

## Chap1 导言

### 1.1 什么是因特网

1. 网络提供的**重要功能**: 连通性, 共享
2. 因特网的**两种定义**: 由一群遵循TCP/IP协议的ISP按照**松散的层次结构**组织而成的**网络的网络**(因特网特点 “**网网**”, 不严格, 统一) 对于通信功能的实现有指导作用: ISP内部实现, 之间互联//是为分布式应用提供通信服务的基础设施(服务接口: 一组用于在网络上发送接收数据的应用编程接口API), 对于服务接口的定义有指导作用: 有序、可靠/不可靠的数据交付服务
3. 具体功能描述  
**终端设备**: 也叫**主机**或**端系统**, 运行网络应用程序  
**通信链路**: 光纤、铜线、无线, 电磁波等; 传输速率常称为带宽。  
**分组交换机**: 转发分组, 路由器和链路层交换机;  
4. 因特网提供的通信服务: 源主机到目的主机的可靠数据交付/尽力而为的数据交付(不可靠)
5. 网络**协议**: 定义通信实体之间交换报文的格式和次序, 及在报文发送、接受受到其他事件后再采取的动作协议, 规定了不同系统的相同层实体之间的通信规则; 终端-终端, 终端-交换设备; 交换设备-交换设备
6. 协议标准: IETF/RFC XXX; 最核心-TCP/IP, 因特网协议统称为TCP/IP协议族
7. **路由**: 分组从发送端到接受终端经过的通信链路以及分组交换机的序列。
8. **ISP**: 由分组交换机和通信链路组成的网络, 为终端提供接入因特网的服务, 每个ISP是自治的, ISP采用多层结构: 本地ISP, 地区ISP, 全球ISP, 因特网交换点

### 1.2 网络边缘

1. 端系统: 与因特网相连的计算机和其他设备
2. **接入网**: 将终端连接到其边缘路由器由物理链路  
住宅接入: 数字用户线DSL(DSL 数据调制解调器和数字信号/Splitter+接入、分离语音和数据(用户侧)/DSLAM+汇聚, 分离多条DSL线路(ISP侧)/L<2.5 Mbps (<1Mbps)下<24Mbps(<10Mbps)/电缆网: 混合光纤同轴电缆HFC, cable modem、Splitter、电缆、光纤、CMTS构成接入网, 下最高30Mbps, 上最高2 Mbps  
企业接入: 以太网(交换机及链路构成接入网) 10Mbps, 100Mbps, 1Gbps, 10Gbps  
无线接入: 无线局域网(几十米内, 基站单位于有线网络11Mbps、54Mbps、450Mbps, 共享的) router, firewall, NAT, wireless access point (54 Mbps), 广域无线接入(蜂窝移动电话, 数万米, 共享, 3G: 最大(静止) 2Mbps, 4G: 下行100Mbps, 上行20Mbps
3. 物理媒体分类: 设备之间通过物理媒体相连, 物理媒体两端各需要一对收/发设备, 同路径-物理媒体可不同, 分为导引型(**双绞线**(铜): 3类10M、5类100M~1Gbps、6类10Gbps, **同轴电缆**、**光纤**: 几十 ~ 几百Gbps, 低误码率, 长距离传输, 抗电磁干扰)/非导引型(电磁波, 无线电) 长距离大电磁波: 蓝牙: 2.4GHz, 10米左右/WiFi: 2.4GHz, 几十米//红外: 室内短距离//陆地微波: 2GHz, 长距离//卫星: 2GHz, 长距离、大范围//可见光: 1-2km, 50Gbps

### 1.3 网络核心 (选读, 转发)

- define: 由路由器和通信链路组成的网状网络。  
任务: 高效、准确地投递分组到目的地//将全球的本地ISP连接在一起//将数据从发送端的边缘路由器, 转发到接收端的边缘路由器//**核心问题**: 数据包网络核心高效转发 (分组传输延迟小网络吞吐量高) //不是边收边发-分组头含有选路、控制信息差控制、流量控制
- 两个重要功能**: 选路 (交换设备确定不同目的地的转发端口, 生成转发表), 转发 (交换设备按照转发表, 将分组移动到相应的输出链路)

- 数据在网络中的传递方式: 电路/分组交换 (计算机网络使用)
1. **电路交换(固定分配)**: 电话网采用电路交换 (本质是预留资源和独占资源) 链路划分成若干条独立的子信道(电路) 通信前预留端到端资源(对比: 分组交换时预留资源) 资源独占: 保证性能(带宽、延迟), 通信静默期, 资源闲置多路复用技术实现共享通信链路: 频分FDM/时分复TDM TDM相对FDM的优点: FDM需要复杂的模拟硬件来将信号转换到合适频带上。

2. **分组交换(按需分配)**: 主机将应用数据划分成分组, 交换机在接受到完整分组后才可以开始转发(存储转发); (交换) 在传输路径上, 交换设备从一条链路上接收分组, 发送到另一条链路上。 [N条链路有N-1台路由器] **端到端传输时延**: P个分组经过N条链路的总耗时: **P\*(N-1)\*L/R**; (否则PL/R)
- 优点**: 适合突发数据流, 简单, 不需建立连接/允许支持更多的用户/轻负载时, 分组交换可以更快地服务用户
- 缺点**: **当大量分组集中到达时, 排队延迟和丢包较严重**

- 排队延迟**: 输出缓存队列, 等待发送
- 队列头延迟**: 丢包率增大队列变长; 排队延迟增大 (间接丢包)
- 丢包**: 若输出链路的缓存满, 溢出分组被丢弃。
- 需要什么可靠传输和拥塞控制的协议?
- 为什么因特网采用分组交换?** 分组交换适合突发流量; 传统因特网应用 (如电子邮件、文件传输) 具有突发通信的特点
4. **分组交换原理**: 存储转发, 数据路由(每个分组自带源地址, 目的地址, 拓扑发现, 路由选择), 出错由端系统处理;

