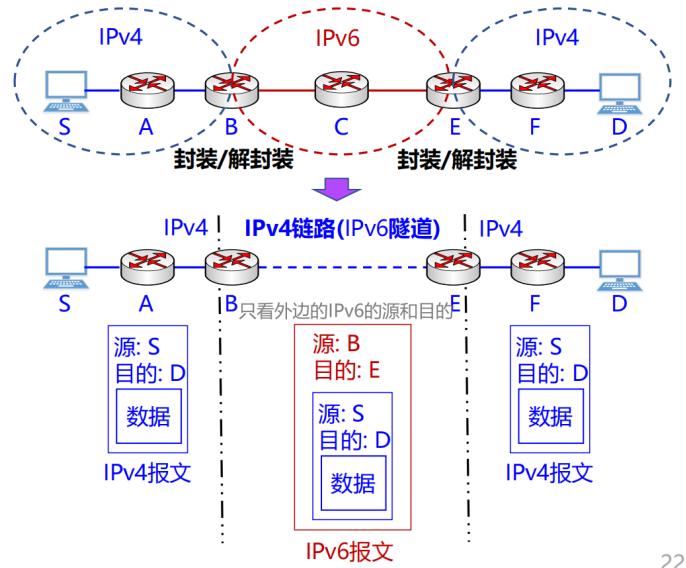
Figure 118

5.9.5 IPv4 到 IPv6 迁移及过渡技术

主流过渡技术：双栈技术、翻译技术、隧道技术

5.9.5.1 隧道技术

隧道技术（2个同构网络的系统，跨越中间异构网络进行通信）：将 A 协议数据包封装在 B 协议中传输



221

Figure 119

5.9.5.2 翻译技术

破坏互联网端到端原则

6. 传输层

6.1 概述和传输层服务

6.1.1 传输层的位置

传输层位于应用层和网络层之间

“端到端”设计原则：应用程序只运行在终端上，进程 -- 进程

传输层应提供进程之间本地通信的抽象

6.1.2 不同终端上的进程如何通信？

应用程序和网络之间存在一扇“门”：这扇“门”称为 **套接字 (socket)**，是应用层和传输层的接口，也是应用程序和网络之间的 API

6.1.3 传输层提供什么服务？

网络层 提供“尽力而为”的服务 Best effort

- 不保证交付，不保证按序交付，不保证数据完整，不保证延迟，不保证带宽等

传输层的 **有所为、有所不为**

- 传输层可以通过差错恢复、重排序等手段提供可靠、按序的交付服务
- 但传输层无法提供延迟保证、带宽保证等服务

- 最低限度的传输服务：
 - 将终端-终端的数据交付扩展到进程-进程的数据交付 (6.3节)
 - 报文检错
- 增强服务：
 - 可靠数据传输
 - 流量控制
 - 拥塞控制
- 因特网传输层通过UDP协议和TCP协议，向应用层提供两种不同的传输服务：
 - UDP协议 (6.4节)：仅提供最低限度的传输服务
 - TCP协议 (6.5节, 6.7节)：提供基础服务和增强服务

Figure 120

6.2 套接字编程

6.2.1 网络应用

应用程序体系结构：客户-服务器体系结构 (C/S) client-server

客户只与服务器通信，客户之间不通信

6.2.2 进程如何标识自己

进程标识包括：

- 主机地址
- 主机上与该进程关联的端口号 port number

6.2.3 应用编程接口：socket API

应用需 **显式地** 创建、使用和释放套接字

采用客户-服务器模式：客户总是主动发起通信的一方，服务器始终在等待客户的服务请求到来

应用可以通过 socket API 调用两种传输服务：

- 不可靠的数据报服务：由 UDP 协议实现
- 可靠的字节流服务：由 TCP 协议实现

6.2.4 创建套接字：socket()

- 客户或服务器调用socket()创建本地套接字，返回套接字描述符
- domain指明网络层地址类型
- type指明传输层协议：
 - SOCK_STREAM代表TCP字节流
 - SOCK_DGRAM代表UDP数据报

```
#include <sys/socket.h>

int socket(
    int domain, /* AF_UNIX, AF_INET, etc. */
    int type,   /* SOCK_STREAM, SOCK_DGRAM */
    int protocol); /* usually zero */
/* Returns file descriptor or -1 on error (sets errno) */
```

Figure 121

6.2.5 套接字描述符

- Linux系统中程序通过访问套接字描述符来进行套接字通信
- 套接字描述符在类型上与文件描述符完全兼容
- 在套接字描述符上的**发送、接收数据**的行为可以用**文件读写**调用完成
 - read从描述符fd中读取(接收)长度为返回值的字节数至buf中
 - write向描述符fd中写入(发送)buf中长度为返回值的字节数

```
#include <unistd.h>

int read(int fd, void *buf, size_t count);
int write(int fd, void *buf, size_t count);
```

Figure 122

6.2.6 使用 UDP 套接字实现回音服务

客户或服务器调用 sendto()发送数据

客户或服务器调用 recvfrom()接收数据

6.2.6.1 通信流程

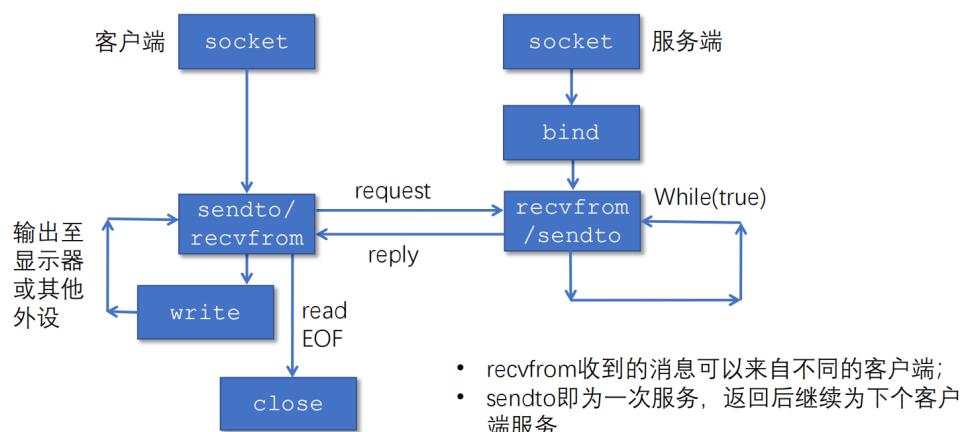


Figure 123

```
1 // UDP服务端代码
2 int main(int argc, char **argv){
3     char mesg[MAXLINE];
4     int sockfd, n, len;
5     struct sockaddr_in cliaddr, servaddr;
6
7     sockfd = socket(AF_INET, SOCK_DGRAM, 0);
8     bzero(&servaddr, sizeof(servaddr));
9     servaddr.sin_family = AF_INET;
10    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
11    servaddr.sin_port = htons(SERV_PORT);
12
13    bind(sockfd, (struct sockaddr *) &servaddr, sizeof(servaddr));
14
15    for ( ; ; ) {
16        len = sizeof(cliaddr);
17        n = recvfrom(sockfd, mesg, MAXLINE, 0, &cliaddr, &len);
18        sendto(sockfd, mesg, n, 0, cliaddr, len);
```

```

19     }
20 }
21
22 // UDP 回音 客户端代码
23 int main(int argc, char **argv){
24     int sockfd, n;
25     struct sockaddr_in servaddr;
26     char sendline[MAXLINE], recvline[MAXLINE + 1];
27     if (argc != 2){
28         fprintf(stderr, "%s\n", "usage: echoCli <IPaddress>");
29         exit(EXIT_FAILURE);
30     }
31     sockfd = socket(AF_INET, SOCK_DGRAM, 0);
32     bzero(&servaddr, sizeof(servaddr));
33     servaddr.sin_family = AF_INET;
34     inet_pton(AF_INET, argv[1], &servaddr.sin_addr);
35     servaddr.sin_port = htons(SERV_PORT);
36     while (fgets(sendline, MAXLINE, stdin) != NULL) {
37         sendto(sockfd, sendline, strlen(sendline), 0,
38                 (struct sockaddr *) &servaddr, sizeof(servaddr));
39         n = recvfrom(sockfd, recvline, MAXLINE, 0, NULL, NULL);
40         recvline[n] = 0;
41         fputs(recvline, stdout);
42     }
43     exit(0);
44 }
```

Fence 1

6.2.7 使用 TCP 套接字实现回音服务

TCP 服务器监听套接字：listen() -- 服务端

TCP 套接字系统调用：accept() -- 服务端

TCP 套接字系统调用：connect() -- 客户端

服务端在 SERV_PORT (9999) 监听

6.2.7.1 服务器使用多个套接字服务客户

1. 服务器进程在 **监听套接字** 上等待客户的连接请求
2. 客户进程创建一个 **本地套接字**, 与服务器的监听套接字通信: 在此过程中, 客户 TCP 向服务器 TCP 发送连接请求
3. 收到连接请求后, 服务器进程创建一个 **临时套接字 (连接套接字)** 和一个新的服务器进程, 与客户进程通信
4. 服务器进程回到监听套接字上继续等待: 此举允许服务器同时服务多个客户
5. 客户服务结束后, 服务器销毁进程, 关闭连接套接字

6.2.7.2 基于 TCP 的套接字通信流程

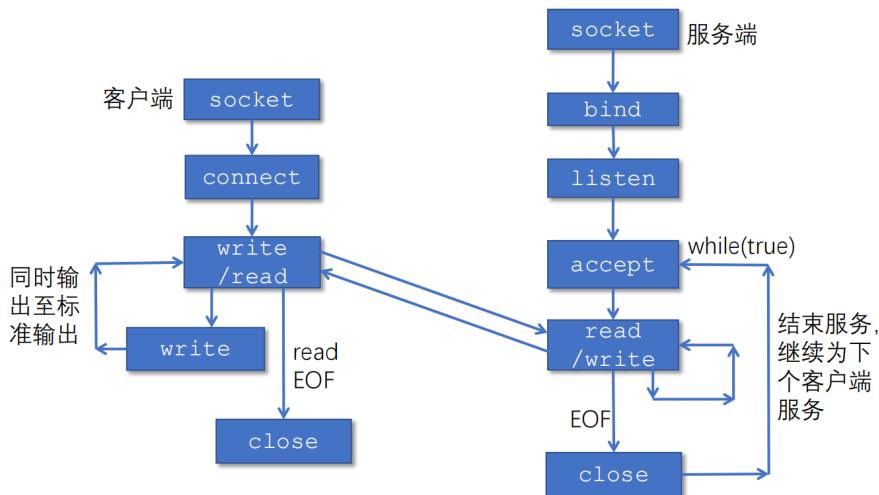


Figure 124

```
1 // TCP回音服务端
2 int main(int argc, char **argv){
3     int listenfd, connfd;
4     socklen_t clilen;
5     struct sockaddr_in cliaddr, servaddr;
6
7     listenfd = socket(AF_INET, SOCK_STREAM, 0);
8     bzero(&servaddr, sizeof(servaddr));
9     servaddr.sin_family = AF_INET;
10    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
11    servaddr.sin_port = htons(SERV_PORT);
12    bind(listenfd, (struct sockaddr *) &servaddr, sizeof(servaddr));
13    listen(listenfd, LISTENQ);
14
15    for ( ; ; ) {
16        clilen = sizeof(cliaddr);
17        connfd = accept(listenfd, (struct sockaddr *) &cliaddr, &clilen);
18        str_echo(connfd); /* process the request */
19    }
20}
21
22}
```

Fence 2

6.2.8 套接字标识与端口号

端口号是套接字标识的一部分：

- 每个套接字在本地关联一个端口号
- 端口号是一个 16 比特的数

👉 端口号的分类：

- 熟知端口：0 ~ 1023，由公共域协议使用
- 注册端口：1024 ~ 49151，需要向 IANA 注册才能使用
- 动态和/或私有端口：49152 ~ 65535，一般程序使用

👉 报文段中有两个字段携带端口号

- 源端口号：与发送进程关联的本地端口号
- 目的端口号：与接收进程关联的本地端口号

6.2.8.1 套接字端口号的分配

自动分配（客户端）：

- 创建套接字时不指定端口号
- 由操作系统从 49152 ~ 65535 中分配

使用指定端口号创建套接字（服务器）：

- 创建套接字时指定端口号
- 实现公共域协议的服务器应分配众所周知的端口号（0 ~ 1023）

6.3 传输层复用和分用

传输层基本服务：将主机间交付扩展到进程间交付，通过 **复用** 和 **分用** 实现

- （发送端）复用：传输层从多个套接字收集数据，交给网络层发送
- （接收方）分用：传输层将从网络层收到的数据，交付给正确的套接字

6.3.1 UDP 套接字

UDP 套接字使用 `<IP地址, 端口号>` 二元组进行标识

接收方传输层收到一个 UDP 报文段后：

- 检查报文段中的目的端口号，将 UDP 报文段交付到具有该端口号的套接字
- `<目的IP地址, 目的端口号>` 相同的 UDP 报文段被交付给同一个套接字，与 `<源IP地址, 源端口号>` 无关
- 报文段中的 `<源IP地址, 源端口号>` 被接收进程用来发送响应报文

6.3.2 TCP 套接字

一个 TCP 服务器为了同时服务很多个客户，使用两种套接字

监听套接字：

- 服务器平时在监听套接字上等待客户的连接请求，该套接字具有众所周知的端口号

连接套接字：

- 服务器在收到客户的连接请求后，创建一个连接套接字，使用临时分配的端口号
- 服务器同时创建一个新的进程，在该连接套接字上服务该客户
- **每个连接套接字只与一个客户通信**，即只接收具有以下四元组的报文段：
 - 源 IP 地址 = 客户 IP 地址，源端口号 = 客户套接字端口号
 - 目的 IP 地址 = 服务器 IP 地址，目的端口号 = 服务器监听套接字的端口号

