网原期末复习

ch7 Network Layer 路由选择和网络层

7.1 网络层概述

- ISO定义：网络层为一个网络连接的两个传送实体间**交换网络服务数据单元**提供功能和规程的方法，它使**传送实体独立于路由选择和交换**的方式。
- 位置：通信子网的最高层，端到端传输的最底层
- 功能：屏蔽各种不同类型网络之间的差异，实现 互连 ；了解通信子网的拓扑结构，选择 路由 ，实现报文的网络传输。
- 为传输层提供服务：
  - 面向连接服务（传统电信观点：通信子网应该提供可靠、面向连接的服务），将复杂功能放在网络层
  - 无连接服务（互联网观点：通信子网无论怎么设计都不可靠，只需无连接），将复杂功能放在传输层

> chap5 数据链路层为网络层提供的服务：无确认无连接/有确认无连接/有确认有连接

- 网络层的内部结构
  - **数据报子网**：数据报分组交换，每个分组被独立转发，分组带有全网唯一地址 eg IP
  - **虚电路子网**：虚电路分组交换，先建立虚电路，按次序存储转发分组，最后拆除 eg ATM
    - 虚电路子网与数据报子网的比较
      - **带宽与状态的权衡**
        - **数据报子网中，每个数据报都携带完整的目的/源地址，开销大，浪费带宽** IPv4是32位，IPv6是128位
        - **虚电路子网中，路由器需要维护虚电路的状态信息；** 开销小，但需要维护端到端的连接状态（N个节点→N^2，扩展性存在问题）
      - **地址查找时间与连接建立时间的权衡**
        - **数据报子网对每个分组的路由查找过程复杂** 最长前缀匹配
        - **虚电路子网需要在建立连接时花费较长的路由查找时间**
      - **可靠性与服务质量的权衡**
        - **数据报不太容易保证服务质量QoS(Quality of Service)，但是对于通信线路的故障，适应性很强** 有竞争，不容易做资源的预留；但健壮性强
        - **虚电路方式很容易保证服务质量，适用于实时操作，但比较脆弱**
      - **可扩展性**
        - **数据报子网具有更好的可扩展性**

7.2 路由算法（是网络层协议的一部分）

- 应有特性：正确性、简单性、健壮性、稳定性、公平性、最优性、可扩展性
- 分类：静态（非自适应）算法vs动态（自适应）算法
- **最优化原则（optimality principle）**：如果路由器 J 在路由器 I 到 K 的最优路由上，那么从J到 K 的最优路由会落在同一路由上。
- （def）**汇集树（sink tree）**：从所有源结点到一个目的结点的最优路由的集合
- 路由算法的 目的 是 找出并使用汇集树 ！
- 最短路径路由算法 shortest path routing：Dijkstra

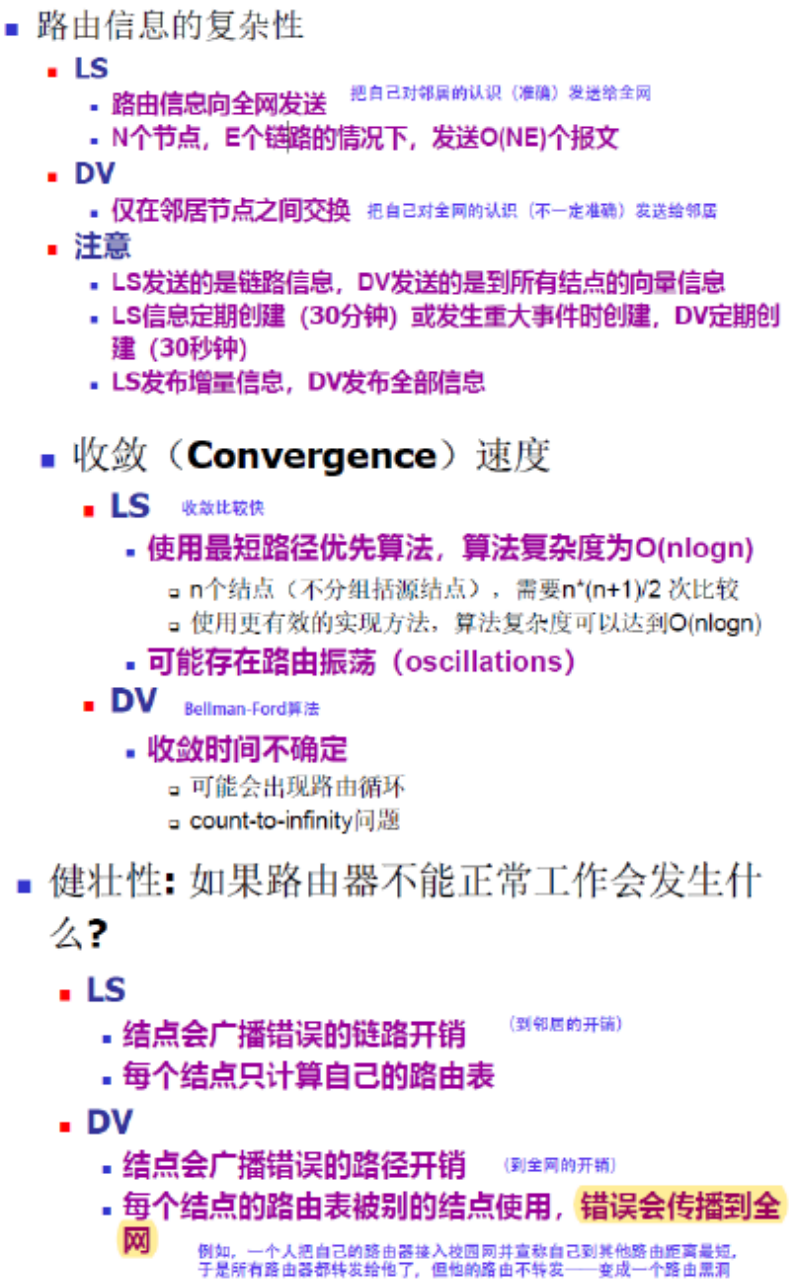
- 测量路径长度方法：节点数量、信道带宽、地理距离、传输延迟、前面的加权

路由算法

1. **洪泛算法flooding：静态**
   - 除了来的地方都发送，可能产生回路->分组头含站点计数器Hoplimit，每次-1，到0丢弃；或者记录路径
   - **选择性洪泛**：仅发送到与正确方向接近的线路
   - 特点：浪费资源，健壮性极好可用于军事应用，作为baseline
2. **基于流量的路由算法flow-based routing：静态**
   - 既考虑拓扑结构，又兼顾网络负载
   - 前提：每对结点间平均数据流相对稳定可预测 （针对大的稳定流量！）
   - 需要的信息：网络拓扑，通信量矩阵，带宽矩阵
   - 提前离线计算，可用于 流量工程
3. **距离向量路由算法distance vector routing（DV/Bellman-Ford/Ford-Fulkerson）：动态**
   - 最初用于 ARPANET ，被 RIP 协议采用
   - 算法：每个路由器向邻居发送自己到所有路由器的距离表，本路由器到i的距离利用邻居路由器X到i的距离Xi、自己到X的距离m更新，为min(Xi + m)；本路由器的老路由表不参与计算！！
   - 缺点
     - 无穷计算 ，对好消息反应迅速，对坏消息反应迟钝
       - 改进：水平分裂算法（从X学到的不告诉X），有时候会失败
     - 没有考虑链路带宽，路由收敛速度慢，路由报文开销大（不是增量更新），不适用大规模网络（RIP最多15跳）
4. **链路状态路由算法link state routing（LS）：动态**
   - 用于OSPF，IS-IS协议（？？？？）
   - 算法
     - 发现邻居结点，并学习它们的网络地址：路由器启动后，通过发送HELLO分组发现邻居结点。两个或多个路由器连在同一个LAN时，引入人工结点来代表他们（称为 代表路由器 DR, Designated Router）：原来共享介质里两两全连通，现在大家都通过DR交互
     - 测量每个邻居节点的延迟或开销（用ECHO分组的往返时间/2，或者根据带宽）
     - 将学习到的邻居信息封装成分组（链路状态声明LSA）
       - 分组以发送方的标识符开头，后面是 序号、年龄 和一个邻居结点列表（对应到它们的延迟）
         - **问题1 序号循环会混乱：**32位够用
         - **问题2 序号出错和路由器崩溃后序号重置：**年龄每秒减1，到0后丢弃
       - 链路状态分组定期创建或发生重大事件时创建
     - 将这个分组发送给所有其它路由器（**不只是邻居**）（发布**增量**信息）
       - 控制洪泛：每次发送序号+1，接收方记录（src, seq）对，若分组重复或过时则丢弃
       - 分组到达后延迟一段时间，多接收一些分组，把来源相同的进行比较，丢弃重复的，保留新的（抑制抖动）

