可靠数据传输 rdt1.0-3.0

1. 1.0 假设底层信道完全可靠
2. 2.0 假设分组顺序正确但比特可受损
    - 引入自动重传请求协议发送ACK/NAK 停等协议 没有考虑ACKNAK受损
3. 2.1 引入冗余分组的情况
    - 在分组中加入字段存放分组编号 确保分组不丢失
4. 2.2 无NAK
    - ACK有序号 提出冗余ACK
5. 3.0

   又叫做比特交替协议（ABP）
    - 允许丢包 引入倒计数定时器

路由器和交换机维护的转发表的区别

1. （这玩意在ppt里没有一个明确的结构说明，很迷）
2. 交换机的转发表：
    - 最关键的是**目标MAC+出端口**
3. 路由器的转发表：
    - 最关键的是**目标IP+下一跳IP**（+出接口）

二，三层交换机区别

1. 三层交换机除了交换功能也可以实现路由功能
2. 因为port-based VLAN的存在，一个交换机中的端口可以被分配到不同的广播域即vlan，三层交换机的路由功能就是为了使一个交换机中的不同VLAN间能够通信。
3. 一个VLAN的interface可能分配在多个交换机中，802.1Qframe加上了VLAN ID，通过对VLAN号的显式声明实现了不同交换机中VLAN的通信

TCP累计确认Cumulative ACK

例题

(1) Suppose Host A sends four TCP segments back to Host B over a TCP connection. The first segment has sequence number 78; the second has sequence number 138; the third has sequence number 258; the forth has sequence number 348 with 60 byte of data in it.

138 − 78

(a) How much data are in the first, second and third segment respectively? (2 scores)

Answer: 60 byte, 120 byte, 90 byte

(b) Suppose that the first segment and the forth segment arrive at B in turn, but the second segment and the third segment are lost. What will be the acknowledgment number respectively that Host B sends to Host A for each arriving segment? (2 scores)

Answer: 138, 138

(c) When Host B receives the second segment and the third segment re-sending by Host A in turn, what will be the acknowledgment number that Host B sends to Host A for each arriving segment. (2 scores)

Answer: 258, 408

注意：TCP不是严格的GBN！TCP如果发现最大收到的段前面的段没收到，补收到前面的段后，直接从最大收到位开始返回ACK！（累计确认）

- GBN无缓存机制，发送的ACK是连续收到段的最后一个字节+1
  - 比如上一题第二问中收到的序号为1→4→2→3，则GBN接收方返回的ACK只能是第二段的开头seq！
- **TCP有缓存机制，结合了GBN和SR。**

同时注意区分MAC协议和滑动窗口协议（GBN,SR）的区别！

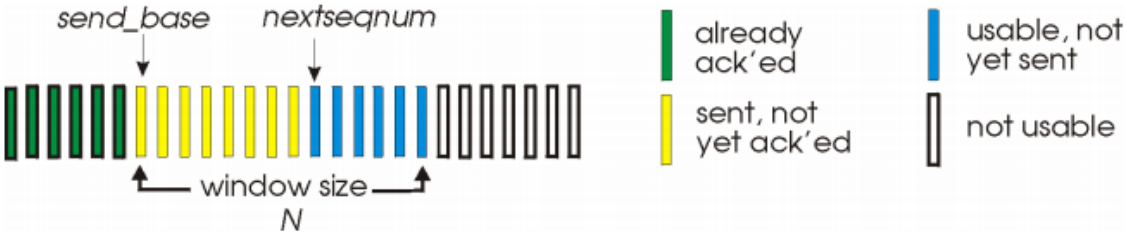
1. **滑动窗口协议属于传输层，是为了实现RDT，分两种**
   1. **GBN**
   2. **SR**
      - 接收方独立缓存乱序分组：
        1. 与 GBN 直接丢弃乱序分组不同，SR 接收方会缓存所有正确到达的分组
        2. 等待缺失分组到达后组装有序数据
      - 发送方选择性重传：
        1. 仅重传超时未确认的分组
        2. 不涉及窗口内其他分组（避免 GBN 的"全窗口回退"问题）
2. **MAC协议属于链路层,分类如下：**
   1. **信道划分类协议（Channel Partitioning)**
      - **TDMA**
      - **FDMA**
   2. **随机访问类协议（Random Access)**
      - **ALOHA**
      - **CSMA、CSMA/CD、CSMA/CA**