连接套接字需要使用 `<源IP地址, 目的IP地址, 源端口号, 目的端口号>` **四元组** 进行标识，服务器使用该四元组将 TCP 报文段交付到正确的连接套接字

6.4 无连接传输：UDP

UDP 提供的服务：

- 进程到进程之间的报文交付
- 报文完整性检查（可选）：检测并丢弃出错的报文

💡 UDP 需要实现的功能：

- 复用和分用
- 报文检错

6.4.1 UDP 报文段结构

➤ UDP报文：

- 报头：携带协议处理需要的信息
- 载荷（payload）：携带上层数据

➤ 用于复用和分用的字段：

- 源端口号
- 目的端口号

➤ 用于检测报文错误的字段：

- 报文总长度
- 校验和（checksum）

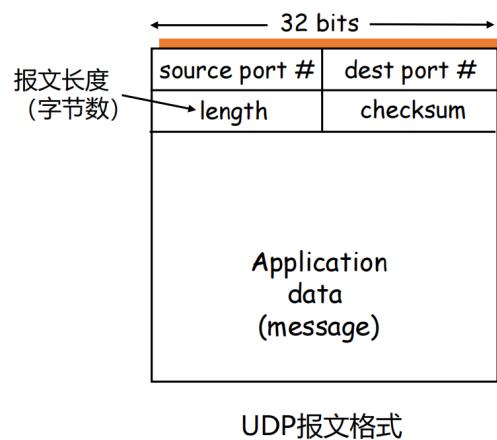


Figure 125

6.4.2 校验和计算

➤ 计算校验和时，checksum字段填0

➤ 接收方对UDP报文（包括校验和）及伪头求和，若结果为0xFFFF，认为没有错误

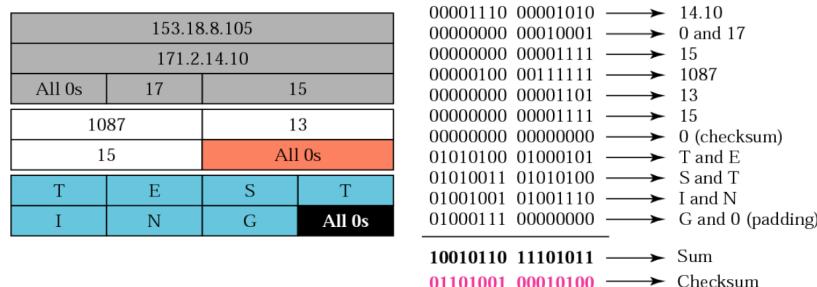


Figure 126

6.4.3 为什么需要 UDP？

1. 为什么需要 UDP？

应用可以尽可能快地发送报文：

- 无建立连接的延迟
- 不限制发送速率（不进行拥塞控制和流量控制）
- 报头开销小

- 协议处理简单

2. UDP 适合哪些应用?

- 容忍丢包但对延迟敏感的应用: 如流媒体
- 以单次请求/响应为主的应用: 如 DNS
- 若应用要求基于 UDP 进行可靠传输: 由应用层实现可靠性

6.5 面向连接的传输: TCP

6.5.1 TCP 概述

TCP 服务模型: 在一对通信的进程之间提供一条理想的 **字节流管道**

点到点通信: 仅涉及一对通信进程

全双工: 可以同时双向传输数据

可靠、有序的字节流: **不保留报文边界**

需要的机制:

1. 建立连接: 通信双方为本次通信建立数据传输所需的状态 (套接字、缓存、变量等)
2. 可靠数据传输: 流水线式发送, 报文段检错, 丢失重传
3. 流量控制: 发送方不会令接收方缓存溢出

6.5.2 TCP 报文段结构

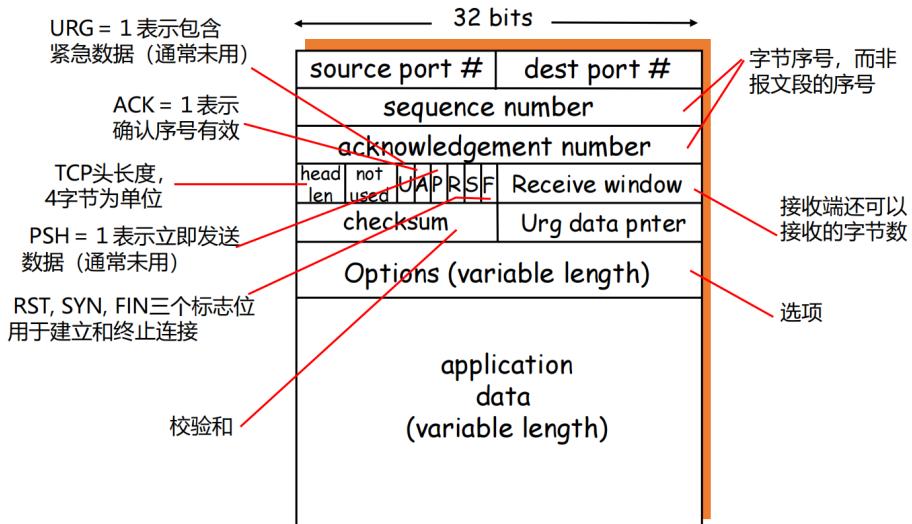


Figure 127

发送序号 seq number: 数据载荷中第一个字节在字节流中的序号

确认序号 ack number: 期望接收的下一个字节的序号

最大段长度 (MSS) : TCP 段中可以携带的最大数据字节数; 建立连接时, 每个主机可声明自己能够接受的 MSS, 缺省为 536 字节

- 主机A向主机B发送仅包含一个字符‘C’的报文段：
 - 发送序号为42
 - 确认序号为79（对前一次数据的确认）
- 主机B将字符‘C’回送给主机A：
 - 发送序号为79
 - 确认序号为43（对收到‘C’的确认）
- 主机A向主机B发送确认报文段（不包含数据）：
 - 确认序号为80（对收到‘C’的确认）

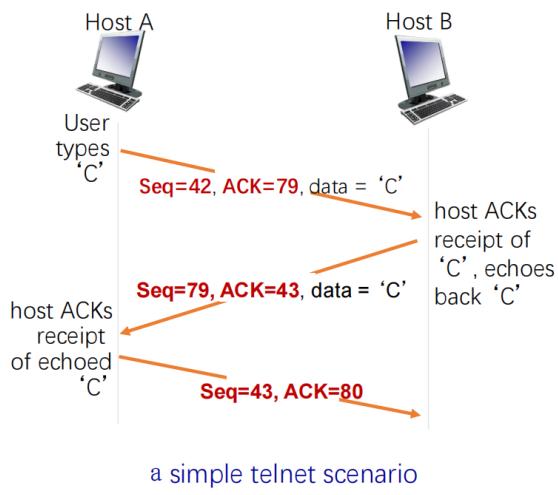


Figure 128

6.5.3 TCP 可靠数据传输

TCP 在不可靠的 IP 服务上建立可靠的数据传输

💡 基本机制

- 发送端：流水线式发送数据、等待确认、超时重传
- 接收端：进行差错检测，采用累积确认机制

💡 乱序段处理：协议没有明确规定

- 接收端不缓存：可以正常工作，处理简单，但效率低
- 接收端缓存：效率高，但处理复杂

6.5.3.1 高度简化的 TCP 协议

仅考虑可靠传输机制，且数据仅在一个方向上传输

➤ 接收方：

- 确认方式：采用累积确认，仅在正确、按序收到报文段后，更新确认序号；其余情况，重复前一次的确认序号（与GBN类似）
- 失序报文段处理：缓存失序的报文段（与SR类似）

➤ 发送方：

- 发送策略：流水线式发送报文段
- 定时器的使用：仅对最早未确认的报文段使用一个重传定时器（与GBN类似）
- 重发策略：仅在超时后重发最早未确认的报文段（与SR类似，因为接收端缓存了失序的报文段）

Figure 129

6.5.3.2 TCP 发送方要处理的事件

💡 收到应用数据：

- 创建并发送 TCP 报文段
- 若当前没有定时器在运行（没有已发送、未确认的报文段），启动定时器

💡 超时：

- 重传包含最小序号的、未确认的报文段
- 重启定时器

👉 收到 ACK:

- 如果确认序号大于基序号（已发送未确认的最小序号）：
- 推进发送窗口（更新基序号）
- 如果发送窗口中还有未确认的报文段，启动定时器，否则终止定时器

6.5.3.3 如何设置超时值

SampleRTT: 瞬时 从发出某个报文段到收到其确认报文段之间经过的时间

EstimatedRTT: 平均值

平均 RTT 的估算方法（指数加权移动平均）：

- EstimatedRTT = $(1 - \alpha) \times \text{EstimatedRTT} + \alpha \times \text{SampleRTT}$, 典型地, $\alpha = 0.125$!
- 如果 α 比较大，则是

6.5.3.4 TCP 确认的二义性

重传的 TCP 报文段使用与原报文段相同的序号

发送端收到确认后，无法得知是对哪个报文段进行的确认

解决方法：定时器补偿

- 发送方每重传一个报文段，就直接将超时值增大一倍（不依赖于 RTT 的更新）
- 若连续发生超时事件，超时值呈指数增长（至一个设定的上限值）

6.5.3.5 小结

- | | |
|---|--|
| <p>➤ TCP 可靠传输的设计要点：</p> <ul style="list-style-type: none"> • 流水式发送报文段 • 缓存失序的报文段 • 采用累积确认 • 只对最早未确认的报文段使用一个重传定时器 • 超时后只重传包含最小序号的、未确认的报文段 | <p>➤ 超时值的确定：</p> <ul style="list-style-type: none"> • 基于 RTT 估计超时值 + 定时器补偿策略 <p>➤ 测量 RTT：</p> <ul style="list-style-type: none"> • 不对重传的报文段测量 RTT • 不连续使用推迟确认 <p>➤ 快速重传：</p> <ul style="list-style-type: none"> • 收到 3 次重复确认，重发报文段 |
|---|--|
- 以上措施可大量减少因 ACK 丢失、定时器过早超时引起的重传

Figure 130

6.5.3.6 TCP 使用 GBN 还是 SR

TCP 结合了 GBN 和 SR 的优点

TCP 在减小定时器开销和重传开销方面要优于 GBN 和 SR！

6.5.3.7 Crash Recovery

6.5.4 TCP 流量控制

TCP 接收端有一个接收缓存：

① Note

本质：发送端 TCP 通过调节发送速率，不使接收端缓存溢出

6.5.4.1 TCP 如何进行流量控制

接收缓存中的可用空间称为 **接收窗口**



6.5.5 TCP 连接管理

6.5.5.1 TCP 两次次握手建立连接

建立一条 TCP 连接需要确定两件事：

- 双方都同意建立连接（知晓另一方想建立连接）
- 初始化连接参数（序号，MSS 等）

① Caution

在网络中，2 次握手总是可行的吗？

在一个不可靠的网络中，总会有一些意外发生：

- 包传输延迟变化很大
- 存在重传的报文段
- 存在报文重排序

==> delayed duplicates problem

Delayed Duplicates Problem

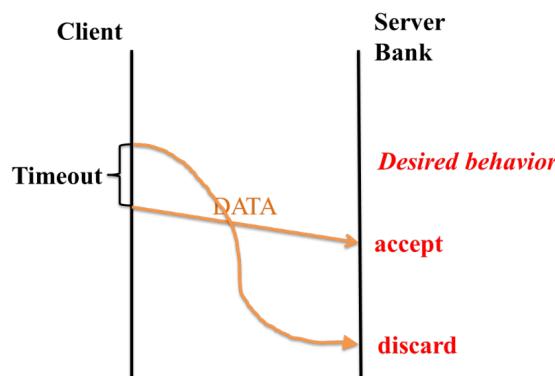


Figure 131

Solution 1

Problems

There are two problems which complicates the scenario:

- Possible wrap around of seqno

- Client/Host or Server may crash
Assume seqno = 0,1,...,7 (i.e. 3 bits)

Problem: How to differentiate **delay duplicate** and **new pkt with wrapped around seqno** ?

Idea: use time

Two assumptions that simplifies the problem

- Assumption 1. Time for seqno wrap around (**T1**) is typically large if, e.g., seqno is 32 bits long.
- Assumption 2. If pkt delays a relatively short time (**T2 < T1**), we can use time to differentiate the two.

Solution 2

6.5.5.2 How to realize the assumptions?

1. Restrict packet lifetime

Packet lifetime can be restricted to a known maximum using one of the following techniques

- Restricted subnet design.
- Putting a hop counter in each packet.
- Timestamping each packet (router synchronization required)

$$T = n * (\text{pkt lifetime})$$

- It is impossible to receive a delay duplicate after **T**

2. use time-of-day clock

- Time-of-day clock at hosts

Each clock is assumed to take the form of a binary counter that increments itself at uniform intervals. 时钟不断增加

The number of bits in the counter must equal or exceed the number of bits in the sequence numbers.

The clock is assumed to continue running even if the host goes down.

The clocks at different hosts need **not** be synchronized.