        - 对分组进行应答，包含ack信息和send信息，指示这个路由器学会后应该向谁回复以及再教给谁
    - 每个路由器中根据完整的拓扑，使用dijkstra算法计算最短路径
    - 缺点：路由振荡，如link cost = amount of carried traffic

- LS与DV比较



    - 【DV有但LS没有的优点】（因此DV现在还在使用）
      DV的开销计算尺度是一致的（可以把带宽、延迟等因素加权计算出统一的），但LS可能各个路由算各自的，互连时难以度量。因此在不同国家/运营商之间（往往不便透露路由策略）网络互连使用DV（距离向量）算法。
    - 相应使用的协议：
        - DV：RIP

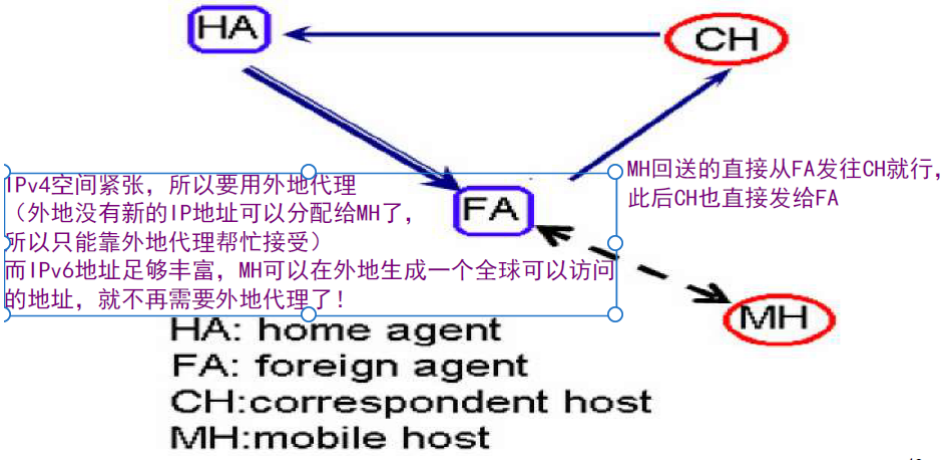
          > 一个类似的版本是PV (path vector)路径向量算法，被BGP采用（域外路由协议EGP）

        - LS：OSPF、IS-IS（都是域内路由协议IGP）

分层路由

- 分成若干域domain，但不一定得到最优路由
- 【计算】sigma(每层的组数-1)+自身所在组的路由器总数。即自身在路由表中，但自身所在组不在。

移动主机的路由

- 移动用户
- 家乡位置（每个移动用户都有一个永久的家乡位置，用地址来标识）
- 家乡代理（每个区域一个，记录不在家的移动用户）
- 外部代理（每个区域若干个，记录正在访问该区域的移动用户）
  - 移动用户进入一个新区域时，必须首先向外部代理注册
    - 外部代理定期广播声明自己的存在和地址的分组，新到达的移动主机接收该信息；若移动用户未能收到该信息，则移动主机广播分组，询问外部代理的地址
    - 移动主机向外部代理注册，告知其家乡地址、目前的数据链路层地址和一些安全信息
    - 外部代理与移动主机的家乡代理联系，告知移动主机的目前位置、自己的网络地址和一些安全信息
    - 家乡代理检查安全信息，通过，则给外部代理确认
    - 外部代理收到确认后，在登记表中加入一项，并通知移动主机注册成功
- 过程：三角路由
  1. 移动用户向外部代理注册
  2. 外部代理与移动主机的家乡代理联系
  3. 当一个分组发给移动用户时，首先被转发到用户的家乡局域网
  4. 被家乡代理接收，家乡代理查询移动用户的新位置和与其对应的外部代理的地址
  5. 家乡代理采用隧道技术，将收到的分组作为净负荷封装到一个新分组中，发给外部代理
     a) 可选：家乡代理告诉发送方，后续分组直接发给外部代理
  6. 外部代理收到分组后，将净负荷封装成数据链路帧发给移动用户
- 

7.3 拥塞控制算法

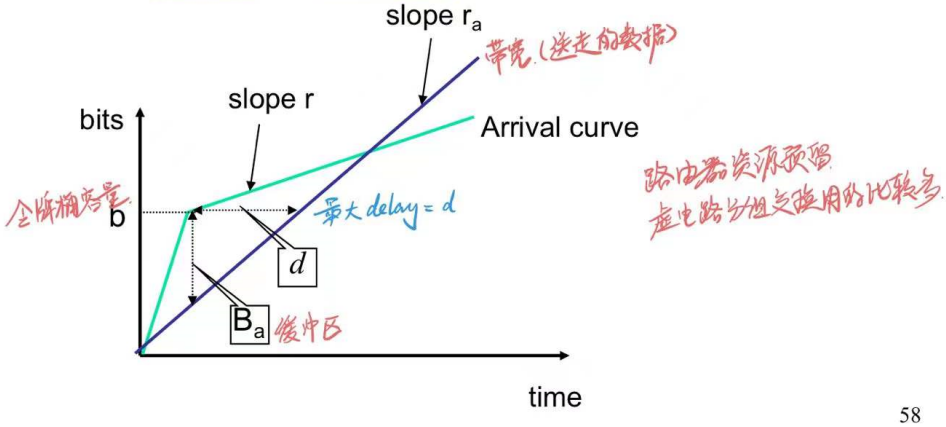
- 网络上有太多的分组时，性能会下降。主要在网络层和传输层（TCP）进行控制。
- 原因
  - 设备问题 —— 网络设备处理器性能低
  - I/O问题 —— 高速端口输入，低速端口输出；多个输入对应一个输出
- 拥塞控制 vs. 流控制的差别：
  - 拥塞控制是 全局 问题，解决通信子网和数据流问题（涉及主机、路由器等）