5. 统计复用 (分组交换): 信道使用模式不固定; 同步时分复用 (电路交换): 信道使用模式固定
6. 连接全球ISP-因特网交换点 (IXP), 对等链路接入ISP-地区ISP-第一层ISP/存在点PoP: 低层ISP接高层ISP的多宿 (multi-home): 一个低层ISP可接入多个高层ISP 内容提供商网络
- 1.4 **分组延迟**  
衡量网络性能的主要指标: 延迟 (分组从源端到目的地的时间); 丢包率 (未成功交付到目的终端的分组比例) **吞吐量**: 单位时间内网络成功交付的数据量。  
**R=链路带宽(bps)**: L=分组长度(bits) **R/L**=分组序列化时间  
**a=**分组到达平均速率 **d=**物理链路的长度  
**s=**数据在物理介质上的传播速度

1. 来源: 1节点处理 [检查比特错误, 确定输出链路] 2 排队 [在输出缓存等待传输, 时间长短取决于链路负载大小拥塞程度] 3 传输延迟 [将分组发送到链路上的时间 L/R, 取决于链路速率] 4 传播延迟 [数据在物理链路上的时间 d/s, 取决于链路长度]
  2. **节点延迟**: 接收(us)->排队(差大)+传输(usms)+传播(us) **ms d nobal=d proc+d queue+d trans+d prop**
  3. **流量强度**: **La/R <1** 越接近1, **queue延迟越高**
  4. **端到端延迟**: 分组传输链路上所有节点节点延迟和对端到端延迟敏感的应用: 高度敏感: 实时交互应用, 如网络电话、视频会议; 中度敏感: 在线交互应用, 如网页浏览; 探测端到端延迟: ping; traceout
  5. **Rs**: 服务器与路由器之间的链路速率Rc: 路由器与客户
- 端到端吞吐量**: **min(Rc, Rs, R/10)** 10: 链路使用数  
吞吐量与瓶颈链路速率及链路上的负载有关, 限制因素通常是接入网。
- 延迟、丢包率、吞吐量, 均与负载有关**

### 1.5 协议层次和服务模型

网络协议定义了: 通信实体之间交换的报文的格式和次序//在发

送/接收报文、或其它事件后采取的动作

系统功能将系统功能组织成一系列水平的层次, 每层实现一个功能(服务)每层通过以下方式提供的服务: 在本层内执行一些动作; 依靠下层提供的服务 (层次间关系);

**优点**: 易于处理复杂系统, 层式的层次结构易于确定系统的各个部分及其相互关系; 模块化简化了系统的维护和升级; 对于其它层次透明;**缺点**: 一层可能冗余低层功能;

**协议栈**: 应用层, 运输层, 网络层, 链路层, 物理层

**按功能划分层次**, 每层实现一个功能

**应用层**: 在应用程序之间传输应用特定的**报文**: FTP, SMTP, HTTP

**运输层**: 应用程序-网络接口间(进程-进程)传输**报文段** TCP, UDP

**网络层**: 终端-终端 **分组 (数据报)** / IP协议, 路由选择

**链路层**: 以太网, PPP, 电缆接入网DOCSIS协议/帧/相邻设备

**物理层**: 同轴电缆, 光纤的协议物理媒介/比特

**五层中, 应用层在用户态, 运输层和网络层在内核态**其协议运行在**主机的操作系统中**; **链路层和物理层协议运行在wifi和网卡上** **主机上运行全部五个层次**

**OSI** 开放系统互连模型: 在应用层和传输层间加了表示层和会话层

**表示层**: 使应用层能解释交换数据的含义(压缩、加密、描述)

**会话层**: 数据交换的定界和同步功能, 包括建立检查点和恢复方法的方法

网络功能分布式实现: 某一层上的网络功能, 需要该层上的**实体 (分布在不同的节点)** **协同完成** 协同需要功能实体的之间能够交互信息, 需要解决以下问题: 信息交互的载体: 各层上的报文/信息交互的约定: 报文格式及语义规定/报文的传输方式: 封装和解封装

封装: 路由实现1-3层协议, 链路交换机实现1-2 封装形式: 分组=首部+有效载荷字段(来自上一层的分组)

在不同系统的同一层上: 对等实体执行该层的服务, 相同系统的上下层: 调用服务/提供服务, 封装/解封

**分布不同系统的不同层: 不直接通信**

网络攻击: 恶意软件, 僵尸网络, 自我复制, 病毒, 蠕虫, 拒绝服务攻击 (DoS), 弱点击击, 带宽洪泛, 连接洪泛, 分布式DoS (DDoS), 分组嗅探器, IP欺骗, 重放攻击(嗅探敏感信息, 重新注入网络)

针对因特网基础设施的攻击: 恶意软件 (如病毒、蠕虫) 入侵计算机系统//对主机、网络等实施拒绝服务攻击 (Denial of Service), 使其中止服务

针对因特网中信息的攻击: 窃听网络中传输的数据, 在网络中注入虚假信息欺骗用户

计算机网络关注: 功能性问题: 可达性 (选路、转发) 正确性 (可靠性控制); 性能问题: 吞吐量, 延迟, 丢包率; 安全性问题: 基础设施安全, 信息安全

2. 双方连接内容提供商网络实现连接不是ISP的连接方式
  3. 主机: 应-物 路由器-网-物 交换机: 链-物
  4. 带宽: 单位时间能够传输的最大数据量  
吞吐量: 单位时间能够成功实现的最大传输速率
  5. 进程A发送数据到B: 识别IP地址+端口号
  6. SSL工作应用层上, HTTP1.0一次只发送一个对象
  7. P2P有服务器进程和客户端进程, 每个主机既是服务也是客户
- HTTP 无状态服务, FTP有状态服务。
- 访问百度不会遇到哪个DNS服务器: 百度的本地DNS服务器
- TCP: Aseq=32, 给B发8字节, B回ACK40没收到, 又发20, 收ACK60
- TCP接收窗口再连接持续的过程中可能会变化, A的未确认比特不可能超过缓冲区大小, A发了seq=m, 下一个序号不是m+1, 1. head有接收窗口大小 TimeoutInterval<SampleRTT.