- initial seqno of a connection = low k bits of time-of-day clock

6.5.5.3 TCP 起始序号的选择

基于时钟的起始序号选取算法：

- 每个主机使用一个时钟，以二进制计数器的形式工作，每隔 ΔT 时间计数器加 1
- 新建一个连接时，以本地计数器值的最低 32 位作为起始序号
- 该方法确保连接的起始序号随时间单调增长
 ΔT 取较小的值（4 微秒）：确保发送序号的增长速度，不会超过起始序号的增长速度

使用较长的字节序号（32 位）：确保序号回绕的时间远大于分组在网络中的最长寿命

6.5.5.4 Forbidden region of seqno

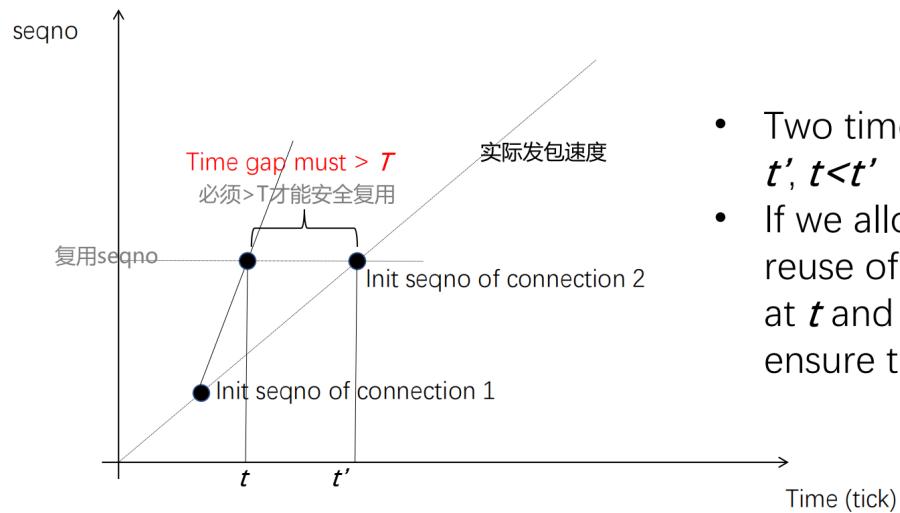


Figure 132

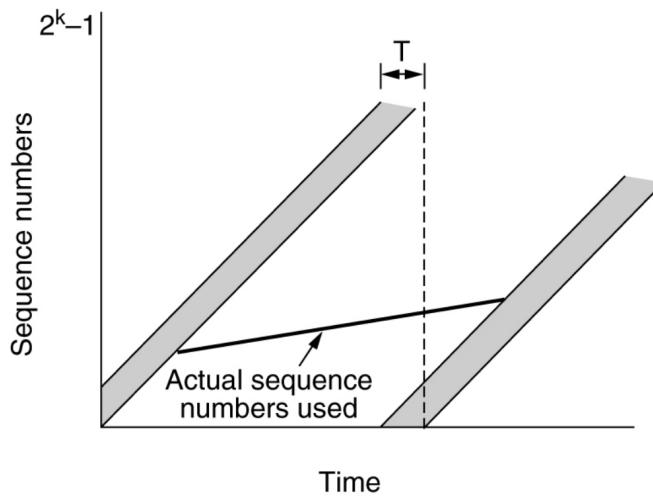


Figure 133

The increments of seqno cannot be **too fast or too slow**

Solution for the delayed duplicates:

1. the maximum data rate on any connection is one segment per clock tick. (i.e., cannot be too fast)
2. limits how slowly sequence numbers can advance on a connection (or how long the connections may last). (i.e., cannot be too slow)

6.5.5.5 TCP 三次握手建立连接

1. 客户TCP发送SYN报文段 (SYN=1, ACK=0)
 - 给出客户选择的起始序号
 - 不包含数据
2. 服务器TCP发送SYNACK报文段 (SYN=ACK=1) (服务器端分配缓存和变量)
 - 给出服务器选择的起始序号
 - 确认客户的起始序号
 - 不包含数据
3. 客户发送ACK报文段 (SYN=0, ACK=1) (客户端分配缓存和变量)
 - 确认服务器的起始序号
 - 可能包含数据

Figure 134

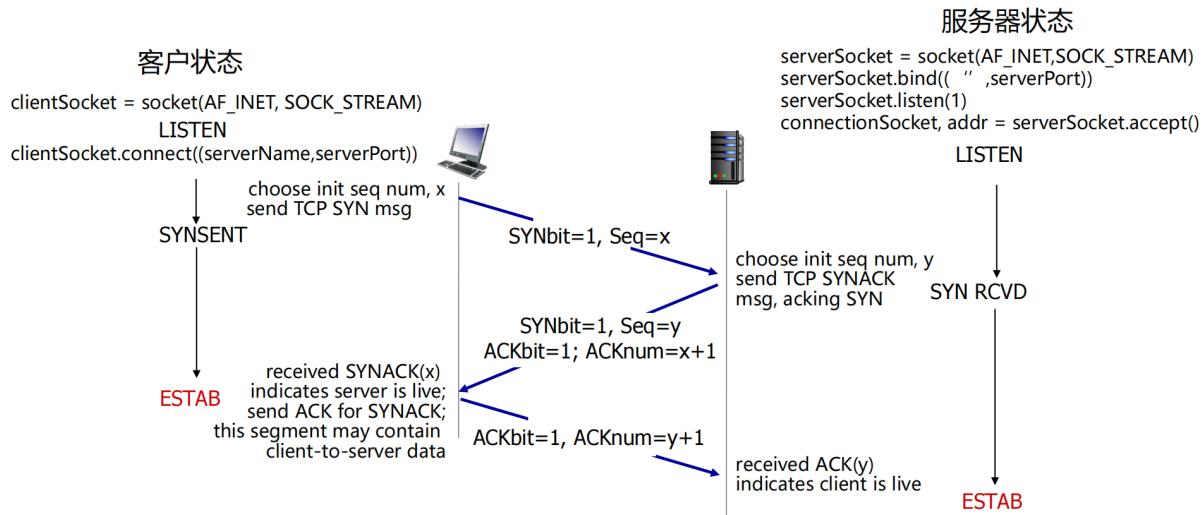


Figure 135

6.5.5.6 关闭 TCP 连接

Asymmetric release 非对称

- When one part hangs up, the connection is broken.
- Asymmetric release is abrupt and may result in data loss

Symmetric release 对称

- to treat the connection as two separate unidirectional connections and require each one to be released separately.

SYMMETRIC

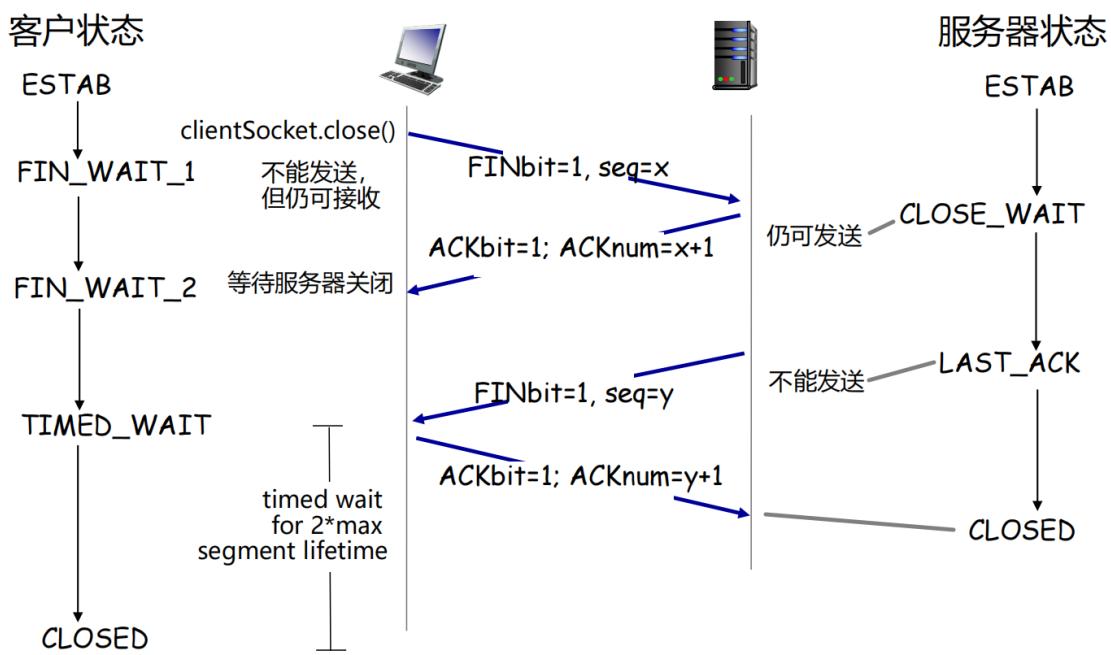


Figure 136

Two army-problem: A white army is encamped in a valley. On both of the surrounding hillsides are blue armies.

6.5.5.7 客户/服务器经历的 TCP 状态序列

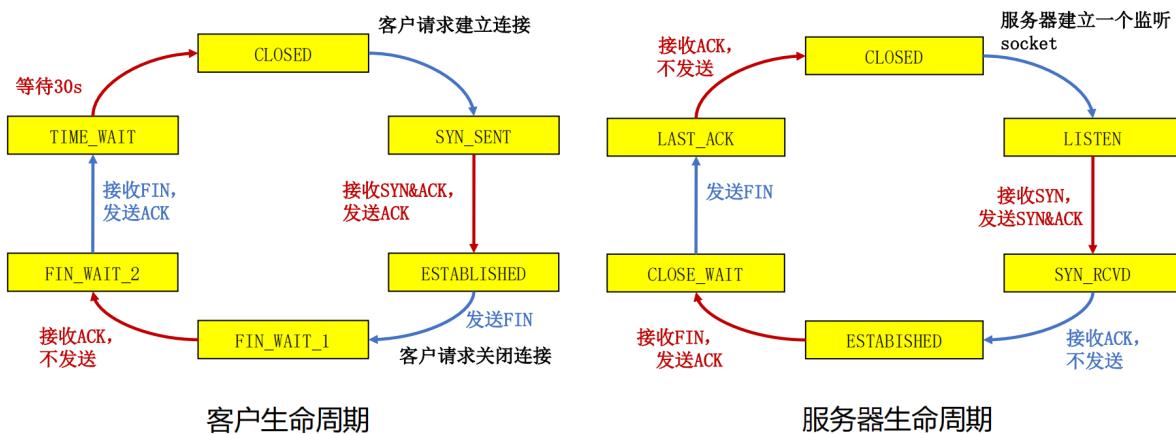


Figure 137

6.5.5.8 SYN 洪泛攻击

攻击者采用伪造的源 IP 地址，向服务器发送大量的 SYN 段，却不发送 ACK 段

服务器为维护一个巨大的半连接表耗尽资源，导致无法处理正常客户的连接请求，表现为服务器停止服务

6.5.5.9 TCP 端口扫描

TCP 端口扫描的原理：

- 扫描程序依次与目标机器的各个端口建立 TCP 连接
- 根据获得的响应来收集目标机器信息

在典型的 TCP 端口扫描过程中，发送端向目标端口发送 SYN 报文段：

- 若收到 SYNACK 段，表明目标端口上有服务在运行
- 若收到 RST 段，表明目标端口上没有服务在运行
- 若什么也没收到，表明路径上有防火墙，有些防火墙会丢弃来自外网的 SYN 报文段

6.6 理解网络拥塞

6.6.1 网络拥塞的后果

网络拥塞造成：

- 丢包：由路由器缓存溢出造成
- 分组延迟增大：链路接近满载造成
大量网络资源用于：
 - 重传丢失的分组
 - (不必要地) 重传延迟过大的分组
 - 转发最终被丢弃的分组

结果：进入网络的负载很重，网络吞吐量却很低

6.6.2 拥塞控制的常用方法

端到端拥塞控制

- 网络层不向端系统提供反馈
- 端系统通过观察丢包和延迟，自行推断拥塞的发生
- TCP 采用此类方法

6.7 TCP 拥塞控制

6.7.1 拥塞检测和速率限制

发送方如何感知拥塞？

发送方利用丢包事件感知拥塞： - 拥塞造成丢包和分组延迟增大

- 无论是丢包还是分组延迟过大，对于发送端来说都是丢包了 丢包事件包括：
 - 重传定时器超时
 - 发送端收到 3 个重复的 ACK

发送方采用什么机制限制发送速率？

发送方使用 **拥塞窗口 cwnd** 限制已发送未确认的数据量： $\text{LastByteSent} - \text{LastByteAcked} \leq \text{cwnd}$

$$\text{rate} = \frac{\text{cwnd}}{\text{RTT}} \text{Bytes/sec}$$

cwnd 随发送方感知的网络拥塞程度而变化

6.7.2 拥塞窗口的调节策略：AIMD

Multiplicative Decrease & Additive Increase

➤ 乘性减 (Multiplicative Decrease)

- 发送方检测到丢包后，将 cwnd 的大小减半（但不能小于一个 MSS）
- 目的：迅速减小发送速率，缓解拥塞

➤ 加性增 (Additive Increase)

- 若无丢包，每经过一个 RTT，将 cwnd 增大一个 MSS，直到检测到丢包
- 目的：缓慢增大发送速率，避免振荡

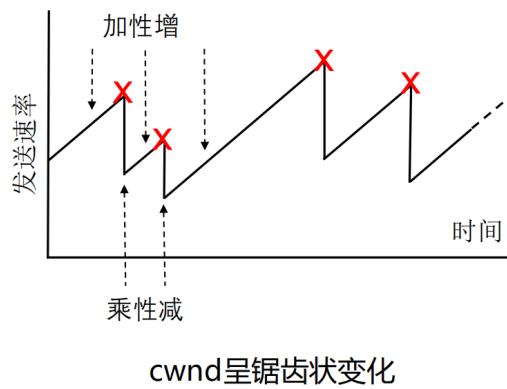


Figure 138

6.7.3 TCP 慢启动

在新建的连接（或沉寂了一段时间的连接）上，以什么速率发送数据（此时接收窗口达最大值）？

早期的 TCP 协议：

- 发送端仅以接收窗口大小限制发送速率，网络经常因为拥塞而崩溃！

采用“加性增”增大发送窗口，太慢！

- 在新建连接上，令 $cwnd = 1 \text{ MSS}$ ，起始速度 = MSS/RTT
- 然而，网络中的可用带宽可能远大于 MSS/RTT

慢启动的基本思想：

- 在新建连接上指数增大 cwnd，直至检测到丢包（此时终止慢启动）
- 希望迅速增大 cwnd 至可用的发送速度

慢启动的策略：每经过一个 RTT，将 cwnd 加倍

慢启动的具体实施：

- 每收到一个 ACK 段，cwnd 增加一个 MSS
- 只要发送窗口允许，发送端可以立即发送下一个报文段

特点：以一个很低的速率开始，按指数增大发送速率

慢启动比谁“慢”？与早期 TCP 按接收窗口发送数据的策略相比，采用慢启动后发送速率的增长较慢

6.7.4 区分不同的丢包事件

- 超时和收到3个重复的ACK，它们反映出来的网络拥塞程度是一样的吗？**当然不一样！**
 - **超时：**说明网络交付能力很差
 - **收到3个重复的ACK：**说明网络仍有一定的交付能力
 - 目前的TCP实现区分上述两种不同的丢包事件
- 收到3个重复的ACK：
• 将cwnd降至一半
• 使用AIMD调节cwnd