    - 流控制是 局部 的点到点传输问题，一般基于反馈控制，解决快发慢收问题（例如滑动窗口）
- 拥塞控制基本方法：
    - 开环控制：提前设计，不考虑网络某次运行时的当前状态
    - 闭环控制：基于反馈，监控拥塞-传输消息-调整解决
- 影响拥塞的网络设计策略（Policies that affect congestion，P50）：
    - 传输层：重传、缓存、确认（ack）、流控、超时等机制
    - 网络层：虚电路 / 数据报子网的设计等
    - 数据链路层：重传、缓存、确认（ack）、流控等机制
- 流量整形 traffic shaping： 开环控制
    - 基本思想：拥塞主要原因是流量通常是突发性的->强迫分组以可预测的速率发送
    - （algo）漏桶算法：漏桶存放数据分组（满了就丢**数据**），转变成平滑的数据分组流
        - 可用于ATM（组为单位）或IP（字节计数）
        - （prop）不灵活
    - （algo）令牌桶算法：漏桶存放令牌（固定间隔产生，满了就丢**令牌**），分组传输之前必须获得一个令牌，传输完成后删除令牌
        - 允许空闲主机积累发送权（最大为桶的大小）
        - （以上两种都可用于①固定分组长的协议如ATM；②可变分组长的协议如IP）
        - 利用令牌桶实现逐跳资源预留

          ■ Given (b,r,R) and per-hop delay d

          ■ Allocate bandwidth r_a and buffer space B_a such that to guarantee d

          ■

          58

虚电路子网中的拥塞控制

- 流说明：描述 发送数据流的模式 和希望得到的 服务质量 的数据结构
    - 子网和接收方可以做出三种回复：同意、拒绝、其它建议
- 准入控制：根据流说明和网络资源分配情况，进行准入控制
    - 可以在解决拥塞前不允许建立虚电路，也可以绕开拥塞地区
- 预留资源：子网根据协议在虚电路上为连接预留资源

数据报子网中的拥塞控制

- 抑制分组
  - 如一个分组的输出线路处于警戒状态，则向源主机发送抑制分组（指出拥塞的目的地址、将原分组打上标记说已经产生过抑制分组了），正常转发
  - 源主机收到抑制分组后，按比例减少发送到特定目的地的流量，并在固定时间间隔内忽略指示同一目的地的抑制分组。监听若线路仍然拥塞，继续循环；否则以常量增加（AIMD）
- 逐跳抑制分组：高速、长距离网络中，源主机相应太慢，前一种效果不好
  - 抑制分组对经过的每个路由器都起作用
  - 能迅速缓解拥塞，但上游路由器要有更多的缓冲区

其他?

- 公平队列算法：路由器的每个输出线路有多个队列，路由器循环扫描各个队列，发送队头的分组。对于变长分组可以改成按字节轮询。加权公平队列：优先级高的队列在一个轮询周期内获得更多的时间片。
  - 按照每个包的到达时间（和该队列上个包的完成时间）、长度、权值算出这个包的完成时间，按照完成时间的从小到大依次输出。用WFQ算法近似。
- 负载丢弃算法：文件传输丢弃新的，wine策略；多媒体服务丢弃旧的，milk策略。随机早期丢包。

7.4 网络互联

- （def）互连网络（internet）：两个或多个网络构成互连网络

网络互联设备

- 中继器（repeater）：物理层，在电缆段之间拷贝比特，对弱信号进行放大/再生以延长传输距离
- 网桥（bridge）：数据链路层，在局域网之间存储转发帧，网桥可以改变帧格式
- 多协议路由器（multiprotocol router）：网络层，在网络之间存储转发分组。必要时，做网络层协议转换
- 传输网关（transport gateway）：传输层转发字节流
- 应用网关（application gateway）：应用层实现互连

无连接网络互联

- 可以理解成IP网络，路由工作过程类似数据报子网，互连网络中每个分组单独路由
- 路由设备：多协议路由器为连接的不同子网进行协议转换（分组格式、地址等）

隧道技术

- （def）**源和目的主机所在网络类型相同**，连接它们的是一个**不同**类型的网络（数据链路层的！！例如局域网、广域网），这种情况下可以采用隧道技术。
- 例子：PPT 76面

互连网络路由

- 路由工作过程：类似单独子网，只是更复杂
- 路由算法：两级
  - 内部网关协议（IGP, Interior Gateway Protocol），自治系统AS内部
    - RIP、OSPF、IS-IS
  - 外部网关协议（EGP, Exterior）
    - BGP

分片 fragmentation

- 解决网络互连时最大分组长度不同的措施：大分组经过小分组网络时，网关要将大分组分成若干片段，每个作为独立的分组传输
- 重组策略
  - 对其他网络透明：使每个片段经过同一出口网关，在那里重组
    - 问题：出口网关要知道何时片段到齐；所有片段必须从同一出口网关离开；大分组经过一系列小分组网络时反复分片重组，开销大。
  - 对其他网络不透明：中间网关不重组，由目的主机完成
    - 问题：对主机要求高（得能够重组）；每个片段都有一个分组头，网络开销增大
- 标记片段
  - 树型标记法：分组0分成三段，分别标记为0.0, 0.1, 0.2，片段0.0构成的分组被分成三片，分别标记为0.0.0, 0.0.1, 0.0.2。问题：段标记域要足够长，分片长度前后要一致（否则如果重传时网络的长度限制改变，就不知道应该传多长）
  - 偏移量法：定义基本片段长度使其能通过所有网络。分片时分组头包括原始分组seq，第一个基本片段的offset，最后片段指示位

防火墙 firewall

- 防止信息泄露或不好的信息渗透，在 网络边缘 设置
- 早期配置：两个路由器中间夹着一个应用网关。

7.5 网络层协议

- 网络层中 互联网可以看作自治系统（AS）的集合 ，是由网络组成的网络。
- 网络之间互连的纽带是 IP(Internet Protocol)协议

IPv4协议（Internet Protocol）

- IPv4头：20个字节的固定部分 + 0-40个字节的变长部分。
  - Version（4）
  - IHL（分组头长度，注意单位是32-bit word，即4 bytes，从5-15）
  - Type of Service（现在基本忽略）
  - Total length
  - Identification（即分片片重组的seq号）
  - DF（don't fragment 所有机器必须能接受<=576字节的片段，可以扔不能分组）
  - MF（若分片，除最后一个片段外都要置More Fragments位）

    - Fragment offset（除最后一个片段外的所有片段的长度必须是8字节的倍数，字段里填的是除8后的结果）
    - TTL（最大值为255，到0丢弃并给源主机发送告警分组）
    - Protocol（上层是哪种传输协议，TCP、UDP...）
    - Header Checksum（每16位求反，循环相加（进位加在末尾），和再求反）TBD?
    - Source address，Destination address
    - Options（4字节的倍数，eg 安全性，严格/松散源路由，路由记录，时间戳）
- IPv地址：网络号+主机号。一个IP 地址并不真正指向一台主机，而是指向一个 网络接口
    - 有类地址划分 classful addressing（A类0，B类10，C类110，D类1110，E类11110）
    - ABC的主机号都按字节对齐（host3字节、2字节、1字节），D没有网络号 + 主机号，表示多播，E是保留
    - 【注意】全0表示本网络 / 本主机，全1表示广播地址，因此进行IP地址分配时要保留这两个
    - network=全0：host=0本主机，否则本网络的host主机
    - host=全1：network=全1在本网络广播，否则在一个network广播
    - 127.x.x.x loopback
- 子网：为了便于管理和使用，可以将网络分成若干供内部使用的部分，对外仍表现为一个网络.
    - 子网掩码高len位全1，与IP地址做AND得到网络地址。(?)