3. **轮流访问类协议（Taking Turns）**

PS.相似概念比较：IP分片中的“号”

> - IP分片中的号：属于网络层，叫做offset，表示每一个IP分片中的数据开头在原数据报中数据部分的位置（8byte为一单位）
> - TCP累计确认中的seq#（序列号）以及ACK#（确认号）
>   - seq#
>   - 理解前提：TCP传输时将数据看为连续的字节流。
>   - 作用为标识本报文**第一个数据字节**在**整个原始数据流中的位置。**
>   - （在实际情况中ISN（初始序列号）可能随机，防止攻击。）
>   - 表示这个segment中的数据从seq#字节开始，到下一segment的seq#-1结束，总共有（seq#下-seq#上）字节。
>   - ACK#
>   - （回复报文时，TCP头部中ACK_flag=1，显式申明这是一条ACK报文。）
>   - 表示自己已经收到了ACK#前的所有字节（不包括ACK#），期待下一次传过来的数据是从ACK#开始的。
> - 所以一般传输数据时，正常无丢包情况下，客户端要传输的下一个数据的seq#和服务器期待的ACK#是一样的！

GBN



sender:

ACK(n): ACKs all pkts up to, including seq # n - “cumulative ACK”
may receive duplicate ACKs (see receiver)
timer for each in-flight pkt
timeout(n): retransmit pkt n and all higher seq # pkts in window

Receiver:

ACK-only: always send ACK for correctly-received pkt with highest *in-order* seq #

may generate duplicate ACKs

need only remember **expectedseqnum**

out-of-order pkt:

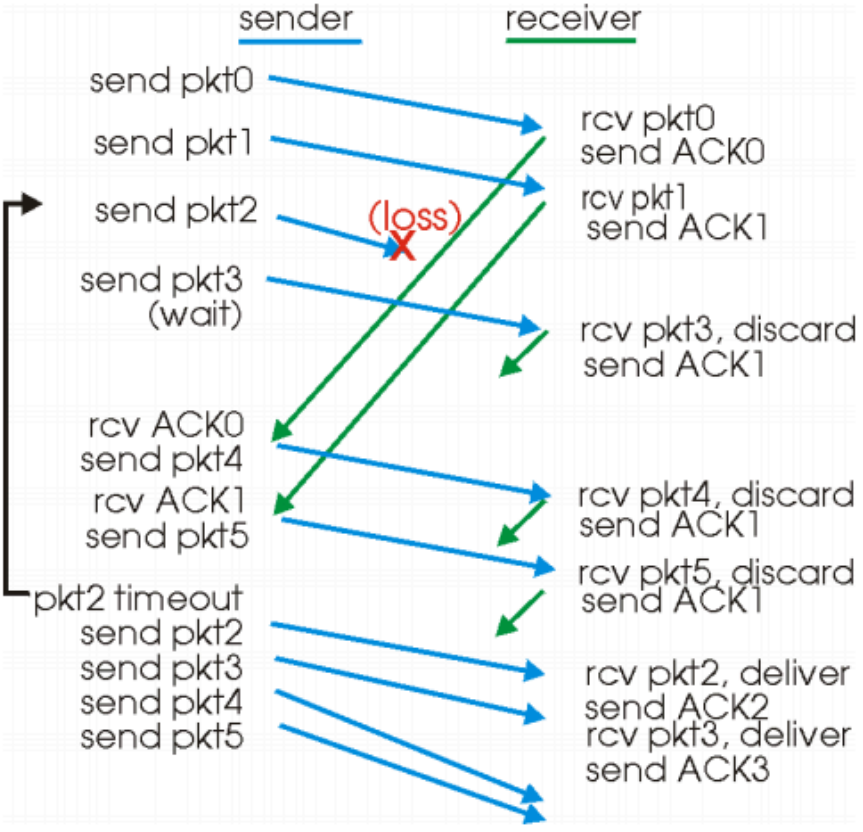
discard (don't buffer) -> no receiver buffering!