➤ 超时：
• 设置门限 = cwnd/2
• cwnd=1MSS
• 使用慢启动增大cwnd至门限
• 使用AIMD调节cwnd

Figure 139

6.7.5 TCP 拥塞控制的实现

发送方维护变量 ssthresh
发生丢包时， $ssthresh = cwnd/2$

ssthresh 是从慢启动转为拥塞避免的分水岭：

- cwnd 低于门限时，执行慢启动
- cwnd 高于门限：执行拥塞避免

拥塞避免阶段，拥塞窗口线性增长：每当收到 ACK， $cwnd = cwnd + MSS*(MSS/cwnd)$

检测到 3 个重复的 ACK 后：

- TCP Reno 实现： $cwnd = ssthresh+3$ ，线性增长
- TCP Tahoe 实现： $cwnd = 1 \times MSS$ ，慢启动

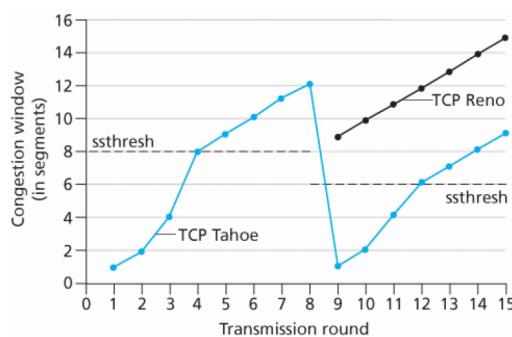


Figure 140

6.7.6 TCP 发送端的事件与动作

State	Event	TCP Sender Action	Commentary
慢启动 (SS)	收到新的确认	$cwnd = cwnd + MSS$, If ($cwnd > ssthresh$) set state to "Congestion Avoidance"	每经过一个RTT, $cwnd$ 加倍
拥塞避免 (CA)	收到新的确认	$cwnd = cwnd + MSS * (MSS/cwnd)$	每经过一个RTT, $cwnd$ 增加一个MSS
SS or CA	收到3个重复的确认	$ssthresh = cwnd/2$, $cwnd = Threshold+3$, Set state to "Congestion Avoidance"	$cwnd$ 减半, 然后线性增长
SS or CA	超时	$ssthresh = cwnd/2$, $cwnd = 1 MSS$, Set state to "Slow Start"	$cwnd$ 降为一个MSS, 进入慢启动
SS or CA	收到一个重复的确认	统计收到的重复确认数	$cwnd$ 和 $ssthresh$ 都不变

Figure 141

6.7.7 Wireless Issues

linker layer 重传 速度快于 TCP 的反应

6.7.8 TCP 连接的吞吐量

一个长期存活的TCP连接，平均吞吐量是多少？

➤忽略慢启动阶段（该阶段时间很短），只考虑
拥塞避免阶段

➤令 W =发生丢包时的拥塞窗口，此时有：

$$\text{throughput} = W/\text{RTT}$$

➤发生丢包后调整 $cwnd=W/2$ ，此时有：

$$\text{throughput}=W/2\text{RTT}$$

➤假设在TCP连接的生命期内，RTT 和 W 几乎不变，有：

$$\text{Average throughout} = 0.75 W/\text{RTT}$$

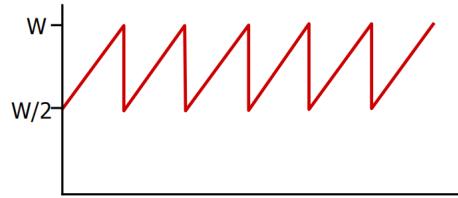


Figure 142

6.7.9 TCP 的公平性

公平性目标：如果 K 条 TCP 连接共享某条带宽为 R 的瓶颈链路，每条连接应具有平均速度 R/K

➤考虑两条竞争的连接（各种参数相同）共享带宽为R的链路：

- 加性增：连接1和连接2按照相同的速率增大各自的拥塞窗口，得到斜率为1的直线
- 乘性减：连接1和连接2将各自的拥塞窗口减半

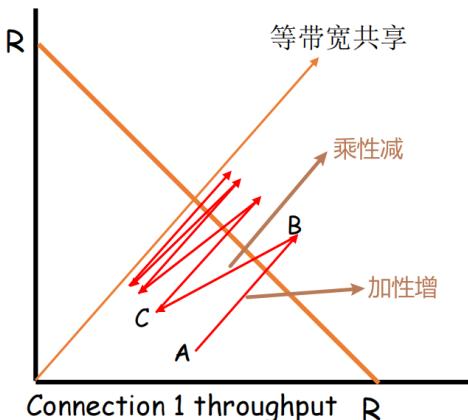


Figure 143

6.8 拥塞控制的发展

6.8.1 TCP CUBIC

拥塞控制算法

6.8.1.1 经典 TCP 拥塞控制的性能问题

TCP Reno 线性增大拥塞窗口，探测当前可用网络带宽，即每经过一个 RTT，拥塞窗口增加一个 MSS；当端到端时延带宽乘积 (BDP) 较大时，拥塞窗口增长过慢，导致信道无法满载

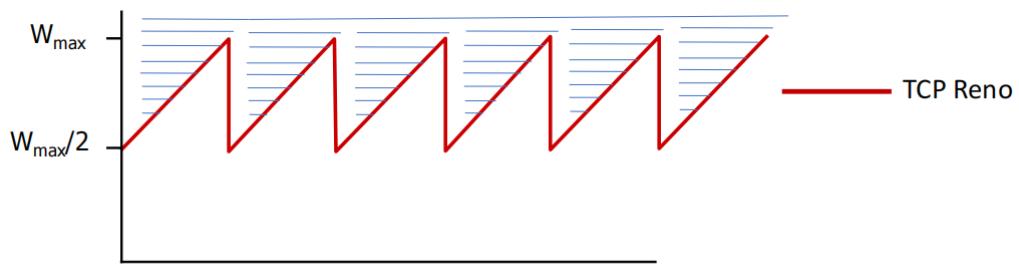


Figure 144

6.8.1.2 TCP-BIC

TCP-BIC: Binary Increase Congestion

BIC 算法对满载窗口进行二分查找：

- 如发生丢包时窗口大小是 W_1 ，为保持满载而不丢包，满载窗口应小于 W_1
- 如检测到丢包并将窗口乘性减小为 W_2 ，则满载窗口应大于 W_2

窗口更新受 ACK 时钟驱动，即以 RTT 为更新间隔时间

二分查找：

- 在 ACK 时钟的驱动下，将拥塞窗口置为 $(W_1 + W_2)/2$ (新的 W_2 值)，不断逼近满载窗口

最大探查：

- 如窗口再次达到 W_1 而没有丢包，说明满载窗口大于 W_1 ，则以逼近 W_1 的镜像过程增大拥塞窗口

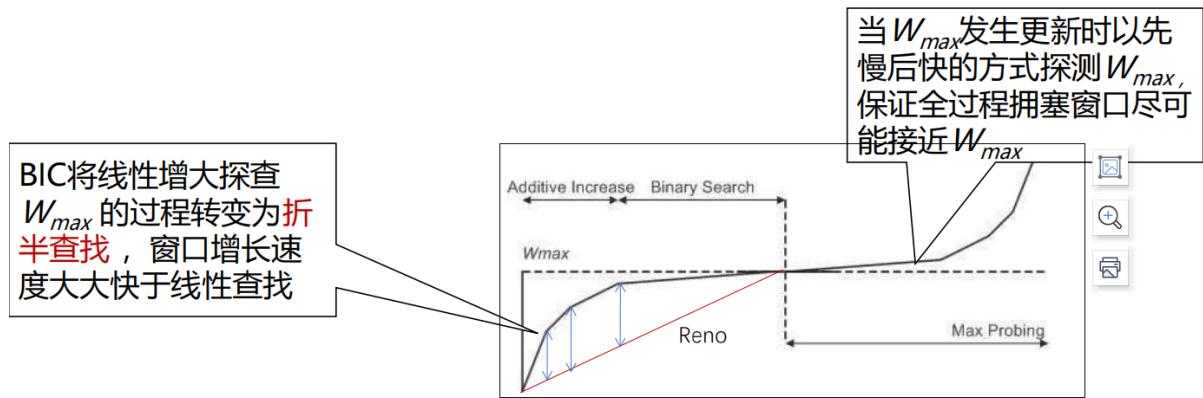


Figure 145

BIC 存在带宽 不公平性问题：BIC 以 ACK 时钟驱动拥塞窗口的更新，RTT 较短的连接会更快到达满载窗口，占据更多的带宽，产生不公平性问题 (RTT-fairness)

6.8.1.3 TCP CUBIC

CUBIC 将 BIC 算法连续化，用三次函数拟合 BIC 算法曲线

拥塞窗口成为距上次丢包的时间 t 的函数， t 取值位于两次丢包之间，不再根据 RTT 间隔来确定调整窗口的时机，避免了 RTT 不公平问题

➤ 三次函数增长分为两个阶段：

- Steady State Behavior 阶段：以凹函数增长逼近最近一次丢包时窗口；
- Max probing 阶段：以凸函数增长探测当前满载窗口

➤ 拥塞窗口在绝大多数时间内接近 W_{max} ，保持了较高的发送效率

➤ CUBIC 已实现在 Linux 2.6.18 中

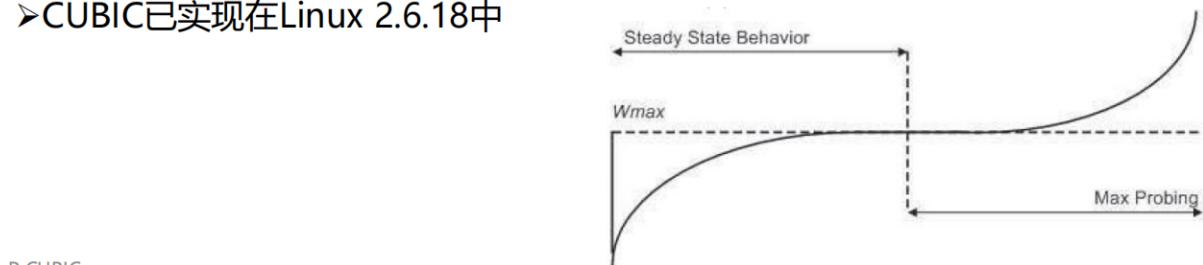


Figure 146

6.8.2 Google BBR

6.8.2.1 拥塞与瓶颈链路带宽

➤拥塞与瓶颈链路带宽：

- 瓶颈链路带宽 $BtlBw$, 决定了端到端路径上的最大数据投递速率
- 拥塞窗口大于 $BtlBw$ 时, 瓶颈链路处会形成排队, 导致RTT延长(直至超时)

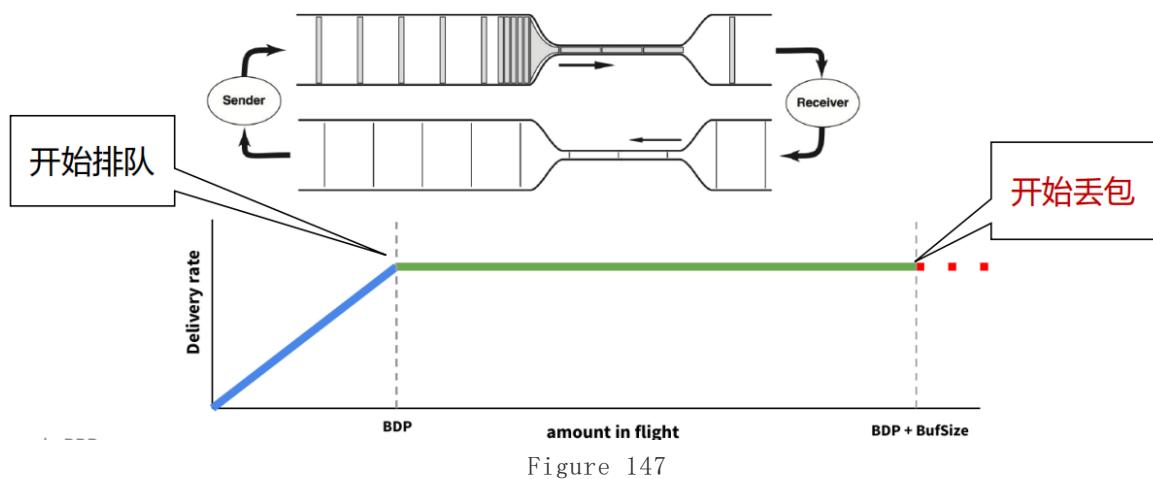


Figure 147

BBR: Bottleneck Bandwidth and Round-trip propagation time

瓶颈链路带宽 $BtlBw$: 不会引起路由器缓存排队的最大发送速率

RTprop: 往返时间

$$BDP = BtlBw \times RTprop$$

6.8.2.2 优化点的近似观测

用过去 10 秒内的最小 RTT (min RTT) 和最大投递率 (max BW), 分别近似 RTprop 和 BtlBw, 并依据这两个值估算当前 BDP

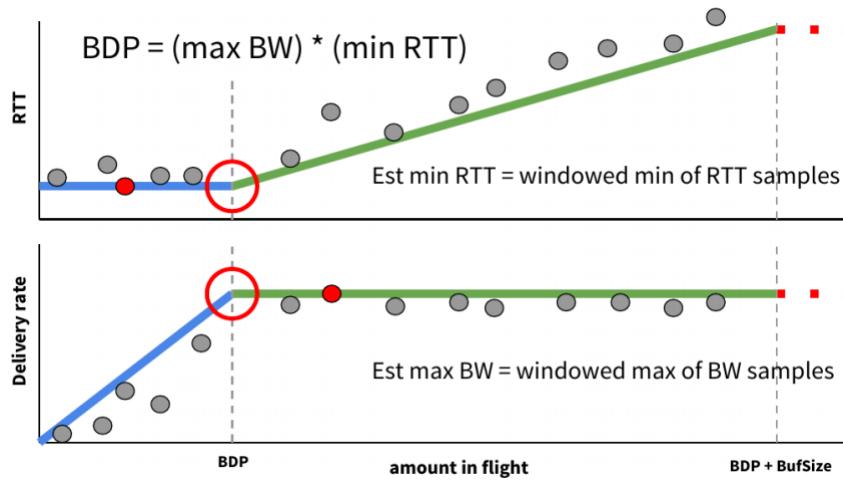


Figure 148

⚠ Caution

Max BW 和 min RTT 不能同时被测得!