ICMP协议（互联网控制消息协议 internet control message protocol）

- 主要用来报告错误和测试，ICMP报文 封装在IP分组中

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

- 解决网络层地址（IP地址）与数据链路层地址（MAC地址）的映射问题
- 工作过程
    - 若目的主机在同一子网内，用目的IP地址在ARP表中查找，否则用缺省网关的IP地址在ARP表中查找
    - 若未找到，则发送广播分组，目的主机收到后给出应答，ARP表增加一项
    - 每个主机启动时会广播自己的IP-MAC地址，ARP表项中的动态ARP有生存期
- ARP攻击：攻击者发出伪造的ARP响应，更改目标主机ARP缓存中的IP-MAC表项，造成网络中断
  /
  中间人攻击，存在于**局域网**（？）

RARP协议（反向地址解析协议）

- 解决数据链路层地址（MAC地址）与网络层地址（IP地址）的映射问题
- 用于无盘工作站的启动（没硬盘没地方存IP地址；有网卡就有MAC地址）
- 缺点：由于路由器不转发广播帧，RARP服务器必须与无盘工作站在同一子网内
- 一种替代协议BOOTP，它使用UDP（上面提到的这几个，ICMP，ARP，RARP，都不经过TCP和UDP）？

RIP协议（Routing Information Protocol）

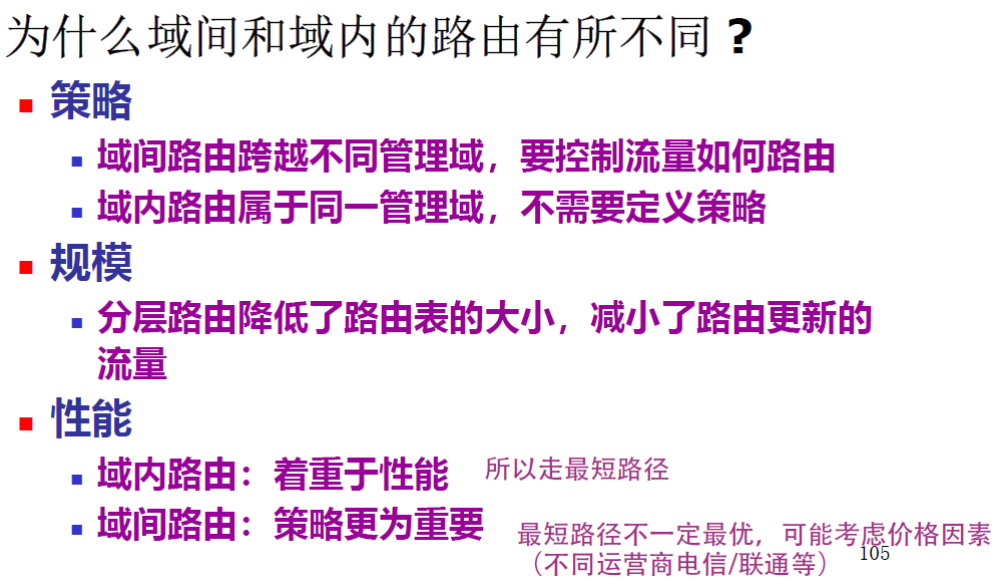
- 属于 内部网关协议（IGP），封装在 UDP分组 中，采用 距离向量 算法（DV）
- 故障处理：
    - 180s未收到邻居路由声明，则认为其失效，链路失效信息迅速传播到全网
    - 使用毒性反转避免ping-pong loops

- 水平分裂：A告诉我的消息我不会告诉A
  毒性反转：A告诉我的消息可以告诉A，但跳数要加上16跳（也就是让它变成不可达的）

OSPF协议（Open Shortest Path First）

- 属于 内部网关协议（IGP），采用 链路状态 算法（LS）
- 因此（TBD？）支持多种距离衡量尺度（物理距离、延迟、带宽...），支持基于服务类型的路由，支持负载平衡，支持分层路由，适量的安全措施，支持隧道技术
- 自治系统可以划分成区域，每个AS有一个主干（backbone）区域，称为区域0。所有其它区域都与主干区域相连，**其它区域之间不能相连。**（路由层面）
  - area内部用ls，之间只传精简信息
- 一般情况下有三种路由：区域内，区域间（从源路由器到主干区域、穿越主干区域到达目的区域、到达目的路由器），自治系统间（送到自治系统边界路由器）
- 四类路由器，允许重叠：（p107）
  i. 完全在一个区域内的内部路由器
  ii. 连接多个区域的区域边界路由器ABR
  iii. 主干路由器（就是主干区域的内部路由器）
  iv. 自治系统边界路由器ASBR

BGP协议（Border Gateway Protocol）



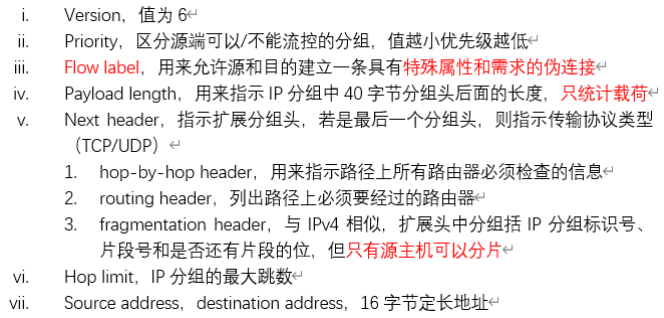
- 属于 外部网关协议EGP，封装在 TCP 分组中，采用 路径向量 算法（类似距离向量，每个BGP网
  关向邻居广播 所有通往目的地的路径；网关W收到了邻居网关X的路径，W依据开销、策略、防止出现路由循环等选择是否使用，若使用则path(w,z)=w, path(x,z)
  - X可以通过不告诉别人到Z的路径实现控制进入流量
- BGP消息：Open，Update，KeepAlive，Notification

无类域间路由CIDR（Classless InterDomain Routing）：RFC1519

- 提出背景：IPv4分配完毕，基于分类的组织浪费了大量地址（罪魁祸首是B类地址）
- 将剩余的C类地址分成大小可变的地址空间。将世界分成4个区，每个区分配一块连续的C类地址空间。
- 已推广到所有单播地址，用掩码确定前缀长度，可用于所有IP地址没有ABC类之分。
- 路由表中增加一个 `32位的掩码域` 。
  - 最长前缀匹配原则：路由查找时，若多个路由表项匹配成功，选择掩码最长（1比特数多）的路由表项。
  - 地址格式为a.b.c.d/x，x为地址中网络号的位数（即mask中连续高位1的长度len）
- 路由劫持

IPv6

- 20世纪90年代由IETF讨论形成。
- IPv6与IPv4不兼容，但与其他Internet协议（TCP、UDP、OSPF、BGP、DNS）等兼容，然而实际上还是需要开发另外一套协议栈。
- 目标：减少路由表大小、提供安全性、更多的关注服务类型特别是实时数据、支持组播（TBD：ipv4？？）、支持移动功能、协议有很好的可扩展项、允许与ipv4共存
- 主要变化
  - 地址从32位变成128位（16字节）
  - IP头由13个域减少为7个域，提高路由器处理速度
    - 例如分组头定长，就取消了IHL域，还有protocol, fragment相关（主机和路由器支持1280字节，如果要分片路由器给主机发错误信息，主机来分片）, checksum域都无了
    - Type of service替换为Priority和Flow label。Protocol替换成next header
  - 安全性提高
  - 更好地支持选项功能
- 分组头



i. Version，值为 6
ii. Priority，区分源端可以/不能流控的分组，值越小优先级越低
iii. Flow label，用来允许源和目的建立一条具有特殊属性和需求的伪连接
iv. Payload length，用来指示 IP 分组中 40 字节分组头后面的长度，只统计载荷
v. Next header，指示扩展分组头，若是最后一个分组头，则指示传输协议类型（TCP/UDP）
   1. hop-by-hop header，用来指示路径上所有路由器必须检查的信息
   2. routing header，列出路径上必须要经过的路由器
   3. fragmentation header，与 IPv4 相似，扩展头中分组括 IP 分组标识号、片段号和是否还有片段的位，但只有源主机可以分片
vi. Hop limit，IP 分组的最大跳数
vii. Source address，destination address，16 字节定长地址