在接收端设置缓存, 缓存使所有到达的分组都经受了延迟, 相应的协议支持分组加上序号-分组的到达可不被干扰, 增加一个时间戳, 区分正常传输和时延

## Chap2 应用层

### 2.1 基本概念

1. **网络应用架构**: (在各个端系统上组织应用程序的方法)
- 客户-服务器架构**: 对等架构P2P
- 服务器**: 永远在线, 运行服务器程序, 具有永久、周知的IP地址, 客户机, 需要时与服务器通信, 动态IP地址。
- 资源集中**: 资源发现简单, 集中式计算带来问题, 如服务器拥挤、压力大、网络流量不均等, 响应延迟长
- 客户之间不通信 客户程序服务器程序+协议=应用**
- P2P**: 每个对等方既可请求服务, 也可提供服务任意端系统可以直接通信, 没有总是运行的服务器, 使用动态IP优点: 高度可伸缩, 缺点: 难于管理 (BT, 迅雷, Skype, PPLive) 资源分散: 易于扩展、均衡网络流量; 资源发现困难, 社会问题 (版权、安全、安全性等)
2. 进程: 主机上运行的程序
- 同一个主机内: 进程间通信机制(OS提供)
- 在不同主机上: 通过交换报文进行通信(主动-客户)
3. **套接字**(门): 进程通过套接字发送和接收报文, 是应用层和传输层的接口&应用程序和网络之间的API
3. 进程描述: **进程标识**包括: 主机地址(IP)与/该进程关联的**端口号** HTTP server: 80; Mail server: 25

4. **因特网提供的传输服务**: 运输层协议如下

**TCP 面向连接的(保证传输顺序)** 可靠传输, 流量控制, 拥塞控制, 不提供及时性, 最低带宽保证, 安全性

**UDP 无连接** 不可靠传输, 不提供顺序保证, 可靠传输,

```
请求行: GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language: fr

首部行:

HTTP/1.1 200 OK 状态行
Connection: close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 ...
Content-Length: 6821 首部行
Content-Type: text/html
data data data data ...
```

流量控制, 拥塞控制, 及时性, 最低带宽保证

**应用层协议**定义了: 交换的报文类型, request, response, 报文语法语义, 发送/响应报文规则

公共域协议Web HTTP(超文本传输协议)电子邮件: SMTP(简单邮件传输协议) 专用协议Skype...

### 2.2 WEB HTTP (应用层资源网页)

1. HTTP协议: 采用**客户-服务器模式**, 定义**浏览器和web服务器**之间的通信规则, 使用**TCP**作为传输层协议, 客户发起到**服务器80端口**的TCP连接(客户端创建一个套接字), 服务器接收TCP连接并创建套接字, 浏览器和服务通过各自的套接字来交换HTTP报文, 其中, 服务器**不保存有关客户请求的任何信息**(对象-就是文件, html文件, jpeg图像, 每个对象通过一个URL访问)
2. 连接方式: RTT-一个小组从客户发送到服务器再返回的时间

**非持久HTTP**: HTTP1.0在一个TCP连接上最多发送一个对象, 接受完响应报文后关闭TCP连接, **获取每个对象均需等待2个RTT**, 每个TCP连接都要消耗操作系统资源, 浏览器通常打开多个TCP连接获取引用对象, 消耗资源, 建立TCP连接用时一个RTT; 发送HTTP请求至收到响应的几个字节字时间一个RTT

**持久HTTP**: HTTP1.1一个TCP连接上发送多个对象, 服务器在发送响应后保持连接, 同一对客户-服务器之间的后续HTTP报文可以在该连接上传输

无流水线方式: 客户仅当收到前一个响应后再发送新的请求-请求每个对象用时1个RTT-请求一个网页用时12RTT [网页包含文本和10个图像]

流水线方式: 客户每解析到一个引用对象就可以发送请求, 可在1RTT时间内请求所有引用对象, 请求一个网页用时约3RTT(网页\*2(建立连接, 发送请求)+图片\*1)

3. 报文格式: 请求行, 首部行, 回车 (表示结束) 实体上传方法: post放在报文体内; get放在URL内

HTTP1.0 GET/POST/HEAD, 1.1 +PUT/DELETE

响应行: 状态码: 200 OK/301 Moved Permanently 400 Bad Request/404 Not Found/505版本不支持

5. **cookie**: 保存状态

存储位置: 服务器端, 信息保存在服务器端的后端数据库, 返回ID给客户/客户端: 浏览器cookie文件, 附在报文中

6. **web缓存**器(代理服务器, proxy) 即是客户端又是服务器, 保存最近请求过的对象的拷贝; 减少客户请求的响应时间, 减少机构接入链路上的流量。

一般由ISP架设, 如当发现缓存对象非最新, 发送含if-modified-since(首部)的报文到原始服务器, 304(not)

**2.3 FTP (文件)** 文件传输协议 TCP端口21、20

通过**FTP用户代理**上传和下载远程文件(客户-服务器) FTP采用两个并行的TCP连接: 传输文件-控制连接(端口21)+数据连接(端口20), 是有状态服务, 21一直保持, 20随文件传输数据而关闭, 每个数据连接一个文件

**分片控制**, 数据连接的原因: 不会混淆数据与命令/相应, 简化协议涉及和实现, 在传输文件的过程中可以继续执行其他操作, 便于控制传输过程。

关闭数据连接的方式: 结束传输, 允许动态创建文件

**2.4 电子邮件系统 TCP**

三部分: 用户代理, 邮件服务器, 邮件件传 (邮件件传用户代理: 1编辑邮件等 e.g. Outlook 2将要发送的邮件发送到用户的邮件服务器 3从用户邮箱中取邮件 邮件服务器: 1用户信箱, 放收到的邮件 2发送邮件队列: 存放要发送出去的邮件 3报文传输代理MTA: 运行在服务器后台的系统守护进程, 负责在邮件服务器之间传输邮件, 及将收到的邮件放入用户信箱。

**电子信箱** (标识用户信箱的str) @ (邮件服务器名字)

(1) 由计算机上的一个存储位置(如磁盘上的一个文件)组成(2) 每个信箱均被分配了唯一的电子邮件地址

**简单邮件传输协议—SMTP** (MIME多因邮扩协议编码)

-邮件服务器之间传输邮件时采用**客户-服务器**模式, -使用**TCP**作为传输层协议, 持久连接, 服务器端口**25**; -发送服务器和接收服务器之间直接传输邮件