- 要测量最大带宽，就要把瓶颈链路填满，此时buffer中存在排队分组，延迟较高
- 要测量最低延迟，就要保证链路队列为空，网络中分组越少越好，cwnd较小

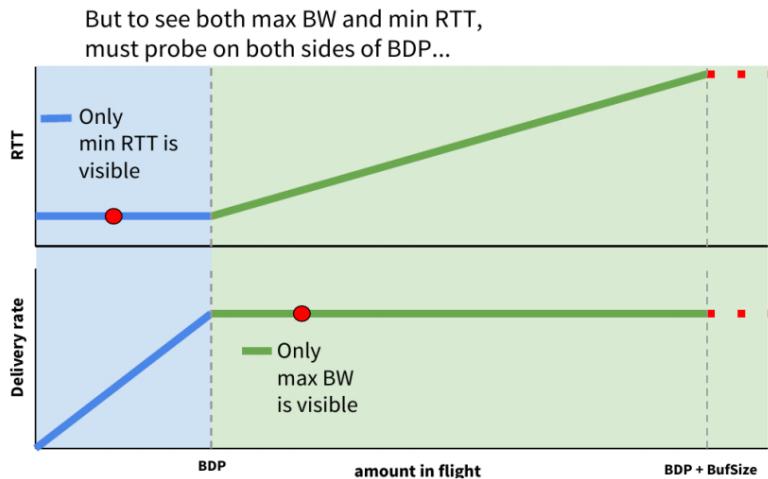


Figure 149

6.8.2.3 BDP 检测阶段

启动阶段 (START_UP)

当连接建立时，类似 TCP 的慢启动，指数增加发送速率，尽可能快地占满管道

若经过三次发现投递率不再增长，说明已达到 BtlBw，瓶颈链路处分组已开始排队（事实上此时占的是三倍 BDP）

排空阶段 (DRAIN)

指数降低发送速率（相当于是 startup 的逆过程），将多占的两倍 buffer 慢慢排空

瓶颈带宽探测 (PROBE_BW)

- 进入稳定状态后，先在一个RTT内
增加发送速率，探测最大带宽
- 如果RTT没有变化，再减小发送
速率，排空前一个RTT多发出来
的包
- 后面6个周期使用更新后的估计
带宽发送

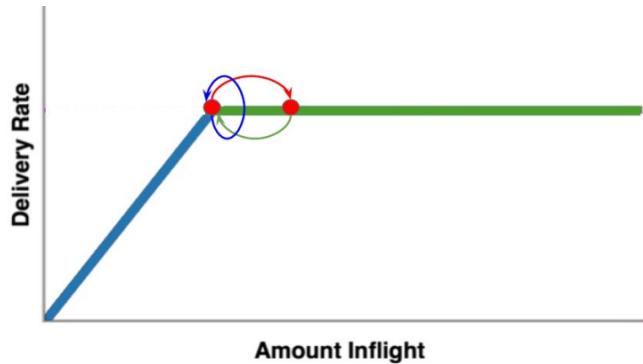


Figure 150

时延探测 (PROBE_RTT)

- 每过10秒，如果估计的RTprop不变，就进入RTprop探测阶段
- 在这段占全过程2%的时间内，cwnd固定为4个包
- 测得的RTprop作为基准，用以判断带宽检测阶段瓶颈链路中是否发生排队
- 为抵消此阶段牺牲的发送速率，结合pacing_gain数组在后期短时间增加发送速率，保证对BtlBw的及时观测
- pacing_gain数组用来微调每个状态下不同时段的拥塞窗口大小

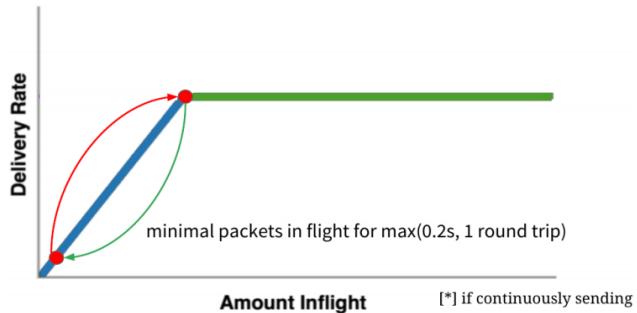


Figure 151

6.8.3 Data Center TCP

数据中心内部有一类经典的通信模式-**Partition/Aggregate**

- Aggregator 将上层应用任务划分给下层的 worker 执行

6.8.3.1 数据中心的性能问题

Incast: 在一个很短的时间内，大量流量同时到达交换机的一个端口，导致缓冲区被占满，最终导致丢包

Queue Buildup:

Buffer Pressure:

6.8.3.2 需要什么样的传输层协议

容忍高突发流量

- 在 Partition/Aggregate 通信模式中，所有 Worker 几乎会在同一时间向 Aggregator 返回执行结果，产生很高的突发流量

低时延

- 数据中心有大量时延敏感的短流，如网页搜索等

高吞吐

- 数据中心有大量吞吐敏感的长流，如文件传输、分布式机器学习中神经

网络模型参数的传输等

6.8.3.3 DCTCP 核心思想

根据网络拥塞程度精细地减小发送窗口

容忍高突发流量

- DCTCP 维持了较低的队列长度，可以留出较大的缓冲区给突发流量
- 采用激进的标记策略，使得发送端在丢包之前就感知到拥塞

低时延

- 由于队列长度较短，也减少了包在队列中的排队时延

高吞吐

- DCTCP 根据拥塞程度精确调节窗口，使得发送窗口的变化比较平滑，不会出现吞吐量骤降的情形

6.8.4 DCCP

DCCP = UDP+拥塞控制

- DCCP 提供不可靠数据报流服务：
 - UDP 报文段可能丢失、乱序到达
 - 无流量控制
- DCCP 提供模块化的拥塞控制：
 - 应用可以根据需要，选择不同的拥塞控制机制

6.9 传输层协议的发展

6.9.1 MPTCP

传统 TCP 协议仅支持单路径传输，即只能利用终端主机上的一个网络接口传输数据

6.9.1.1 MPTCP 的优势

多径带宽聚合

- 终端设备可以聚合不同路径上的可用带宽，以获得更高的网络带宽

提升传输的可靠性

- 使用多条路径传输数据，可以有效避免因单条路径性能恶化或中断导致的应用连接中断

支持链路的平滑切换

- 多路径传输方式允许终端在不同接入网络间快速、平滑地切换，选取当前链路质量最好的路径传输数据

6.9.1.2 MPTCP 在网络体系结构中的位置

MPTCP 位于套接字和 TCP 之间

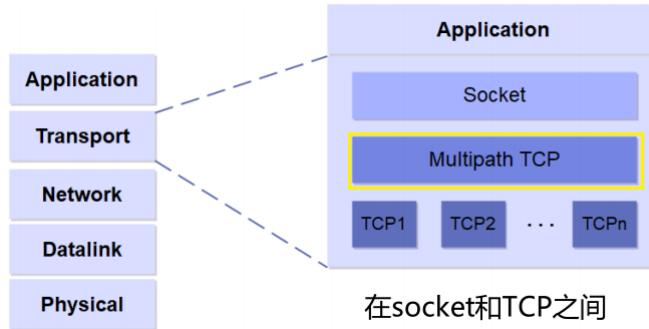


Figure 152

公平性是关注的重点，MPTCP 采用的是网络公平性原则

6.9.2 QUIC

6.9.2.1 TCP 存在的问题

- TCP 实现在操作系统内核中
 - 作为传输优化的最终受益者，应用无法对 TCP 进行修改
 - 操作系统的更新往往跟不上应用的需求和节奏
- TCP 体系握手时延大
 - 互联网上的大趋势：低时延需求越来越强烈；加密流量占比越来越大
 - TLS(传输层安全性协议)+TCP 的体系握手时延很大，传输前需要 3 个 RTT 进行握手

TCP 多流复用存在队头阻塞问题-> 多流复用

TCP 传输需要保持有序性：出现丢包时，后面的数据需要等丢失的包重传完成才能使用，这就导致了队头阻塞

6.9.2.2 QUIC 在网络体系结构中的位置

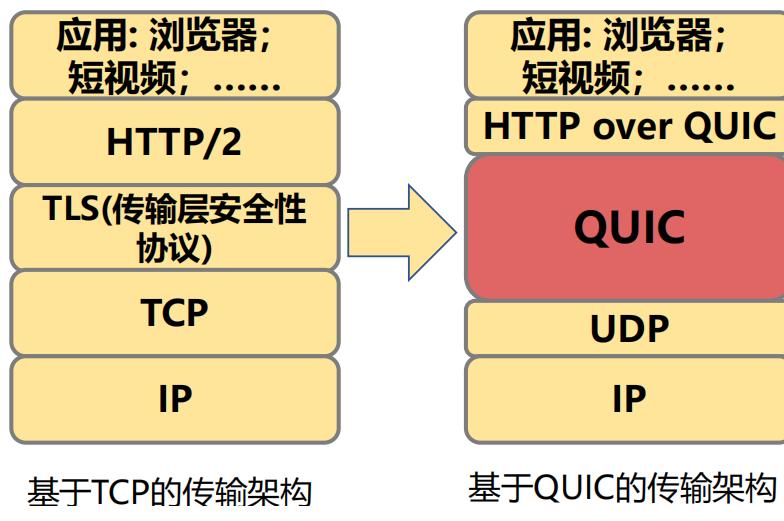


Figure 153

QUIC 替代 TCP、TLS 和部分 HTTP 的功能

7. 应用层

Important

1. 掌握应用进程通信方式以及服务进程工作模式
2. 掌握域名系统基本原理和工作机制
3. 掌握电子邮件系统体系结构及基本工作原理
4. 了解 WWW 系统结构框架、静态和动态 Web 及其应用技术，掌握 HTTP 协议及其工作原理
5. 了解流媒体基本概念、数字音视频与编码、流式存储媒体、直播与实时音视频、流媒体动态自适应传输
6. 了解内容分发背景及内容分发网络、以及 P2P 网络的工作机制
7. 掌握 Telnet、FTP、SNMP 等应用层协议的工作原理

7.1 应用层概述

7.1.1 应用进程通信方式

客户/服务器 (C/S, Client/Server) 方式

- 面向连接时，C/S 通信关系一旦建立，通信就是双向的，双方地位平等，都可发送和接收数据

浏览器/服务器 (B/S, Browser/Server) 方式

- 浏览器请求、服务器响应的工作模式；用户界面完全通过 Web 浏览器实现，一部分事务逻辑在前端实现，但主要的事务逻辑在服务器端实现
- 特点：
 - 界面统一，使用简单。客户端只需要安装浏览器软件
 - 易于维护。对应用系统升级时，只需更新服务器端的软件，减轻了系统维护和升级的成本
 - 可扩展性好。采用标准的 TCP/IP 和 HTTP 协议，具有良好的扩展性
 - 信息共享度高。HTML 是数据格式的一个开放标准，目前大多数流行的软件均支持 HTML

对等 (P2P, Peer to Peer) 方式

- 两个进程在通信时不区分服务的请求方和服务的提供方
- 每一个 P2P 进程既是客户同时也是服务器

7.1.2 服务器进程工作方式

循环方式(iterative mode)

- 一次只运行一个服务进程
- 当有多个客户进程请求服务时，服务进程就按请求的先后顺序依次做出响应(阻塞方式)

并发方式(concurrent mode)

- 可以同时运行多个服务进程
- 每一个服务进程都对某个特定的客户进程做出响应(非阻塞方式)

7.2 域名系统

7.2.1 历史和概述

域名系统 (DNS, Domain Name System) 是互联网重要的基础设施之一，向所有需要域名解析的应用提供服务，主要负责将可读性好的域名映射成 IP 地址