  - 扩展分组头：hop-by-hop header(所有路由器必须检查)、routing header(必须经过的路由器)、fragmentation、encrypted security payload、destination options
- IPv6地址表示：16字节地址表示为用冒号 : 隔开的8组，每组4个16进制位（2 byte），例如 `8000:0000:0000:0000:0123:4567:89AB:CDEF` ,每个数字开头的0可省略，连续的0可以被一对冒号替代，但是一对冒号只能出现一次

IPv4向IPv6过渡

1. 双栈：实现IPv4/v6 两套协议栈 ，主机根据DNS返回的结果或对方发来报文的版本号决定采用哪
   个协议，路由器根据收到IP分组的版本号决定采用哪个协议
2. 翻译（多协议路由器）：有些路由器实现IPv4/v6两套协议栈，在两套之间 进行协议翻译和地址翻译 ，例如NAT-PT（已废弃）——已经被NAT-64等协议代替
3. 隧道：IPv6的报文作为IPv4报文的净负荷在IPv4网络中传输

1.1

ch8 Transport Layer 端到端访问与传输层

8.1 传输服务

- 引入传输层的原因
  - 消除网络层的不可靠性（传输层不存在有无确认的服务之分，只有 有无连接 的服务之分）
  - 提供从源端主机的 进程 到目的端主机的 进程 的可靠的、与实际使用的网络无关的数据传输
- **传输实体**（transport entity）：完成传输层功能的硬**软**件。传输层实体利用网络层提供的服务向上层提供 有效、可靠 或 尽力而为 的服务。
- 1-4层：传输服务提供者；4层以上：传输服务用户
- **传输服务原语**
  - 功能：传输用户（应用程序）通过传输服务原语访问传输服务
  - 举例：一个C/S模式的原语可以有Listen、Connect、Send、Receive、Disconnect
- **TPDU**：transport PDU（（一个非官方的理解）传输层的实体发给对等实体的信息单元）

简单连接管理

- 拆除连接的方式：不对称、对称（P8有一个状态图举例，有空可以看看TBD）
- 一个例子：Berkeley Sockets
  - 原语：Socket、Bind、Listen、Accept、Connect、Send、Receive、Close
  - 数据传输 - 全双工；连接释放 - 对称
  - 应用举例：P10-11
    - 服务端：socket->bind->listen（非阻塞）->accept（阻塞）->派生进程去处理->close
    - 客户端：socket->connect->close

8.2 传输层编址

- 传输服务访问点 TSAP （Transport Service Access Point）：将应用进程与TSAP相连，IP+端口
- 端口号的获取：预先约定（telnet 23），或从名称服务器/目录服务器获取。ppt的例子上获取过程是面向连接的（一个特殊的进程称为 名称服务器/目录服务器 ，它的TSAP大家都知道。客户

进程与其建立连接、发送服务名称，获得其TSAP，释放与名称服务器的连接，与服务进程建立连接）
- 进程服务器：监听多个端口（eg inetd），需要时唤起对应的服务进程。（没人监听时它去帮忙请求）

8.3 连接管理

- 传输层建立连接的复杂性原因：网络可能丢失、重复包，特别是 `延迟重复包` 的存在。（无法得知是延迟还是重复）--> 解决关键：丢弃过时的包
- **建立连接**：采用 `三次握手` （ppt20面的例子！）
  - A发出序号为 X 的CR TPDU（Connection Request）
  - B发出序号为 Y 的CC TPDU（Connection Confirm） & 确认A的序号 X 的CR TPDU
  - A发出序号为 X+1 的第一个数据TPDU & 确认B的序号为 Y 的CR TPDU

    > 一些可以解决的情况举例：
    >
    > - 延迟重复CR：当B收到的是一个step1延迟重复包的时候，回送的TPDU含seq=y,ACK=x（step2指明对方身份），A可以发现这是延迟重复包然后REJECT，不进入step3
    > - 延迟重复CR+ACK
    >   - CR：当B收到了一个step1延迟重复包的时候，回送的TPDU含seq=y,ACK=x1，A可以从x1序号发现是延迟重复包然后REJECT，不进入step3；
    >   - ACK：此后B又收到了step3延迟重复包seq=x2, ACK=z，B可以从z序号发现这不是自己回送的最新seq=y，判定是延迟重复包，不进入step2。
- **释放连接**：采用 `三次握手+定时器`
  - 原因：（DR = Disconnection Request）
    - 非对称式释放可能丢失数据（Data和DR交叉）
    - 对称式存在两军问题（你ack了我的ack）

    > 以下不同的情况都能正常释放：（表明绝大多数时候是成功的）
    >
    > 1. 正常的三次握手：
    >   A发DR给B并启动timer
    >   B回送DR给A并启动timer（A收到DR就释放）
    >   A发送ACK确认B的DR（B收到ACK就释放）
    > 2. 最后的A发的ACK丢了，B的timer超时从而释放
    > 3. B回送的DR丢了，A的timer超时重发DR给B，直到某次B正常回送DR没有丢，归约到case 1 ~ 2
    > 4. B回送的DR丢了，A的timer超时重发DR也丢了，后续一直丢（没有正常），双方timeout N次后认为整个连接断开，双方都会断开并向上层报错

8.4 缓存和流控

缓存

- 发送方：对每个连接单独缓存它发出的所有TPDU，以便错误时进行重传
- 接受方：可做可不做
- 设计：固定大小（按最大包长度，会有浪费） / 可变大小 / 缓存池（时间长了有碎片）

流控

- 可变滑动窗口协议：发送方的发送窗口大小是接收方根据自己的实际缓存情况给出（在ACK里说）的！传输层实现流控的方式。（注意端系统ACK应该周期性发出，避免一个TPDU丢失导致死锁）