**SMTP采用命令/响应交互方式**, 一条连接多个报文命令: ASCII文本, 响应: 状态码和短语

-报文只能包含7位ASCII码) HTTP无此限制

-25端口建立TCP连接-服务器发送就绪报文-客户发送HELO文, 用域名标识自己-服务器250OK-250K-354start mail input-headers(to/from/subject)+\n+content+/-2500K-QUIT-221service closed

-是一个**推**协议, 只能将邮件从用户代理推送到邮件服务器, 不能用SMTP从邮件服务器中获取邮件; HTTP是拉协议; SMTP把所有报文向对象直接放在一个报文中, 中, HTTP把对象封装到相应报文中, **一方请求可向另一方SMTP服务器使用**, "表示报文结束(FTP使用关闭连接表示传输结束, HTTP使用长度域表示报文结束)

-非ASCII内容编码方式: Base64/quoted-printable

**5. 邮件访问协议**: 命令响应POP(110) IMAP(143) // HTTP(80) 允许用户从信箱中提取邮件

**POP3** 认证+授权+事务阶段, 无状态(下载删除/下载保存), **IMAP 有状态**, 所有邮件保存在服务器上, 文件夹

客户用代理为普通浏览器, 文件夹

**区别: 持久/非持久连接** HTTP可以一条TCP传输多个对象, SMTP可传输多个邮件; FTP任一**文件传输结束标识** TCP长度域/FTP关闭连接/SMTPT//内容: SMTP仅ASCII; **交互方式**: HTTP报文(其他命令/响应交互)

**2.5 DNS, 主要UDP, 有时TCP, port53** 因特网目录服务

1. 概念: 由大量按层次组织的DNS服务器实现的分布式数据库, 允许主机查询分布式数据库的**应用层协议**

**DNS是实现在应用层的因特网核心功能**。

**2. DNS 提供的服务**:

-主机名-IP地址转换

-主机别名

-提供拥有复杂主机名的主机具有一个或多个别名

-提供与主机别名对应的服务器的主机名及IP地址

-邮件服务器别名

-提供邮件服务器的规范主机名及IP地址

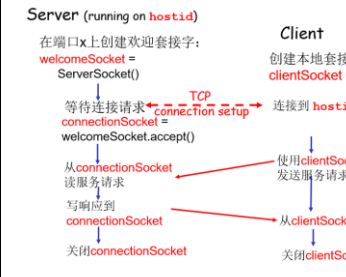
-允许用域名作为邮件服务器别名

-负载均衡

允许一个规范主机名对应一组IP地址

将http请求在一群相同功能的web服务器之间分配

Q: 为什么不用集中式DNS?



(1) 单点失效 (2) 流量集中: 单个DNS服务器需处理全部查询 (3) 响应时间长: 远距离的集中式数据库 (4) 需要维护庞大的数据库

**域名: 域名的任一后缀也是一个域**

每个节点只需保证其孩子节点的标记不重名

**顶级域: 组织域, 国家域, 反向域(ip-域名), inaddr.arpa**

**DNS服务器类型**:

-根服务器13个(知道所有顶级域服务器的IP地址)

-权威域TLD服务器, 每个TLD服务器负责一个顶级域

知道其所有二级子域的域名服务器地址

-权威DNS服务器: 提供机构内服务器(如Web, mail)的主机名-IP地址映射, 提供一个主域名服务器, 一个或多个辅助域名服务器, 可由机构维护, 也可ISP维护

-本地DNS服务器, (本质不属于层次代理的作用)

3. **DNS工作机制**:

1应用程序(如浏览器)调用一个本地例程(解析器), 主机名作为参数传递到解析器, 网络中的DNS服务器发送DNS查询报文(包含要查询的主机名) 3解析器收到包含IP地址的DNS响应报文 4解析器将IP地址返回给调用者(如浏览器)

Q: 客户想知道www.ama.com的IP地址:

1DNS客户查询根服务器, 得到com域DNS服务器地址 2DNS客户查询com域DNS服务器, 得到ama.com域的DNS服务器地址 3DNS客户查询ama.com域的DNS服务器, 得到 www.ama.com的IP地址

域名解析的例子: 迭代查询, 递归查询, [从请求主机到本地DNS服务器的查询是递归的, 其余查询是迭代的.]

### 实际的物理服务器的层次与域名空间的逻辑层次不同

### 4. DNS缓存

每当收到一个响应报文, DNS服务器将报文中的映射信息缓存在本地, 缓存中的映射在一定时间后被丢弃。

**本地DNS服务器通常会缓存TLD服务器的IP地址, 因而很少去访问根服务器**

**5. DNS更准确说法: 存储资源记录(RR)的分布式数据库**

RR format: (name, type, ttl, value)

Type=A Name: 主机名 Value: IP地址

Type=NS Name: 域 (e.g. foo.com) value: 该域的权威DNS服务器的主机名

Type=CNAME Name: 别名 Value: 规范名

Type=MX Name: 域 (e.g. foo.com) Value: 该域的邮件服务器名字

Identification	Flags	
number of questions	number of answer rrs	12 bytes
number of authority RRs	number of additional RRs	
questions (variable number of octets)		
answers (variable number of octets)		
authority (variable number of octets)		
additional information (variable number of octets)		

DNS主要使用UDP, 也使用TCP, 服务器端口53响应报文的长度小于512字节时UDP, 超过512字节时TCP

当解析器事先不知道响应报文的长度时, 先用UDP, 若响应报文的长度超过512字节, 服务器截断这个报文, 置DNS报首部报的TC标志为1, 解析程序打开TCP连接, 并重复这个请求, 以便得到完整的响应。

### 在DNS中插入资源记录

example: new startup "Network Utopia"

向DNS注册机构注册域名 "networkutopia.com"

提供权威DNS服务器(主域名服务器, 辅助域名服务器)的名字和IP地址

对每个权威域名服务器, 注册机构往 com TLD 服务器中插入两条资源记录, 建立权威DNS服务器, 特别是: 建立www.networkutopia.com的Type A记录 建立www.networkutopia.com的Type MX记录

### P2P文件共享

1 **P2P应用架构: 集中式目录** 每个对等方与一个集中式的目录服务器连接(自己的IP地址及IP可以共享的内容)