7.2.2 域名系统名字空间和层次结构

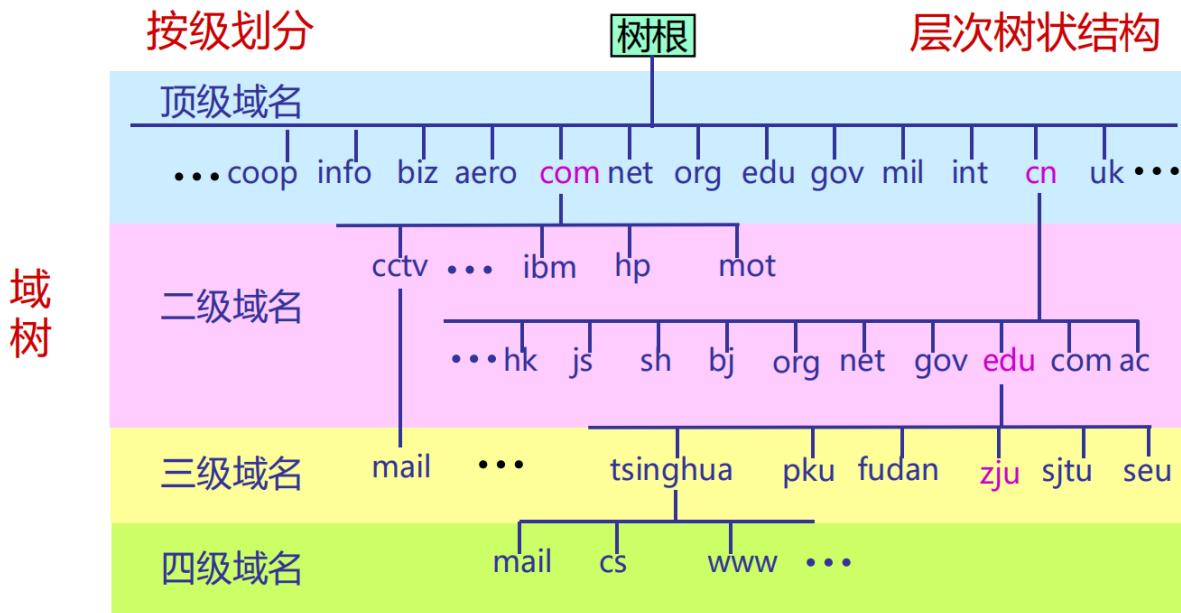


Figure 154

edu 可以时顶级域名也可以是二级域名，美国的大学的 edu 就是顶级域名

7.2.2.1 域名系统名字空间和层次结构

顶级域名TLD (Top Level Domain) 一般有三类

- **国家或地区顶级域nTLD**，也记为ccTLD (cc: country code)
 - 例如.cn 表示中国，.us 表示美国，.uk 表示英国。目前有300多个
- **基础设施域.arpa (Address and Routing Parameter Area)**
 - 专用于Internet基础设施目的
 - 目前有二级域ip6.arpa; iris.arpa; in-addr.arpa; uri.arpa; urn.arpa; home.arpa; as112.arpa; in-addr-servers.arpa; ipv4only.arpa等
- **通用顶级域gTLD**
 - 早期规定了20个通用顶级域名，2011年批准新通用顶级域名 (New Generic Top-level Domain, New gTLD)
 - 截至2020年，已注册有1200多个通用顶级域名

Figure 155

7.2.3 域名服务器

保存关于域树(domain tree)的结构和设置信息的服务器程序称为 **名字服务器(name server)**或**域名服务器**，负责域名解析工作

域名解析过程对用户透明

名字服务器的管辖范围以“区”为单位，而不是以“域”为单位

管辖区是域名“域”的子集。管辖区可以小于或等于域，但不可能大于域

7.2.3.1 域名服务器分类

总体上，域名系统的名字服务器分为两大类

- **权威名字服务器(authoritative name server)**
- 一种根据本地知识知道一个 DNS 区(zone)的内容的服务器，它可以回答

有关该 DNS 区的查询而无需查询其他服务器

- 每个 DNS 区至少应有一个 IPv4 可访问的权威名字服务器提供服务
- **递归解析器(recursive resolver)/递归服务器**
- 以递归方式运行的、使用户程序联系域(domain)名字服务器的程序。

7.2.3.2 权威名字服务器

根据对应域的层次，权威名字服务器又进一步分为以下类别

- 根名字服务器(root name server) /根服务器(root server):
 - 每个根服务器都知道所有的顶级域名服务器的域名及其 IP 地址
 - 根服务器并不直接把主机用户所查的域名转换成 IP 地址
 - 根服务器共有 13 套，每套都可以有多个镜像(mirrored)根服务器，其内容定期与上述对应的根服务器同步
- 顶级域名字服务器(TLD name server)
- 二级域名字服务器(second level domain name server)
 - 一般来说，每个名字服务器只负责解析本辖域的域名
- 三级域名字服务器(third level domain name server)
- 三级域及以下的名字服务器也统称为本地域名服务器

7.2.3.3 递归解析器/递归服务器

每一个 Internet 服务提供者 **ISP(Internet Service Provider)**，都至少有一个递归服务器，距离用户主机较近

在递归服务器/本地域名服务器有多层的结构中，当本层服务器没有解析结果时，通常是逐级向上递归进行查询

当查询请求到达本地域最上一层名字服务器时，该层服务器对自己无法解析的任何一个域名，首先求助于根服务器（而不是二级域名字服务器），开始逐步迭代查询

7.2.4 域名解析过程

当某一应用进程需要进行域名解析时，该应用进程将域名放在 DNS 请求报文（**UDP 数据报, 端口号为 53**）发给递归服务器（使用 UDP 是为了减少开销）。递归服务器得到查询结果后，将对应 IP 地址放在应答报文中返回给应用进程

域名查询有 **递归查询(recursive query)** 和 **迭代查询(或循环查询, iterative query)** 两种方式

- 主机向递归解析器/本地域名字服务器的查询一般采用 **递归查询**
- 递归解析器/本地域名字服务器向根服务器可以采用递归查询，但一般优先采用 **迭代查询**

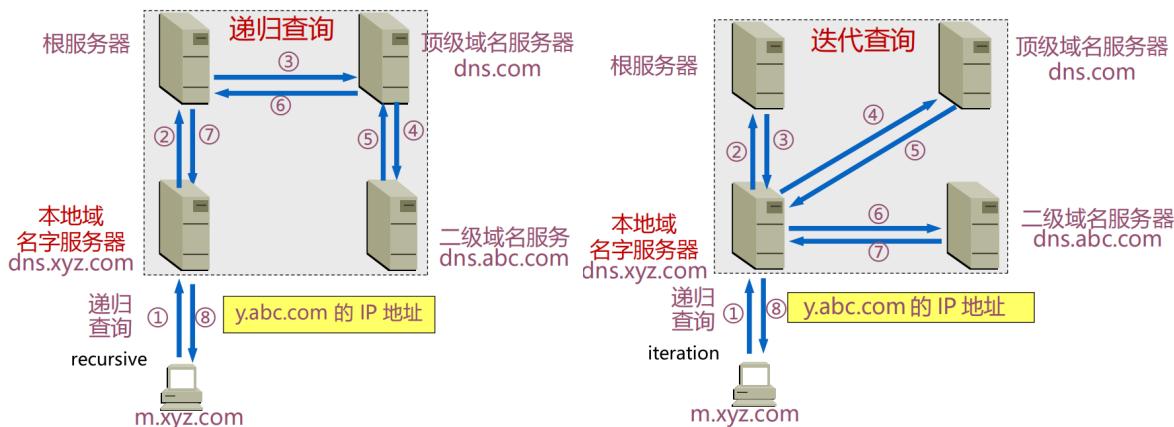


Figure 156

7.2.5 域名系统查询和响应(选讲)

常用的资源(Resource)类型

类型代码	描述	功能
SOA (Start Of Authority)	权威记录的起始	指定有关DNS区域的权威性信息，包含主要名字服务器、域名管理员的电邮地址、域名的流水式编号、和几个有关刷新区域的定时器
A (Address)	IP地址记录	传回一个 32 比特的 IPv4 地址
AAAA (Address)	IPv6地址记录	传回一个 128 比特的 IPv6 地址
MXMX (Mail Exchanger)	电子邮件交互记录	引导域名到该域名的邮件传输代理 (MTA, Message Transfer Agents) 列表
CNAME (Canonical NAME)	规范名称记录	主机名字的别名：域名系统将会继续尝试查找新的名字

Figure 157

类型代码	描述	功能
PTR(pointer)	指针记录	引导至一个规范名称(Canonical Name), 最常用来运行反向 DNS 查找
NS (Name Server)	名称服务器记录	用于说明这个区域有哪些DNS服务器负责解析
SPF (sender policy framework)	SPF记录	让一个域把邮件服务信息进行编码, 即域内哪台机器负责把邮件发送到Internet其余地方, 有助于接受机器检查邮件是否有效
SRV (Service)	服务定位器	把主机标识为域内的一种给定服务。例如 cs.washington.edu的web服务器可以标识成cockatoo.cs.washington.edu
TXT (Text)	文本记录	最初用于允许域以任意方式标识自己, 如今通常被编码成机器可读信息, 一般是SPF信息

Figure 158

资源记录部分是 DNS 报文格式的最后 3 个字段, 只有在 DNS 响应报文中才出现, 包括回答问题区域字段、权威名字服务器区域字段、附加信息区域字段。这 3 个字段都采用资源记录的格式

RR 格式: (name, ttl, class, type, value)

7.2.5.1 DNSSEC

DNSSEC 依靠数字签名保证 DNS 应答报文的真实性和完整性

7.2.6 域名系统高速缓存

DNS 高速缓存作用: 为提高 DNS 查询效率, 并减轻根域名服务器的负荷和减少 Internet 上的 DNS 查询报文数量, 域名服务器广泛使用高速缓存, 用来存放最近查询过的域名以及从何处获得域名映射信息的记录

好处: 可以大大减轻根域名服务器的负荷, 而且也能够使 Internet 上的 DNS 查询请求和回答报文的数量大为减少

7.2.7 域名系统隐私(选讲)

QNAME 是用户请求的域名全名, 它提供了用户的操作信息

DNS 敏感数据主要的泄露途径: 通信链路窃听, 服务器收集

思考题: What happens from opening a URL in a browser to the website being displayed?

<https://chatgpt.com/share/675122ac-d0a8-8002-9b88-0c990dcfe05f>

7.3 电子邮件

7.3.1 电子邮件系统体系结构

电子邮件系统采用客户/服务器 (C/S) 工作模式

- 用户代理 (user agent) ——邮件客户端
 - 无统一标准
- 传输代理 (message transfer agent) ——邮件服务器
- 简单邮件传输协议 SMTP (Simple Mail Transfer Protocol) ——邮件服务器之间传递邮件使用的协议

- SMTP 利用 TCP 可靠地从客户向服务器传递邮件，使用端口 25
- SMTP 的 3 个阶段：连接建立、邮件传送、连接关闭
- SMTP 是一个简单的 ASCII 协议，邮件必须为 7 位 ASCII
- 不包括认证，不能传二进制数据（只能 ASCII），邮件以明文形式出现

7.3.2 邮件格式

RFC 5322—Internet 邮件格式

- 基本 ASCII 电子邮件使用 RFC 5322
- 最新修订的 Internet 邮件格式，其前身是 RFC 822

MIME(Multipurpose Internet Mail Extensions)—多用途 Internet 邮件扩展

- 基本格式的多媒体扩展
- 可传输多媒体消息和二进制文件

7.3.3 最终交付协议

7.3.3.1 POP3 协议

POP3 由 RFC1939 定义，是一个非常简单的最终交付协议

POP3 使用客户/服务器工作方式，在接收邮件的用户 PC 机中必须运行 POP 客户程序，

而在用户所连接的 ISP 的邮件服务器中则运行 POP 服务器程序

7.3.3.2 IMAP

- 用于最终交付的主要协议
- IMAP 是较早使用的最终交付协议—POP3(邮局协议，版本 3)的改进版
- 邮件服务器运行侦听端口 143 的 IMAP 服务器

7.4 WWW

7.4.1 WWW 体系结构概述

WWW = World Wide Web = 万维网

WWW 性能提升方法

缓存 (caching)，如代理技术

多线程 (multiple threads)

前端 (front end)

7.4.2 静态 Web (对象)

· WEB网页对象

- 静态对象与静态网页

- 文本，表格，图片，图像和视频等多媒体类型的信息（实现语言：标记语言，如：HTML，XML，PHP等）

- 字体、颜色和布局等风格类型的信息（实现语言：层叠样式表CSS）

- 动态对象与动态网页

- 交互信息，比如，用户注册信息、登录信息等（实现：PHP/JSP等语言+MySQL等数据库）

- 链接

- 超链接 (HyperLinks)

Figure 159

HTML 和 XML

比较内容	HTML	XML
扩展性	不具有可扩展性，标记固定	元标记语言，可定义新的标记语言，标记可由用户定义
侧重点	侧重信息的表现形式	侧重结构化的信息描述
语法	不严格（嵌套、配对）	严格要求嵌套、配对严格按照DTD的要求
可读性和可维护性	难于阅读，难于维护	结构清晰，便于阅读和维护
数据本身与显示	数据与显示通常合在一起	数据与显示分离
重用性	低	高

Figure 160

7.4.3 动态 Web 和 Web 应用

7.4.3.1 动态 Web 页面

- 通用网关接口 CGI (*.cgi)
- 脚本语言+数据库技术 (*.php, *.asp, *.aspx) -服务器端动态网页
- 客户端动态网页 (*.js)

7.4.3.2 通用网关接口 CGI

CGI (Common Gateway Interface) 是一种标准

7.4.3.3 脚本语言+数据库技术

PHP (Hypertext Preprocessor)

JSP (JavaServer Pages)

ASP.NET (Active Server Pages .NET)

7.4.3.4 Client-side dynamic web page generation

JavaScript 客户端动态 web

7.4.3.5 典型的动态 Web 技术：AJAX

AJAX: Asynchronous JavaScript and XML

- 超文本标记语言 HTML+CSS: 用于 Web 网页内容的显示
- 文档对象模型 DOM (Document Object Model) : 文档对象模型 (DOM) 是 HTML 和 XML 文档的编程接口，其本质上是页面的 API，允许程序读取和操作页面的内容、结构和样式；采用树形结构组织
- 扩展标记语言 XML (eXtensible Markup Language) : 用于程序与服务器交换应用数据
- 异步式工作方式：用于发送和检索 XML 数据
- JavaScript: 用于将以上所有功能进行组合，并协同工作