8.5 互联网传输协议

传输控制协议TCP（Transmission Control Protocol）：

- 面向连接、可靠、端到端、基于字节流
- RFC 793,1122,1323,2018,2581

> 回忆：chap2数据链路层协议
> HDLC、X.25 LAPB协议：面向比特
> PPP协议：面向字符

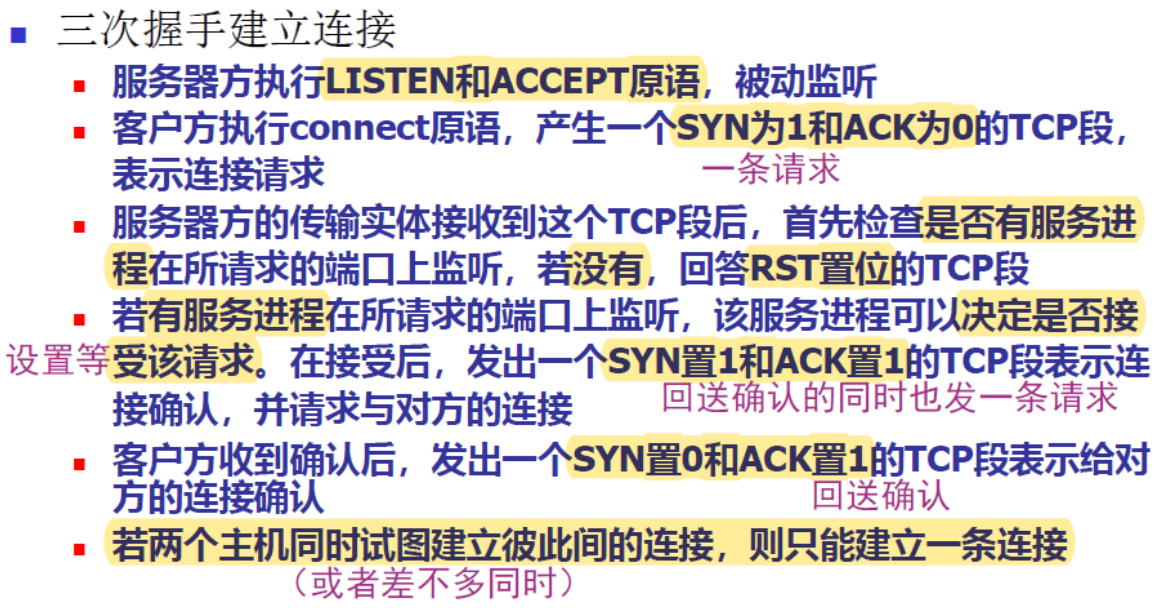
用户数据协议UDP（User Data Protocol）：

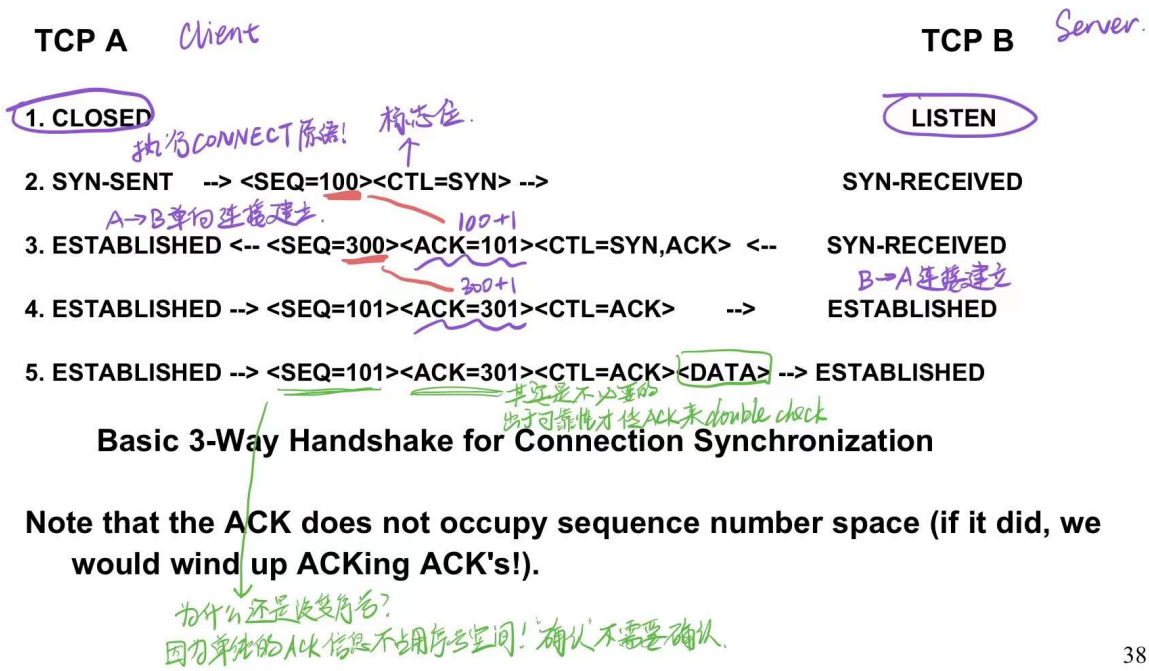
- 无连接、端到端、基于消息流
- RFC 768

TCP

- 应用程序访问TCP服务：通过sender、receiver之间创建套接字
- 表示法：
  - 套接字地址用 (IP address, port) 表示，256以下的端口号被标准服务保留（eg FTP(21), TELNET(23)）
  - 一条连接用 (socket1, socket2) 表示，是点到点的全双工通道
    - 因此TCP流的标识是个五元组，（src 地址, src 端口, dst 地址, dst 端口,协议）
- 特点：
  - TCP不支持组播、广播（想可靠很难）
  - TCP基于字节流，消息的边界无法保留
  - 对于应用程序发来的数据，可以立即发送，也可以缓存一段时间再发（因为header比较长），可以用各种标记位标识数据发送要求（如PUSH强迫发送、URGENT紧急数据）
  - 按 字节 分配序号，每个字节有一个 32位 的序号（TBD？）
  - TPDU称为 段 （segment），段的大小要满足
    - 网络层：IP包对payload的长度限制（65536字节）
    - 数据链路层：最大传输单元（MTU）的限制（e.g. 以太网MTU为1500字节）
  - 使用滑动窗口协议，确认序号等于接收方希望接受的下一个 字节 序号
- **TCP头**
  - TCP头：**定长20字节**+变长
  - 源和目的端口：各16位
  - 序号和确认号：以字节为单位编号，各32位
    - 序号：

        - 在 SYN flag 置 1 时，此为当前连接的初始序列号（Initial Sequence Number, ISN），数据的第一个字节序号为此 ISN + 1
        - 在 SYN flag 置 0 时，为当前连接报文段的累计数据包字节数。
      - 确认号：32 位确认序号，ACK flag 置 1 时才有效，指接收方 期待的下一个报文段的序列号 。
    - TCP头的长度：4位，长度单位为32位字，包含选项域
    - 6位的标识位 （长度+这个2字节）
      - URG：紧急指针是否有效
      - ACK：确认号是否有效
      - PSH：指示发送方和接收方将数据不做缓存，立刻发送或接收
      - RST：不可恢复的错误重置连接
      - SYN：连接建立指示
      - FIN：连接释放指示
    - 窗口大小：用于基于可变滑动窗口的流控，指示发送方从确认号开始可以再发送窗口大小的字节流
    - 校验和（2字节）：TCP头，数据和IP伪头（网络层包括12字节：2个4字节的地址，0，protocol=6,2字节TCP段长度）的校验和
    - 紧急指针（2字节）：指向从当前序号开始找到紧急数据的字节偏移量，URG=1时有效
- **TCP连接管理**





- 释放连接：三次握手 + 定时器（对称方式释放）
- （TBD）p40状态机

- **TCP的窗口管理机制**
  - 基于确认和可变窗口大小
  - 正常情况下，窗口大小为0时发送方不能再发TCP段，但有两个例外：**紧急数据**可以发送；为防止死锁，发送方**可以发送1字节的TCP段**，以便让接收方重新声明确认号和窗口大小
  - 改进
    - 发送方 缓存 应用程序的数据，等到形成一个比较大的段再发出
    - 在没有可能进行“捎带”的情况下，接收方延迟发送确认段
    - Nagle算法：当应用程序向传输实体发出一个字节时，传输实体 只发出第一个字节 并缓存所有其后的字节 直至收到对第一个字节的确认，然后将已缓存的所有字节组段发出并对再收到的字节缓存，直至收到下一个确认
      - 可以避免多次发送小包，但是不利于实时性强的应用。可能需要禁用Nagle算法
    - 使用 Clark算法 解决傻窗口症状
      - 傻窗口症状：当接收方一次从传输层实体读出一个字节时，传输层实体会产生一个一字节的窗口更新段，使得发送方只能发送一个字节
      - 解决办法：限制收方只有在具备**一半的空缓存或最大段长的空缓存**时，才产生一个窗口更新段