问题: 单点失效, 目录服务器成为性能瓶颈, 版权问题

2 **P2P应用架构: 查询洪泛**

查询方在已有的TCP连接上发送查询报文, 收到消息的对等方向其邻居转发, 有内容的对等方向QueryHit

3 **P2P应用架构: 层次结构的查询网**

介于1和1之间每个对等方是超级节点/从属于某超级节点的普通节点, 每个普通节点与其超级节点之间建立TCP连接, 某些超级节点之间建立TCP连接

超级节点知道其所有从属节点的共享内容列表

比较 us: 服务器上, 上传带宽, ui/di 第一个终端的上下带宽

C-架构: dcs vs max (NF/us, F/min(ui))

P2P: dP2P=max(F/us, F/min(ui), d(us+sum(ui)))

BitTorrent对等方在下载数据块时, 也向其~ 上载。

PullingChunks: 周期性对等方询问每个邻居拥有的数据块集合(先请从邻居中拷贝数量最少的数据块)

SendingChunks: 选择当前向其发送数据最快的4个邻居, 响应他们请求的数据块

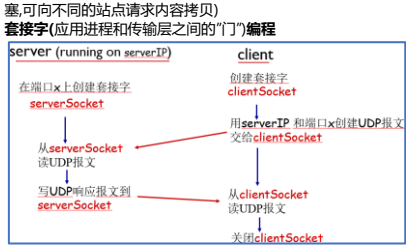
**视频流和流媒体分发网络(CDNs)**

**1流式多媒体: DASH**

服务器(将视频划分成多个块, 每个块以不同码率编码和存储, 元文件为每次请求提供URL) 客户(周期性测量到服务器的带宽, 查询元文件, 每次请求一个块, 当前带宽支持的最大码率, 不同时刻可以选择不同码率的块)

**内容分发网络(CDN)** 在多个CDN站点存储内容的拷贝, 用户从CDN请求内容, 用户请求被定向到附近的站点, 获取内容(网络拥





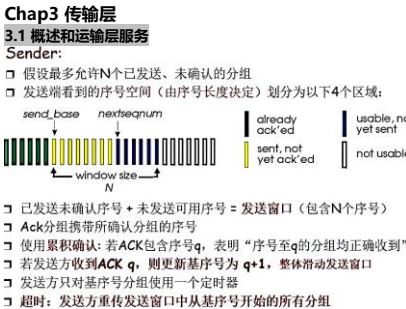
**TCP(一个端口有一个连接在监听->n+1)**

**服务器使用多个套接字服务客户:**

- 1 服务器进程在欢迎套接字上等待客户的连接请求; 客户进程需要通信时,创建与服务器欢迎套接字通信的客户套接字,在此过程中,客户TCP向服务器TCP发送连接请求
- 2 服务器进程创建一临时套接字(称连接套接字)和一个新的服务器进程,与客户进程通信
- 3 服务器进程回到欢迎套接字上继续等待
- 4 允许服务器同时服务多个客户
- 4 客户服务结束后,服务器销毁进程,关闭连接套接字

**UDP&TCP**

- |                                      |                                     |
|--------------------------------------|-------------------------------------|
| <b>UDP</b>                           | <b>TCP</b>                          |
| <b>报文传输服务</b>                        | <b>字节流传输服务</b>                      |
| 由于没有建立管道,应用程序发送每报文必须给出远进程地址          | 由于建立了管道,应用程序只需向套接字中写入字节序列,不需指出远进程地址 |
| 服务器使用一个进程和一个套接字为所有客户服务,一次请求-响应完成一次服务 | 服务器为每个客户单独生成一个套接字和一个新进程,允许双方长时间通信   |



**网络层:**提供主机之间的逻辑通信,尽力而为的服务,网络层尽最大努力在主机间交付分组,但不提供任何承诺-不保证交付,不保证按序交付,不保证数据完整

**传输层:**提供进程之间的逻辑通信,依赖并增强网络层服务,基本服务:将主机间交付扩展到进程间交付

1.传输协议运行在主机系统上:

发送方: 将应用报文封装成报文段,交给网络层发送.

接收方: 从收到的报文段中取出载荷,交给应用层

2.传输层不能提供的服务:延迟保证;带宽保证;

传输层可以提供的服务:保证可靠,按序的交付;TCP;不保证可靠、按序的交付;UDP;

**3.2 多路复用与多路分解(传输层必须提供的服务)**

1发送端多路复用:从多个套接字收集数据交给网络层

2接收端多路分解:将接收到的报文段交付到正确的套接字

3. 如何进行多路复用和多路分解?

为将报文段交付给正确的套接字:

主机中每个套接字应分配一个唯一的标识

报文段中有特殊字段指示要交付的套接字

**“发送方”**传输层需在报文段中包含目的套接字标识(多路复用)

**“接收方”**传输层需将报文段中的目的套接字标识与本地套接字标识进行匹配,将报文段交付到正确的套接字(多路分解)

4.端口号是套接字标识的组成部分,是16比特的数,其中0~1023保留给公共域协议使用

5.UDP套接字标识:(目的IP地址,目的端口号) UDP的源是用来发响应的报文的.TCP套接字标识 (不同标识不同套接字)(源IP源端口号,目的IP的端口号)

**3.3 UDP(流媒体,DNS,SNMP)**

1. 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

2.UDP需要实现的功能:多路复用与解复用,报文检错

3.UDP不提供的服务:可靠/按序交付;延迟及带宽保证;【若要UDP实现可靠传输,在应用层实现可靠性】

4.UDP 报文结构:报头+载荷,报头如下:

用于多路复用/多路分解的字段:源端口号,目的端口号;用于检测报头错误校验的字段:报文段长度,检核和;

5.UDP检查和/所有 16 位比特字相加,若溢出,则相加和+1,然后取反,接收方计算检核和,然后与checksum相加,若所有比特位都是1,正确,否则有错.