AJAX 不是新的编程语言，而是一种使用现有标准的新方法

7.4.4 HTTP 协议

超文本传输协议 HTTP (HyperText Transfer Protocol) 在传输层通常使用 **TCP 协议**，**缺省使用 TCP 的 80 端口**

HTTP 为 **无状态协议**，服务器端 **不保留之前请求的状态信息**

- 无状态协议：效率低、但简单
- 有状态协议：维护状态相对复杂，需要维护历史信息，在客户端或服务器出现故障时，需要保持状态的一致性等

7.4.4.1 HTTP 发展现状

- HTTP/1.0 (1996)
 - 无状态，非持久连接
- 与 HTTP/1.1 (1999)
 - 支持长连接和流水线机制
 - 缓存策略优化、部分资源请求及断点续传
- HTTPS: HTTP+TLS (2008)
 - 增加SSL/TLS (TLS 1.2) 层，在TCP之上提供安全机制
- HTTP/2.0 (2015、2020)
 - 目标：提高带宽利用率、降低延迟
 - 增加二进制格式、TCP多路复用、头压缩、服务端推送等功能

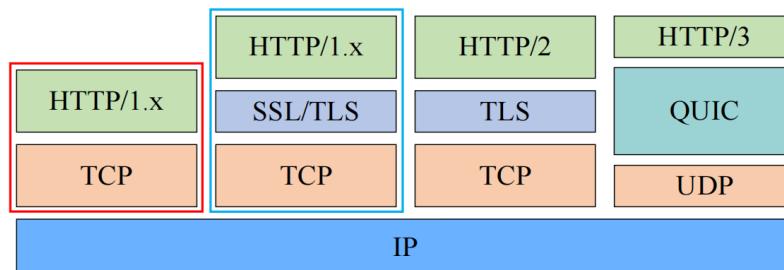


Figure 161

HTTP/1.0 缺省为非持久连接

服务器接收请求、响应、关闭

TCP 连接

- 获取每个对象需要两阶段
- 建立 TCP 连接
- 对象请求和传输
- 每次连接需要经历 TCP 慢启动阶段

HTTP/1.1 缺省为持久连接

- 在相同的 TCP 连接上，服务器接收请求、响应；再接收请求、响应；响应后保持连接

HTTP/1.1 支持流水线机制

- 需要按序响应
- 经历较少的慢启动过程，减少往返

返时间

- 降低响应时间

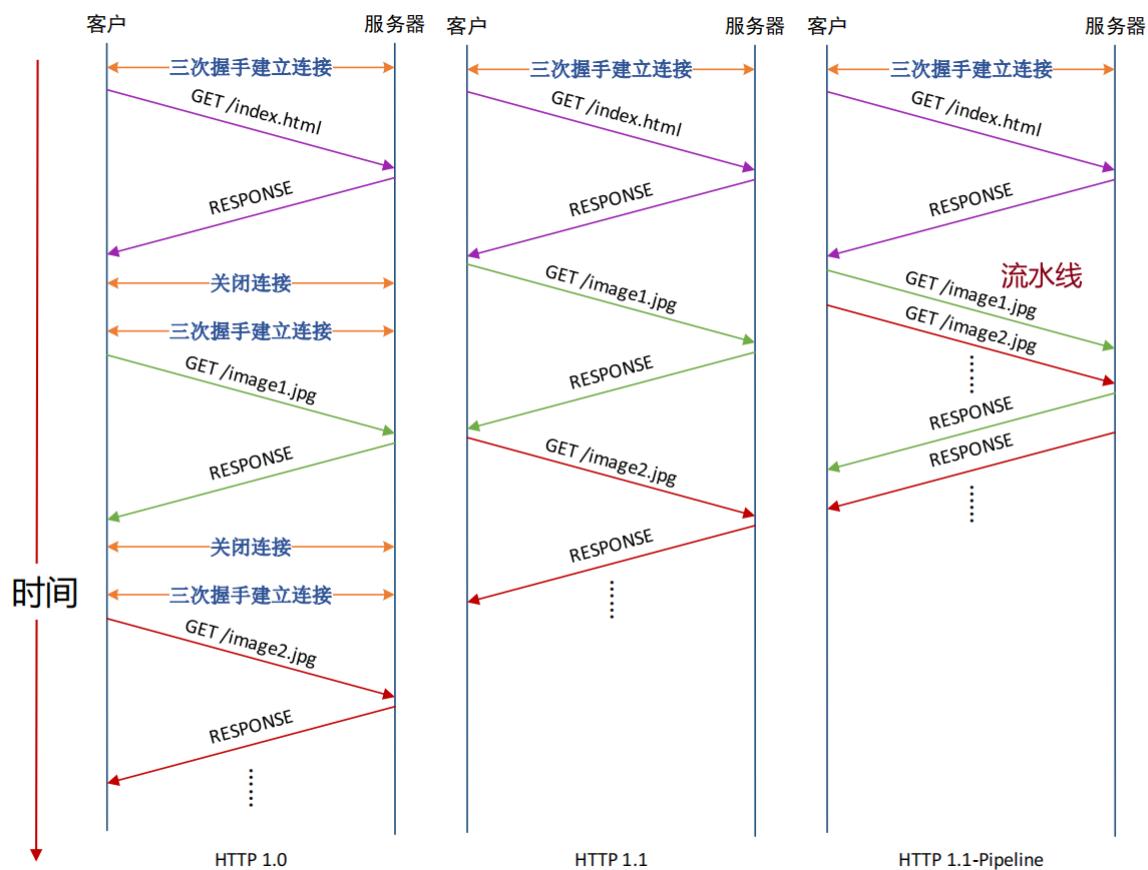


Figure 162

7.4.4.2 思考题

- Suppose within your Web browser you click on a link to obtain a Web page. Suppose that the IP address for the associated URL is not cached in your local host, so that a DNS lookup is necessary to obtain the IP address. Suppose that n DNS servers are visited before your host receives the IP address from DNS; the successive visits incur an RTT of RTT_1, \dots, RTT_n . Further suppose that the Web page associated with the link contains exactly one object, a small amount of HTML text. Let RTT_0 denote the RTT between the local host and the server containing the object. Assuming zero transmission time of the object, how much time elapses from when the client clicks on the link until the client receives the object? $RTT_1 + \dots + RTT_n$ DNS解析
- Suppose the HTML file indexes *three* very small objects on the same server. Neglecting transmission times, how much time elapses with
 - Multiple HTTP connections and sequential requests, $(2+2*3)RTT_0 + RTT_1 + \dots + RTT_n$
 - persistent HTTP connection and sequential requests, $(2+3)RTT_0 + RTT_1 + \dots + RTT_n$
 - persistent HTTP connection and pipelined requests $(2+1)RTT_0 + RTT_1 + \dots + RTT_n$

Figure 163

7.4.5 Web 缓存技术与 Web 代理

7.4.5.1 浏览器缓存

- 目标：再次访问缓存在浏览器主机中的 Web 页副本，不必从原始服务器读取

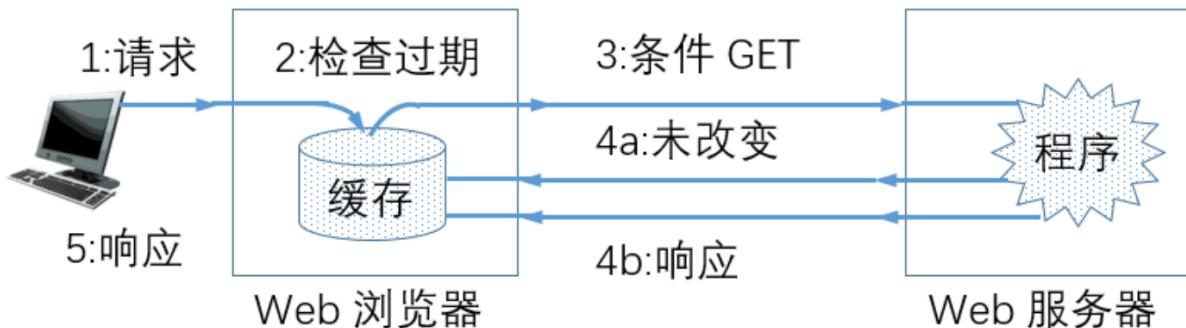


Figure 164

7.4.5.2 Web 代理服务器缓存

- 目标：代理服务器缓存已访问过的 Web 页副本，满足用户浏览器从代理服务器提取 Web 页，尽量减少原始服务器参与
- 设置用户浏览器，通过代理服务器进行 Web 访问
- 浏览器将所有的 HTTP 请求发送到代理服务器
- 如果缓存中有被请求的对象，则直接返回对象
- 否则，代理服务器向原始服务器请求对象，再将对象返回给客户端

降低时延、减少网络流量

7.4.5.3 Web 代理服务器缓存性能分析

➤ Web代理服务器缓存性能分析

- 假设:

- ▶ 平均每个Web页对象大小: 1M bits
- ▶ 机构网络内用户向原始服务器请求对象速率为平均每秒15个请求:15/sec
- ▶ 互联网访问原始Web页的平均往返时延为2秒
- ▶ 接入链路速率为15Mbps

- 结果:

- ▶ 机构网络内带宽使用率1.5%，接入链路的使用率接近1
- ▶ 总的时延: = 互联网时延+接入链路时延+ 机构网络时延
 $= 2 \text{ sec} + \text{minutes} + \text{msecs}$ (机构网络时延以毫秒记)

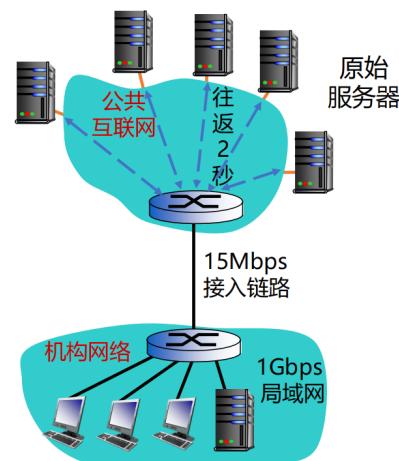


Figure 165

➤ Web代理服务器缓存性能分析

- 假设:

- ▶ 缓存代理访问命中率为40%
- ▶ 平均每个Web页对象大小: 1M bits
- ▶ 机构网络内用户向原始服务器请求对象速率为平均每秒15个请求:15/sec

- 结果:

- ▶ 机构网络内带宽使用率1.5%，接入链路的使用率接近0.6
- ▶ 总的时延: = $0.6 * (\text{互联网延迟}) + 0.4 * (\text{Web代理命中率的延迟})$
 $= 0.6 (2\text{秒} + \text{计10毫秒机构网络内延迟}) + 0.4 (\text{计10毫秒机构网络内延迟})$
 $= \sim 1.2 \text{ secs}$

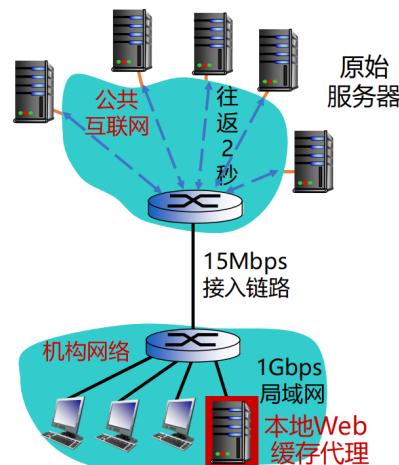


Figure 166

Web 缓存存在的困难，即 Web 缓存与原始服务器 Web 页一致性问题

- 启发式策略: 服务器响应 Web 页的 Last-Modified 头和 Expires 头启发，原始 Web 页一段时间没更改，最近一段时间也不会更改。Expires 头很少用
- 询问式策略: 通过特殊的关键字头询问原始服务器，Web 副本是否已更新

7.4.6 Web 安全与隐私

7.4.6.1 Web 访问安全

- 无状态: 客户端需要在每个请求中携带认证信息
- 认证方法: 通常在 HTTP 请求中使用“用户名-密码”
- 每个请求头中包含关键字 authorization:
- 如果请求头中无 authorization:，则服务器拒绝访问，并在响应头中包含 WWW-authenticate:

7.4.6.2 Cookie

HTTP 无状态协议，服务器用 cookies 保持用户状态

- HTTP 在响应的首部行里使用一个关键字头 set-cookie:

选择的 cookie 号具有唯一性

- 后继的 HTTP 请求中使用服务器响应分配的 cookie:
- Cookie 文件保存在用户的主机中，内容是服务器返回

的一些附加信息，由用户主机中的浏览器管理

- Web 服务器建立后端数据库，记录用户信息，cookie 作为关键字

Cookie 技术是把双刃剑，能分析用户喜好，向用户进行个性化推荐

Cookie 技术是把双刃剑，也能跟踪用户网络浏览痕迹，泄露用户隐私

① Caution

误区：Cookie 容易嵌入间谍程序

Cookie 文件保存的只是文本串，没有可执行程序，不会被嵌入间谍程序

- 用户可以设置浏览器限制使用 Cookie

7.5 流式音频和视频

7.5.1 流媒体概述

常见的流媒体服务：

- 点播：提前录制好，边下载边播放（起始时延 < 10s；类 VCR 操作（例如拖动进度条）< 1~2s）
- 直播：边录制边上传，边下载边播放（大规模直播往往有数秒的时延）
- 实时交互：双方或多方实时交互式通信（时延 < 400ms 可接受，VR 则需要 < 25ms）

流媒体的特性：

- 端到端时延约束
- 时序性约束：流媒体数据必须按照一定的顺序连续播放
- 具有一定程度的容错性：丢失部分数据包也可完成基本功能

7.5.2 数字音视频与编码

MPEG 视频压缩

I 帧必须周期性地出现在媒体流中

7.5.3 直播与实时音视频

实际流媒体系统往往先尝试 UDP，如果失败则转为 TCP

实时流式协议 RTSP (Real-Time Streaming Protocol)