- **TCP拥塞控制**
  - 拥塞的两种情况：
    1. 网络快，receiver缓存小 —— 只需要做流控，采用 可变滑动窗口 / 设置静态的最大可接收窗口
    2. 网络慢，receiver缓存够 —— 要做拥塞控制，采用 拥塞窗口 （按照慢启动+拥塞避免算法）
  - TCP发送的段长取二者的最小值
  - 维护一个拥塞窗口，窗口大小是任何时候sender可以往网络发送的字节数（但实际题目中通常
    是字节 / 最大段长）
  - 慢启动算法：

        - 连接建立时拥塞窗口初始值为1个最大段长MSS，阈值为64K；
        - 每发一个段被ack时窗口大小就+1（相当于每个RTT周期内窗口大小乘2），直到丢包超时或者窗口大于阈值，进入拥塞避免算法。
        - 注意可能有累计确认的情况！！（一个ack帧可能是对多个发出段的ack，那么要+多个congwin）
    - 拥塞避免算法：
        - 拥塞窗口大于阈值时，窗口线性增长，一个RTT周期congwin+1，直到丢包超时。
        - 若超时，阈值设置为当前拥塞窗口大小的一半，拥塞窗口重新设置为一个最大段长，进入慢启动
    - 检测丢包：发生了①计时器timeout / ②收到3个重复确认ack
    - 快速重传算法：（就是把拥塞避免算法的until timeout替代为until ack×3）
        - 连续收到3个重复确认后，不再等待计时器超时，而是
        - 将阈值设置为当前拥塞窗口大小的一半，拥塞窗口重新设置为一个最大段长，进入慢启动算法
    - 控制拥塞分析：有效分配、公平分配
    - 系统模型和选择拥塞控制函数（TBD？？？？）
        - $x_i(t+1)=\begin{cases}a_I+b_I x_i(t) increase\\ a_D+b_D x_i(t) decrease\end{cases}$
        - MIAD：公平性不收敛也不稳定。有效性不收敛，在x1=x2=bIaD/(1-bI)时稳定。
        - AIAD：公平性不收敛但稳定。有效性不收敛，在aI+aD=0时稳定。
        - MIMD：公平性不收敛但稳定。有效性收敛。条件是平凡的。
        - AIMD：公平性收敛。有效性收敛。（TCP采用！！！）
    - 【重要】例题：p63！！！（ACK是加上本段长度？）

        - **A、B双方已经建立了TCP连接，初始阈值为32K字节(1K = 1024)，最大发送段长MSS为1K字节。发送方向为A->B，B没有数据要发送，B每收到一个数据段都会发出一个应答段。在整个过程中上层一直有数据要发送，并且都以MSS大小的段发送。A的发送序列号从0开始。**
            - **在传输过程中，A收到1个ACK为10240的数据段，收到这个应答段后，A处拥塞窗口的大小是多少?**
            - **当收到ACK为32768的数据段后，A处拥塞窗口的大小是多少?**
            - **当阈值为32K字节、拥塞窗口为40K字节时，发送方发生了超时，求超时发生后拥塞窗口的大小和阈值的大小。**

判断是处于慢启动还是拥塞避免阶段

在慢启动：一批都得到确认则翻倍=每收到一个段则拥塞窗口+1，所以拥塞窗口的大小就是ACK所确认的序号+1！！！

1.收到的是对第10个段的应答，变化后拥塞窗口的大小为11

编号顺序增加！而不是每一批都是从0开始（这里应该的第四批）

2.收到的是对第32个段的应答，这时拥塞窗口已经超过阈值，应该使用线性增长，变化后的拥塞窗口大小为 32 K字节。

收到第31段之后就变成32K了，之后得等32个都收到ACK才加

3.拥塞窗口 = 1 MSS = 1KB，阈值 = 40 / 2 = 20 KB

UDP

- 优点：不需要建立连接，延迟小；简单，没有连接状态；报文头小；没拥塞控制，可以尽快的发送
- 特点：
  - 简单的传输协议；
  - “best effort”服务，UDP 报文可能会丢失、乱序；
  - 无连接：无需握手，每个UDP报文的处理都独立于其他报文。
  - 经常用于流媒体应用 ： 可以容忍丢包，速率敏感
  - 其他应用范围：
    - RIP，路由信息周期发送
    - DNS，（域名解析）避免TCP连接建立延迟
    - SNMP，当网络拥塞时网管也要运行。信息带内传输
    - 基于UDP的可靠传输：应用程序自己定义错误恢复
    - BOOTP
- 头格式：
  - src接口，dst接口，udp length和udp checksum（4个16位，一共8字节）

ch9 Application Layer 网络应用

9.1 应用层概述

- 网络应用程序：互相通信的分布式进程
- 应用层协议：应用程序的一部分，定义应用程序之间交换的信息以及相应的动作，利用底层协议提供的服务
- 进程通信：
  - 同一主机 —— 利用操作系统提供IPC（interprocess communication 进程间通信）
  - 不同主机 —— 利用应用层协议
    - 用户代理（user agent）： 用户和网络应用程序间 的接口。比如web浏览器，流媒体播放器等。
    - API（应用程序编程接口）：定义 应用程序和传输层 之间的接口
    - “应用层寻址”：进程通过<IP address, local port> 定位对方进程，其中IP指明对方主机，port指明对方进程
- 应用程序需要的传输服务：
  - 数据丢失容忍度（音频/文件）
  - 带宽容忍度（多媒体/文件传输）
  - 延迟容忍度（IP电话交互游戏/电子邮件）
- 互联网传输协议提供的服务
  - TCP：面向连接、进程间的可靠传输、流量控制、拥塞控制，不提供延迟保证和最小带宽保证
  - UDP：不可靠传输，不提供上述所有

9.2 客户/服务器模型

- 客户/服务器模型是 网络应用的基础 ，客户/服务器分别指参与一次通信的两个应用实体，客户方主动地发起通信请求，服务器方被动地等待通信的建立，并提供服务。
    - 服务器软件的 并发性 ：支持多个客户的同时访问，为每一个人建立一个进程/线程
    - 服务器软件的 组成 ：一部分用于接受请求并创建新的进程或线程，另一部分用于处理实际的通信过程。
- 典型C/S模型系统： DNS，HTTP，FTP
- 使用的传输层协议：
    - 基于连接的TCP（适用于可靠的交互过程）
    - 无连接的UDP（适用于可靠性要求不高 / 实时交互的过程）
    - 同时使用TCP、UDP

9.3 域名服务DNS（domain name system）

- DNS的request / response报文封装在 UDP 分组中
- 网络规模小时，（例如ARPANET），每个主机只要查找一个文件（UNIX的host），列出了主机和IP地址对应关系 ---> 规模大时不适用
- 典型的 客户 / 服务器 交互系统
- 多层次、基于域的命名系统，使用分布式数据库实现
- RFC 1034， 1035
- 操作过程
    - request - 应用程序需要进行域名解析时，它成为client，并试图与本地域名服务器发出请求，建立连接
    2. reply - 本地域名服务器找到对应IP地址，给出响应
    3. （多层次的DNS：本地域名服务器找不到时，自身成为上级域名服务器的client，继续解析）
- 域名结构
    - 互联网的顶级域名分为 组织结构(.org)和地理结构(.cn) 两种。每个域对它下面的子域和机器进行管理。
    - 域名是“.”分的字符、数字串组成的，大小写无关，最长255个字符，每个部分最长63个字符。
- 资源记录：DNS数据库中用来表示主机和子域的信息，域名解析成功的返回结果就是它，形式上是五元式<Domain_name, Time_to_live, Type, Class, Value>（具体含义见P20-22）
    - Type=A，Name：hostname，Value：IP地址
    - Type=MX，Value：与name对应的邮件服务器的主机名（hostname）
    - Type=NS，Name：域名（例如，edu.cn），Value：该域权威域名服务器的IP地址
    - Type=CNAME，Name：规范名称（ canonical name）的别名，Value：规范名称
- 区域划分：DNS将域名空间划分为许多**无重叠区域**，每个区域覆盖了域名空间的一部分，区域的边界划分是人工设置的。每个区域有一个**主**域名服务器和若干个**备份**域名服务器。
- 域名解析过程：主机->本地域名服务器-（若找不到）->根域名服务器-（若找不到）->权威域名服务器
- 根域名服务器可能不知道 authoritative域名服务器，但知道 中间域名服务器 ，而中间域名服务器知道如何与authoritative域名服务器联系
- 两种请求模式 （一般本地DNS用递归，根服务器用反复）