6. 适合应用:容忍丢包但对延迟敏感的应用;如流媒体

【单次请求/响应为主的应用如DNS

**3.4 可靠数据传输(RDT)原理** 有限状态机 (FSM)

数据通过可靠信道传输,没有损坏和丢失,且按序接收.(有序无损)

Rdt1.0: 底层信道完全可靠,只需发送和接收即可. 1-1

Rdt2.0: 下层信道可能产生比特错误 **错误检测,接收方反馈** ACK/NAK,发送方发送后等待反馈,NAK重传 3-2

Rdt2.1:ACK/NAK 出错:发送方发现受损就重传,但接收方会出现冗余,发送方给分组添加序号[0,1],接收方发现序号相同就丢弃,重新发送ACK(ACK不带序号,但带checksum) 2-2

Rdt2.2:不用NAK,接收方收到出错的分组/非期待分组,重发最近一次ACK(带序号);发送方:若ACK的序号非期待(表明当前分组未被确认),重发当前分组.非期待重发(4-2)

Rdt3.0:可能丢包,检测丢包与恢复; 启动定时器,在超时前没收到期待的ACK,重发.受到非期待ACK不处理是停等协议,效率低,需要流水线机制.仅超时重传

**传输时延(发送包用的时间)=t=(L/R)\*(N流水线)**

**信道占用时间 RTT + L/R**

**发送方利用率:U=t/(RTT+L/R)**

**GBN:**发送方只对基序号分组使用定时器,若ACK 不对,不管,仅超时重传整个发送窗口 (确认更新).接收端不缓存失序分组,非期待重发,收到4,5, ACK=5

**SR:**自动调整窗口,避免不必要的重传.

发送窗口:包含已发送未确认和未发送可用的序号,可能有已发送已确认的序号交织其中,基序号是已发送未确认或未发送可用的序号.

发送方收到的ACK不是最小未确认分组 (基序号),则标记.若是,滑动窗口.

接收方一旦收到就ACK,失序就缓存.若收到窗口前冗余分组,说明发送方出错,发送ACK(n), 收方窗口大小限制在序号空间的一半以内N<m/2

接收窗口:包含期待但未收到和允许接收的序号可能有已确认已缓存的序号在其中基序号为期待但未收到或允许接收的序号,收到基序号分组时,按顺序交付分组

滑动接收窗口

发送方:从上层接收数据;若发送窗口不满,发送分组,启动定时器 n 超时:重传分组n,重启定时器

收到发送窗口内的ACK(n),标记分组n为已确认,若n=基序号, 滑动发送窗口,使基序号=最小未确认的序号或下一个序号//其余情形:忽略.

接收方:收到接收窗口内的分组n:发送ACK(n);若为失序分组: 缓存该分组;若n=基序号交付从n开始的若干连续分组.滑动接收窗口,使基序号=下一个期待接收的序号; 若收到[rcvbase-N,rcvbase-1]内的分组n: 发送ACK(n) //其余情形:忽略

**发送端滑动窗口的序号数目 + 接受端滑动窗口的序号数目 <= 分组编号数目**

分组重排、序号重引用: 存活时间+长序号

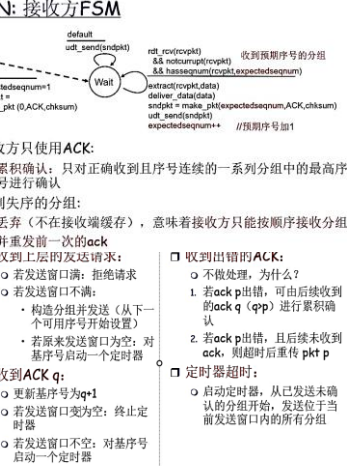
**3.5 TCP 协议**

服务模型: 点到点,全双工,面向连接,可靠有序字节流

需要机制: 建立连接流水式发送报文(可靠)流量控制

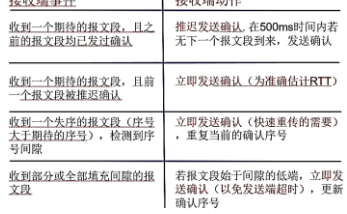
**2. TCP报文段结构:**

源2,目的端口号2;序号:首字节在字节流中的序号,非报文段序号(seq4,确认号:希望从对方接收的下一字节序号,隐含累积确认(ack)4;首部长度:32bits为单位的首部长度(head len);接收窗口2;检验和2;紧急数据指针(指向最后一个字节Urg data pointer);数据段(data



多路复用/分解:

标志位:URG:紧急数据 PSH:立即交给上层 RST:不接受连接 SYN:建立连接 FIN 拆除连接



接收窗口:接收端还可以接受的字节数

重要TCP选项:MSS:TCP段可以携带的最大数据字节数(缺省=536Byte);window scale:窗口比例因子,实际接收窗口大小=window size\*2^window scale; SACK: 选择确认,在累积确认的基础上允许接收端指出缺失的数据字节

5. 往返时间估计

估计平均RTT 指数加权移动平均  $a = 0.125$

Estimated RTT =  $(1-a) * Est + RTT + a * SampleRTT$

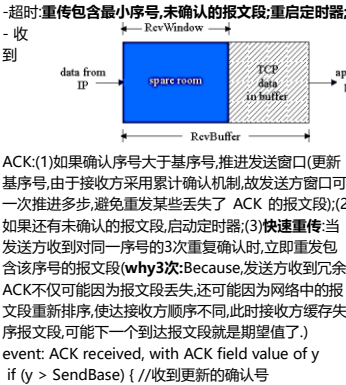
安全距离  $\beta = 0.25 DevRTT + (1-\beta) * DevRTT + \beta * [SampleRTT - EsRTT]$

超时值: TimeoutInterval = 1s

TimeoutInterval = EstimatedRTT + 4\*DevRTT

6. TCP 发送方处理的事件:

收到应用数据:(1)创建,发送TCP报文段;(2)若当前没有定时器运行(没有已发送未确认的报文段),启动定时器;



ACK:(1)如果确认序号大于基序号,推进发送窗口(更新基序号,由于接收方采用累计确认机制,故发送窗口可一次推进多步,避免重发某些丢失了 ACK 的报文段);(2)如果还有未确认的报文段,启动定时器;(3)快速重传:当发送方收到对同一序号的3次重复确认时,立即重发包含该序号的报文段(why3次?Because,发送方收到冗余ACK不仅可能因为报文段丢失,还可能因为网络中的报文段重新排序,使达接收方顺序不同,此时接收方缓存失序报文段,可能下一个到达报文段就是期望值了.)