实时传输协议 RTP (Real-time Transport Protocol)

实时传输控制协议 RTCP (RTP Control Protocol)

网页实时通信 WebRTC (Web Real-Time Communication)

7.6 内容分发

7.6.1 内容和 Internet 流量

内容分发，尤其是视频，是 Internet 流量的一个重要构成

7.6.2 服务器群和 Web 代理

> 代理缓存帮助组织扩展 Web

- 将服务器内容缓存到客户上以提高性能
- 实现组织的策略(如访问策略)

7.6.3 内容分发网络 CDN

怎样将内容（如从百万的视频中选定的内容）分发给同时发起访问的数百万用户？

- 单个、大型的“服务器”？
 - 单点故障，网络拥塞，远程用户的长路径
 - 显然，这种解决方案 可扩展性差

内容分发网络 CDN (Content Delivery Network, or Content Distribution Network)

- 基本思想源于 MIT 对 Web 服务瞬间拥塞问题的解决 (1998)
 - 一种 Web 缓存系统，靠近 网络边缘（用户） 提供内容服务
 - 目前提供更丰富的服务，包括静态内容、流媒体、用户上传视频等
- 主要优点
 - 降低响应时延，避免网络拥塞
 - 避免原始服务器过载及防止 DDoS 攻击
 - 分布式架构，具有良好的可扩展性
 - 对用户透明，无需用户感知

DNS 重定向实现 CDN

将请求调度到较近或负载较轻的 CDN 服务器

> HTTP 重定向请求内容，服务提供者返回清单 CDN

- 原始服务器决策 CDN 服务器

- HTTP 响应：状态码 30X，Location：指明新的位置

DNS 辅助实现 CDN

7.6.4 P2P 网络实现内容分发

P2P 文件分发协议：BitTorrent

- 文件被划分为 256Kb 大小的块
- 具有种子(torrents)的节点发送或接收文件

分布式哈希表 (Distributed Hash Tables, DHTs) 是一个完全分布式索引，可扩展到多客户端 (或实体)

7.7 其它应用层协议

7.7.1 Telnet

1983 年由 RFC 854 确定为 Internet 标准

Telnet 协议使用 C/S 方式实现

Telnet 协议使用 TCP 连接通信，服务器进程默认监听 TCP23 端口

Telnet 协议引入网络虚拟终端 NVT (Network Virtual Terminal)

NVT 是 Telnet 协议定义的一组通用字符集

使用 Telnet 协议在网络中传输的数据都是 NVT 格式，不同的用户终端与服务器进程均与本地终端格式无关

7.7.2 FTP

文件传输协议 FTP(File Transfer Protocol)是 Internet 上使用最广泛的应用层协议之一

RFC 959 早在 1985 年就已经成为 Internet 的正式标准

FTP 使用 C/S 方式实现

服务器主进程打开 TCP21 端口，等待客户进程发出的连接请求

客户请求到来时，服务器主进程启动从属进程来处理客户进程发来的请求

FTP 的两个端口与两个连接

- 控制连接在整个会话期间一直保持，客户进程发出的文件传输请求通过控制连接发送给服

务器控制进程 (工作在 TCP21 端口)，但控制连接不用来传输文件

- 服务器控制进程在接收到客户进程发送来的文件传输请求后就创建数据传输进程 (工作在 TCP20 端口) 和数据连接
- 数据连接用来连接客户进程和服务器数据传输进程，实际完成文件的传输。服务器数据传输进程在文件传输完毕后关闭数据连接并结束运行

使用 C/S 方式和 UDP 协议实现

- 只支持文件传输而不支持交互

TFTP 的工作过程类似停止-等待协议

UDP 端口号为 69

7.7.3 SNMP

使用 UDP 协议，以简化设计和降低通信开销

SNMP 使用 C/S 方式实现

- 代理运行服务器端进程在 UDP 端口 161 接收 get 或 set 报文以及发送 response 报文，与 161 端口通信的客户端进程则使用临时端口
- 管理器运行客户端进程在 UDP 端口 162 接收来自代理的 trap 报文

8. 网络安全

8.1 网络安全的基本属性

破坏数据机密性、完整性，系统的可用性、可认证性、不可抵赖性、可控性

木桶原理

伴生攻击

有线网络和无线网络中的窃听

8.2 网络攻击与威胁

被动式攻击

- 窃听 Eavesdropping
- 嗅探 Sniffing

主动式攻击

- 网络欺骗 Spoofing
- 干扰与瘫痪 Disruption(DDos)

8.2.1 窃听

8.2.2 嗅探

8.2.3 ARP 协议回顾

ARP 缓存：保存网络中各个电脑的IP地址和MAC地址的对照关系

有以下两种可能的攻击：

- 对路由器ARP表的欺骗
- 对局域网内PC的网关欺骗

8.2.3.1 ARP 欺骗 - “毒化”

目标

- 使ARP缓存中毒，又称“毒化”
- 进行中间人攻击
- 使所有从目标主机传出的流量传输到攻击者的主机

方法

- 伪造ARP回复（伪造IP地址，MAC地址是真实的）
- 对整个网络欺骗

8.2.4 拒绝服务攻击 (Deny of Service)

目标：瘫痪服务乃至整个网络

手段

- 大量的，超过处理能力的请求/流量

效果

- 攻击者：不费劲
- 目标：代价高昂

类型

- 系统缺陷：不对称协议、返回服务器状态或计算结果
- 洪泛攻击：控制大量僵尸机器/网络发送请求

8.2.4.1 网络各层中的拒绝服务攻击

链路层/网络层：对路由器交换机发送大量流量

传输层：请求服务器保持大量连接或者特定状态

应用层：请求大量复杂查询、加密等高开销操作

8.2.4.2 SYN Floods

TCP三次握手中，攻击者伪造大量SYN包发送到服务器端，产生大量半连接（不发生ACK）

不实际的解决方案：

- 增加队列规模（Increase backlog queue size）
- 减少计时器的时长（减小timeout）

可行的解决方案：

- 避免保持状态（用cookie）

8.2.5 放大攻击

原理

- 请求所需的包很少（1个或少量）
- 服务器端响应需要很多包
- 可用于放大Dos攻击效果

攻击方式

- 伪造报文的源地址
- 将源地址设为攻击目标的IP地址
- 服务器将大量响应报文发送给目标

必要条件

- 攻击者需要准备大量的服务请求
- 否则目标将通过简单的丢包化解

8.2.6 MAC Flooding

攻击者对交换机发送很多非法的、包含不同源MAC地址的封包

交换机会把所有这些MAC地址记录到自己的CAM(Content Addressable Memory)表之中。

当这些记录超过交换机所能承载的内存的时候，MAC flooding的效果就达成了。

这时，交换机就变成了集线器，对所有信息进行无定向广播。

8.3 安全机制与手段

8.3.1 密码学

密码原则

Kerckhoffs 原则：即使密码系统的任何细节已为人悉知，只要密钥（key）未泄漏，它也应是安全的。**秘密=key and 秘密≠加密算法**

Shannon提出加密的两个基本方法——扩散和混淆

Encryption and decryption:

- Encryption: $E_k(P) = C$.
- Decryption: $D_k(C) = D_k(E_k(P)) = P$.

From the cryptanalyst's point of view, the cryptanalysis problem has three principal variations:

- ciphertext only, 唯密文
- known plaintext, and 已知明文
- chosen plaintext. 选择明文（最难防）

To achieve security, the cryptographer should be conservative and make sure that the system is unbreakable even if his opponent can encrypt arbitrary amounts of chosen plaintext.

8.3.1.1 Substitution Ciphers 替换密码

8.3.1.2 Transposition Ciphers 换位密码

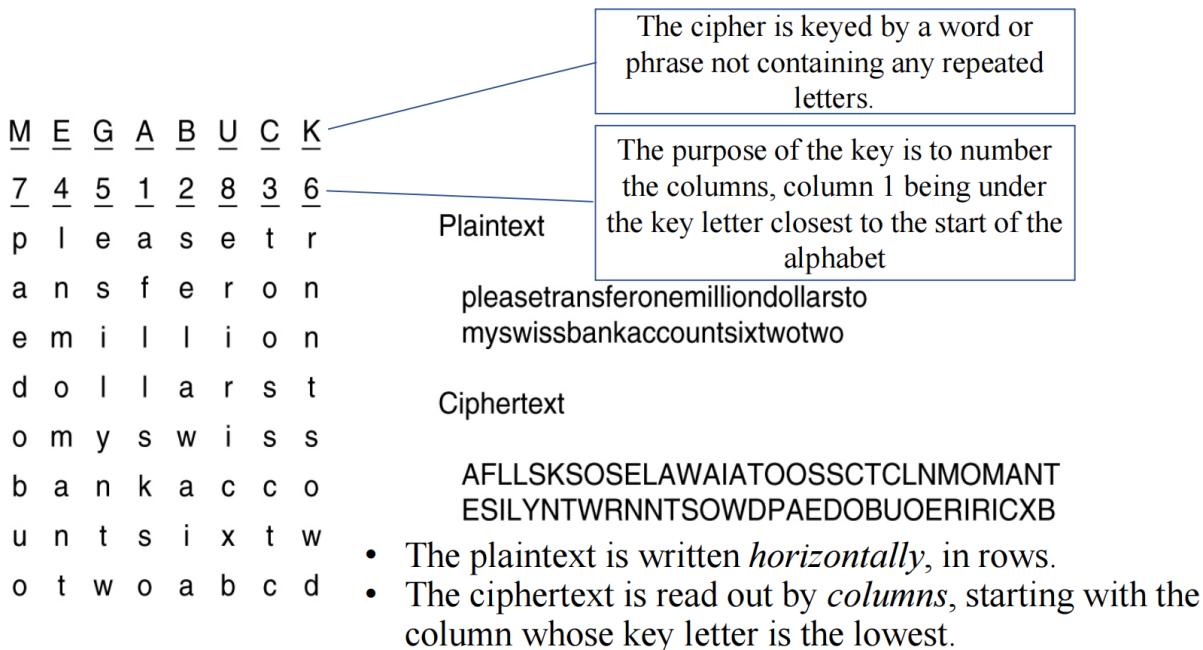


Figure 167

8.3.1.3 One-Time Pads

8.3.2 现代加密算法

8.3.2.1 对称密码学

1. **Feistel结构** —— 对合密码/乘积密码 (重要开端)
2. **DES** (Data Encryption Standard) 算法 (门禁系统) ; **3DES** (银行系统)
 - 分组64bit, 密钥56bit+8bit校验码=64bit
3. IDEA
4. **AES** (当前安全!)
 - 密钥可以是128bit,192bit,256bit
 - AES在当前网络空间是足够安全的, 除非部署未达到标准, 或者是运行模式不合理

8.3.2.2 Cipher Modes

Electronic Code Book Mode (ECB)

对称密码存在的问题

- 加密能力与解密能力是捆绑在一起的
 - 能加密即能解密, 能解密也能加密
 - 不能实现数字签名
- 密钥更换、传递和交换需要可靠信道, 密钥分发困难
 - 如有N用户, 则需要C=N(N-1)/2个密钥, N=1000时, C(1000,2)≈500000
- 密钥管理困难

□无法满足不相识的人之间通信的保密要求

8.3.2.3 非对称密码学

本质上是寻找到陷门单向函数

对称+非对称=混合密码学

□解决了密钥分发/密钥交换难题

□But,存在中间人伪装的攻击风险

□解决方案：对带有数字签名的公钥进行认证

1. RSA公钥密码体制

2. 椭圆曲线密码ECC

3. 密钥分发 —— D-H算法

◦ D-H算法存在中间人攻击风险

8.3.3 数字签名与报文完整性

数字签名=公钥加密的逆应用=私钥运算+公钥验证

接收者可以凭借消息上所附加的某种东西，确认发信人的身份。这种东西，就是签名。

密码散列函数 (HASH)

消息认证码(Message Authentication Code MAC)

- MAC =消息摘要+报文鉴别码
- HMAC (Hash MAC)是一种使用单向散列函数来构造消息认证码的方法
- 消息认证码不能解决的问题：
 - 对第三方证明
 - 防止否认，消息认证码无法防止否认，是因为消息认证码需要在发送者和接收者两者之间共享一个密钥

公钥证书 (Public-Key Certificate, PKC) = 驾照

- 驾照包含：姓名、组织、邮箱、地址等个人信息，以及属于此人的公钥，并由官方认证机构 (Certification Authority, CA) 施加数字签名。
- X.509是一种公钥证书生成和交换的标准规范
- CA签名后不可更改

(a) A hierarchical PKI. (b) A chain of certificates.

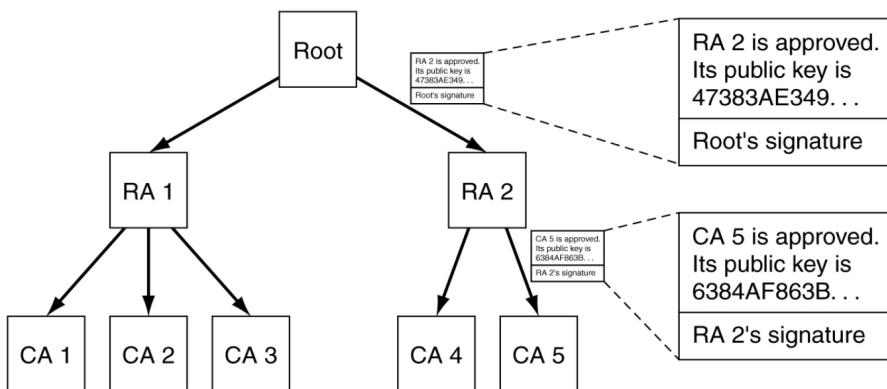


Figure 168

9. PPT 例題

A protocol is an agreement between the _____ on how communication is to proceed.

- 1. A. network layers on the same machine
- B. network nodes
- C. adjacent entities
- D. peers on different nodes

D

2.

3.