- Recursive query（递归式）：问一个人就可以得到回答
  - 【优点】向用户提供较好的服务【缺点】根域名服务器的负担重
- Iterated query（反复式）：别人告诉你可以问谁，自己反复问不同的人
  - 【优点】负担较小【缺点】提供的服务不太好

9.4 简单网络管理协议SNMP（simple network management protocol）

- 封装在 UDP 中，RFC 1441 - 1452
- 五大管理功能： 性能管理、故障管理、配置管理、记账管理、安全管理
- SNMP模型的四个组成部分：agent、manager、information (MIB)、protocol (SNMP)
  - 被管理节点：运行 SNMP代理程序 （agent），维护 本地数据库 （记录状态历史）
  - 管理工作站：运行专门的 网络管理软件 （manager），使用管理协议 与被管理节点上的SNMP代理通信 ，维护 管理数据库
  - 管理信息：每个站点使用多个对象描述自己的状态，所有的对象组成MIB（ 管理信息库 management information base）
  - 管理协议（SNMP）：管理协议用于 管理工作站查询和修改被管理节点的状态 ，被管理节点可以使用管理协议向管理站点产生 “陷阱（trap）”报告 。
    - 管理工作站发给SNMP代理：数据请求get-request，更新请求set-request
    - 管理工作站之间的MIB交换：inform-request
    - SNMP代理发给管理工作站：snmpV2-trap
- 管理信息结构SMI：变量被定义为对象（object），相关对象们被集合成组（group），组被汇集成
  模块（module）。每个对象的四个属性：对象类型（名字）、语法（数据类型）、存取（能否读写）、状态（必备 可选 已经被废弃）
- 管理信息库MIB：包含10个组，manager向agent查询的就是保存在MIB中的对象的值。
- 抽象语法表示法1（ASN.1）
  - 标准的**对象定义**语言。分为**数据描述**定义和**传输语法**定义两部分。可以作为异种计算机设备之间“对象”描述和传输的表示方法。
  - 数据描述定义：
    - 基本数据类型：Integer=2、Bit String（bits）=3、Octet String（bytes）=4、Null=5、Object Identifier（type）=6
    - 对象命名树：使用编码（如上1~6），唯一确定每个标准中的对象
    - 构造新类型的方法：Sequence（多种类型的list）、Sequence of（一种类型的list）、Set、Set of、Choice（ union ），或者重新标记一个老的类型（类似#define ）
  - 传输语法定义：
    - 基本编码规则BER(basic encoding rules)：ASN.1类型的值 → 无二义的字节序列
    - 传输内容：
      - 标志符：tag | type | number（具体见P36）
        - 高两位：00~11分别为Universal，Application，Context specific，Private。第三位：0为primitive，1为constructed。
        - tag在0~30之间时，用低5位表示
        - tag > 30时，低5位为11111，用后面字节表示
      - 数据长度域

        - 数据长度<128则填入该长度，用一个字节表示长度，最高位为0
        - 数据长度>128则最高位=1，低7位=x ，后面x个字节表示长度
            - eg 数据长度1000字节，则长度域有3个字节，分别为“10000010”，“00000011”，“11101000”（表示1000的16位bit）
    - 数据域：
        - INTEGER：二进制编码
        - BIT STRING：编码表示不变，长度域表示字节个数（需考虑表示...不用的位数的额外字节），在传位串前先传一个字节表示位串最后一个字节不用的位数。例，位串“010011111”传输时变为“07 4f 80”（十六进制），长度为3
        -

Fig. 7-34. ASN.1 encoding of some example values.

9.5 电子邮件

- 封装在 TCP 内
- 1982年ARPANET提出RFC821（传输协议） RFC822（消息格式）
- 体系结构：
    - 用户代理：允许用户阅读和发送电子邮件，一般为用户进程
    - 消息传输代理：将消息从源端发送至目的端，一般为系统的后台进程
    - 简单邮件传输协议SMTP（Simple Mail Transfer Protocol，注意和SNMP区别）：只支持用户代理→邮件服务器，或邮件服务器之间
- 电子邮件的组成：
    - 信封：接收方的信息，如名字、地址、邮件的优先级和安全级别
    - 信件内容：由 信头和信体 组成，信头包含了用户代理所需的控制信息，信体是真正的内容
- 电子邮件的扩展：MIME（Multipurpose Internet Mail Extensions），对图像声音等支持，使用编码将信息转化为ASCII字节流。
- 消息传输代理在源端主机和目的主机的**25号端口建立TCP连接**，使用SMTP进行通信（还有其他可用
  的协议如POP3、IMAP，SMTP是最基本的）；以CRLF.CRLF结束
- 消息传输代理和用户代理两个方向的发送完成后，释放TCP连接(QUIT)。属于**持久**（Persistent）方
  式。
- 以7比特ASCII码为单位，某些特殊字符串不能在消息中出现，需要编码（eg base64）

- POP3：RFC1939，用户代理和邮箱不在一个机器，用户代理使用此协议将邮箱里的信件取回本地
- IMAP：RFC1730，收信人使用多个用户代理访问同一个邮箱，邮件始终保持在邮箱里
- 加密：PGP、PEM协议

9.6 WWW

- WWW(World Wide Web)是用于访问遍布于互联网上的相互链接在一起的超文本的一种结构框架
- WWW模型的四个元素：
  - Web页面：由对象（object）组成，用URL标识地址（协议类型+server地址+object路径名）
  - 浏览器（browser）：用户访问网页的客户端client
  - Web服务器：存储Web对象（object）的server
  - 超文本传输协议HTTP（基于 `TCP` ，在80端口）： `C/S模型，无状态协议` （不保存客户信息）
- HTTP1.0：非持久连接，每个object取得至少要两个RTT（因为TCP连接至少需要一个RTT来建立），每个object的传输都要经历慢启动（不过大多数1.0浏览器会并行发TCP连接请求）
- HTTP1.1：持久连接，客户端一旦得到基本的HTML文件就发出请求索取全部object，较少的RTT和慢启动时间
- Web缓存：
  - 工作原理：浏览器不访问原服务器（origin server），而去访问代理服务器（proxy server）
    - 缓存命中，Web对象（object）被立刻通过HTTP response返回
    - 不命中，代理服务器向原服务器请求获得该object再返回
  - 响应快，减轻远端服务器的负担

9.7 文件传输协议FTP

- `客户/服务器模式` ：由客户端发起文件传输
- 客户端连接到ftp服务器 `TCP的21号端口` ，然后建立两个并行的TCP连接:
  - **控制**: （port21）在客户端和服务器之间交换命令、响应(采用 `telnet`)
  - **数据**: （port20）传递文件数据(可双向，不必始终存在)
- FTP服务器维护 `状态` ：当前目录，身份认证
- 带内传输：HTTP、SMTP（电子邮件）；带外传输：FTP