event: ACK received, with ACK field value of y

if (y > SendBase) { //收到更新的确认号

SendBase = y;

if (there are currently not-yet-acknowledged segments) start timer }

else { //收到重复的ACK

increment count of dup ACKs received for y

if (count of dup ACKs received for y = 3) {

resend segment with sequence number y }

**定时器补偿 - 发送方每重传一个报文段,超时值就增大一倍.** 若连续发生超时事件,超时值呈指数增长(规定的上限值).Karn算法& EstimateRTT & 定时器补偿算出超时 超时重传+快速重传

**7.接收端处理的事情 非期待重发**

**推迟发送ACK:**500ms 每隔一个正常确认

- 优点:可以减少通信量;

- 缺点:当延迟太大时,会导致不必要的重传

推迟确认造成RTT估计不准确.TCP协议规定:

**Go-Back-N TCP SR**

接收方: 接收方: 接收方:

使用累积确认 使用累积确认 缓存失序的分组

不缓存失序的分组 缓存失序的报文 每个分组使用一个定时器

对失序分组发 对失序报文段发 单独确认每个正重复ACK 重重复ACK 确认到的分组

发送方: 发送方: 发送方:

超时后重传从基序号开始的 未确认的报文段 被确认的分组

所有分组 增加了快速重传

推迟确认的时间最多为500ms,接收方至少每隔一个报文段使用正常方式进行确认;

**6. TCP 流量控制:**调节发送速度,使 接收缓存不会溢出;接收缓存中的可用空间= RcvWindow //接收窗口 = RcvBuffer-[LastByteRcvd - LastByteRead]

接收方将RcvWindow放在报文段中,向发送方通告接收缓存的可用空间; LastByteRcvd >= SendBase-1;

发送方限制未确认的字节数不超过接收窗口的大小,即LastByteSent-LastByteAcked <= RcvWindow

接收窗口为0时,发送方必须停止发送.此时发送方启动定时器,超时后发送零窗口报文段(seq为上一段最后一个字节),从接收方的响应中获知窗口大小,仍0, 重定时糊涂窗口综合症:双方处理速度严重失衡;接收方不断发送微小窗口通告,引起发送方不断发送很小的数据分组,导致带宽浪费.接收方:一旦窗口大小显著增加(达到min(带宽空间(一半,MSS))之后才发送更新的窗口

当窗口大小不满足以上策略时,推迟发送确认(但最多推迟500ms,且至少每隔一个报文段使用正常方式确认),仅当窗口大小满足以上时,再通告新的窗口大小;

**发送方:Nagle算法**连接后,数据到来时组成一个TCP段发送.收到确认后,后到来的数据放在发送缓存,当数据量达一个MSS或上次传输确认到来,将缓存中数据发适应网络延迟, MSS长度及应用速度的各种组合/常规情况下不会降低网络的吞吐量

**连接管理:**发送方和接收方在交换数据前先握手.

- 两次握手的不可行性:一个不可靠网络中存在许多干扰连接正常存在的因素:包传输延迟变化很大,存在重传的报文段,存在报文重排;

- 三次握手建立连接:

Step1)不包含数据,2)不包含数据(服务器端分配缓存和变量)3)可能包含数据(客户端分配缓存和变量)

- 四次挥手 closing a connection:

客户端:向服务器发送 FIN,等待服务器确认;

服务器:向客户端发送 ACK,确认请求;

客户端:向服务器发送 FIN,等待客户端确认; (可合并)

客户端:向服务器发送 ACK,等待一段时间结束后.

**Problems:**服务器收到SYN段后,发送SYNACK,分配资源,若未收到ACK段,服务器超时后重发该段.服务器等待一段时间(称SYN超时)后丢弃未完成连接.SYN超时类型为30~120s-SYN洪泛攻击:攻击者用伪造的源IP向服务器发送大量SYN段,不发送ACK段,服务器为维护巨大的半连接表耗尽资源,无法处理正常客户连接扫描-FIN扫描(ACK=1, RST=1/no rep)

**7.拥塞控制原理:** 网络辅助TCP ECN, ATM, DeCBT

流量控制:限制发送速度,使不超过接收端处理能力;拥塞控制(多个源):限制发送速度, <= 网络的处理能力;拥塞造成(用来判断拥塞):1丢包(路由缓存溢出);2分组延迟增大(路由来不及转发,等待转发分组多);3拥塞指数:分组延迟大,网络负载重,但网络吞吐量很低-TCP采用从端到端的拥塞控制机制,发送方根据自己感知的网络拥塞程度,限制其发送速度.

**发送方感知到拥塞后,改变其发送速率的算法是什么?**

发送方使用拥塞窗口CongWin>= (限制)已发送未确认的数据量,其间按线性方式发送速率.

大概的:Rate = CongWin/RTT bytes/sec

TCP拥塞窗口的调节(拥塞控制):加性增,乘性减(AIMD)慢启动:每个RTT,CW加倍,(收到1ACK,CW增加1MSS)

发送方维护变量 Threshold

超过阈值,慢启动状态转为拥塞控制

发生丢包时,Threshold = CongWin/2 转为快速恢复

- 收到3个冗余ACK:说明网络仍然有一定交付能力

**Threshold = CongWin/2**

Reno快速恢复 CongWin = Threshold + 3MSS;

- 超时: 网络的交付能力很差

**Threshold=CongWin/2**

CongWin = 1MSS(发送速率降至最低)

无条件到慢启动,增大CW, CW<threshold,每次加倍;否则只加 1MSS(拥塞避免:缓慢增大CW).

**8. TCP 吞吐量CW/RTT,丢包后= CW/2RTT(平均3/4)**

吞吐量与丢包率L的关系: **[1.22MSS / (RTT 根号 L)]**

TCP的公平性:如果K条TCP连接共享某条带宽为R的瓶颈链路,每条连接具有平均速率R/K.

**1. 面向连接服务的优点缺点?**

优:可靠/有序传输;资源预置(使用)缺:需要全局信息

**2. 无连接服务的优点与缺点?**

优点:无需知道网络状态(包括网络资源)或只需知道局部网络状态;缺点:具有不确定性(是否有满足服务的网络资源不确定,能否完成服务不确定)

**3. 分层网络体系结构的不足:**

上层协议的性能依赖于下层协议

**4. 分组交换原理:**

1存储转发;2动态路由(包括每个分组自带源地址、目的地址,拓朴发现、路由选择);3输出端由端系统处理

**5. 若一个WWW文档中除有文字外,还有7个图像.试问使用单个http/1.0,4个并行http/1.0与单个http/1.1各需要建立几次TCP连接?** (1);8; (3); (3); (1)

6.假定要传送的报文共有L(单位bit),从源节点到目的节点共有k跳链路,每条链路的传播时延为d(单位s),链路带宽为b(单位bit/s);电路交换(包括连接建立与拆除)使用的控制帧(或信号)长度,在各节点的排队时延忽略不计;分组交换使用的帧头长度d,分组长度分别为h、p(单位bit),分组在各节点的排队时延q(单位s).试分析在何种条件下电路交换的总时延要小于分组交换的总时延?

电路交换总时延D(c):

连接建立时间: kd;连接拆除时间: kd

数据传输时间: x/b;数据传输时间: kd

D(c)=3kd+x/b

-分组交换总时延D(p):

-单个分组传输时间: (p+h)/b

-第1跳传输时间: (x/p)/(p+h)/b (x/p为分组个数)

-传输时间每1跳增加1个分组的传输时间到总的传输-时间为x/p\*(p+h)/b+(k-1)\*(p+h)/b

-排队时间: kq;传播时间: kd

D(p)=x/p\*(p+h)/b+(k-1)\*(p+h)/b+kd+kq

若D(c)<D(p),则电路交换总时延小于分组交换总时延

1.停等,GBN,选择重传三个滑动窗口协议有关问题:

答: (1)三个协议的发送窗口、接收窗口大小分别是多少? (2)发送窗口最大有效值由什么因素决定?

答: (1)1; 1/N; 1; 1; N; N

(2)网络缓冲能力(或RTT\*瓶颈节点带宽,即BDP)

**2. 实现TCP连接目标的主要机制.**

(1)通过传输层地址(端口号)实现进程间通信(2)通过确认机制实现可靠传送(3)通过接收方缓存实现按序传送(4)流量控制(5)拥塞控制(6)连接建立与拆除机制

3.在TCP连接中,客户端的初始号215,客户打开连接,只发送一个携带有300字节数据的报文段,然后关闭连接下面从客户端发送的各个报文段的序号分别是多少?

SYN报段215,数据报段216,FIN报段5164.

在一条新建TCP连接上发送一个长度为40KB的文件,发送端每次发送一个最大长度的报文(MSS),MSS=1KB,接收端正确收到一个TCP段后立即给予确认,发送端的初始拥塞窗口T限设置为16KB.假设发送端尽可能快地传输数据,但只要发送窗口允许,发送端就发送一个MSS.

(1)发生第一次超时后,发送端将拥塞窗口门限调整为4KB.请问发生超时的时,发送端的拥塞窗口是多大? 发送端共发送了多少数据,有多少数据被成功确认了?

(2)发送端从未被确认的数据开始使用慢启动进行重传.假设此后未再发生超时,当文件全部发送完毕时,发送端的拥塞窗口是多大?

答: (1)发送端拥塞窗口大小 = 4KB\*2 = 8KB

新建的TCP连接上,采用慢启动开始发送,当第一次超时发生时,已发送=1KB+2KB+4KB+8KB=15KB.

此时,最后等一批8个TCP段未获确认,之前发送的TCP段都被确认,因此成功确认的数据量为7KB.

(2)此问题是在(1)问题的基础开始的,即已确认过7KB,在未发送与未发送有33KB,发送端采用慢启动重新开始发送,在拥塞窗口达到4KB时发送数据量=1KB+2KB+4KB=7KB.然后进入拥塞避免阶段: 在收到全部4个MSS的确认后,拥塞窗口增至5KB,相应地发送端发送了5KB数据;收到全部5个MSS的确认后,拥塞窗口增至6KB,相应地发送端发送了6KB数据;收到全部6个MSS的确认后,拥塞窗口增至7KB,相应地发送端发送了7KB数据;收到全部7个MSS的确认后,拥塞窗口增至8KB,相应地发送端发送了8KB数据;此时刚好发完. 因此,文件发送结束时,发送端的拥塞窗口大小为8KB.

**5. TCP如何发送紧急数据?**

1)紧急标志位U(URG)置1;(2)紧急数据置于TCP段数据(载荷)前部;(3)紧急指针指向紧急数据的最后一个字节.

**6. TCP接收方向何种情形需要立即进行确认?**

答:(1)连续两个段按序到达,且前一个未确认;(2)收到失序段(序号比期望的序号大);(3)收到丢失分段;

**7. TCP协议中 ACK 的作用.**

(1)建立连接,拆除连接(2)差错控制(或可靠传送)

(3)流量控制(4)拥塞控制

一个基于重传的可靠传输协议通常包含以下要素:差错编码,确认,重传,定时器,分组序号,为什么需要这些要素

差错编码: 差错编码提供了一种发现传输错误的机制

确认:确认提供了反馈机制,使发送方可以得知接收方的接收情况;重传:重传提供了修正错误的方,即用新的正确的替换替换错误的;定时器:定时器提供了发现/定义超时时间的方法,避免死锁无限等待;分组序号/分组序号提供了区分不同分组的方式,流水线操作不

会引起混乱。

以下是DNS数据库的一些片段，请回答以下问题：

(1) abc.com的邮件服务器的别名是：mail.abc.com

(2) abc.com的权威域名服务器的IP地址是：202.68.69.1

(3) abc.com的web服务器的规范名是：venus.abc.com

(4) abc.com的FTP服务器的IP地址是：202.68.69.3

(5) 邮箱bob@mail.abc.com所在主机的IP地址是：

202.68.69.7

Name	type	value
abc.com.	NS	dns.abc.com
dns.abc.com.	A	202.68.69.1
abc.com.	MX	mail.abc.com
jupiter.abc.com.	A	202.68.69.3
mars.abc.com.	A	202.68.69.7
venus.abc.com.	A	202.68.69.11
mail.abc.com.	CNAME	mars.abc.com
www.abc.com.	CNAME	venus.abc.com
ftp.abc.com.	CNAME	jupiter.abc.com

当分组到达交换设备时，若输出链路的缓冲队列满，发生丢包。