Re-ACK pkt with highest in-order seq #

- **发送方（Sender）**：
  - 维护一个发送窗口（**窗口最大大小为序号空间大小-1**），窗口内的包可连续发送。
  - 每收到一个 `ACK`（不重复），窗口滑动，**滑动量=已确认的分组量**。**
    1. 一个窗口范围中的所有包可以并行发出，但是只有在收到ACK时窗口才滑动！！！ **（收到ACK后确认多少，就滑动多少）**
    2. 只有窗口中最早未确认的包需要设立一个timer。
  - 如果某个包超时（如 `P2` 丢失），**重传 `P2` 及之后的所有包**（即使 `P3` 已正确到达）。
  - （上面这只是GBN的特性！！！TCP的接收方可以缓存已经正确到达的包！！！而GBN只能返回连续确认包的ACK！）
- **接收方（Receiver）**：
  - 只按顺序接收包（如期望 `P2` 但收到 `P3`，则丢弃 `P3`，没有receiver buffering！！）。
  - 发送最近正确接收的包的 `ACK`（累积确认）。

GBN过程图示

GBN in action



左边：

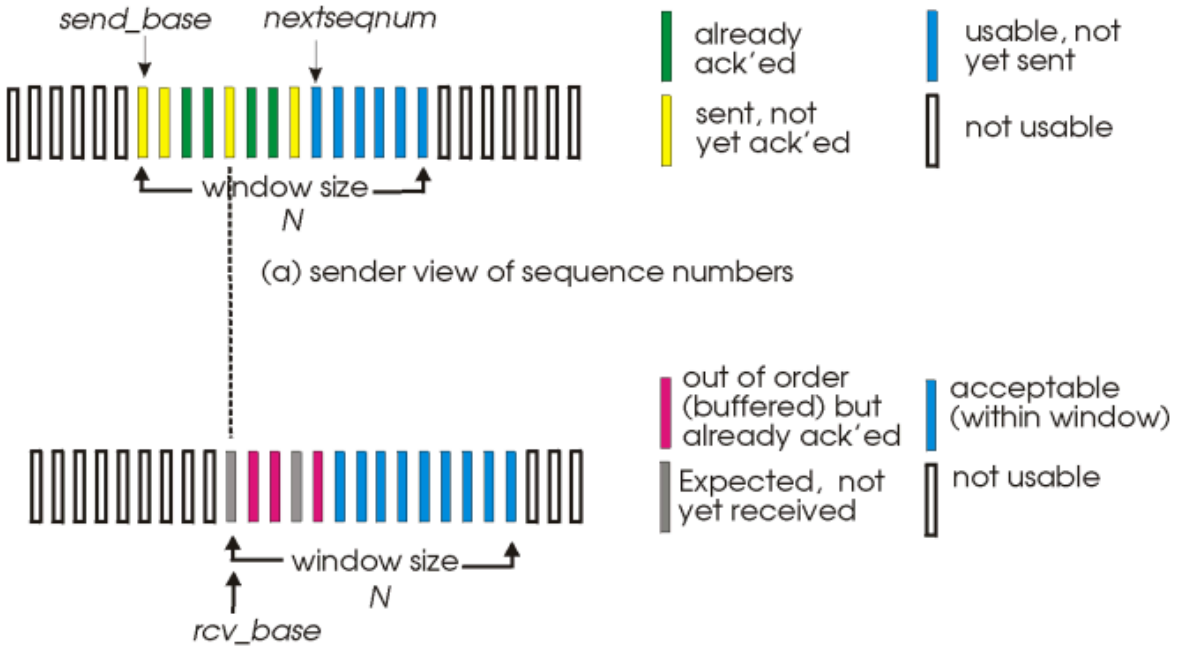
- 第四行，这里窗口大小为4，有没有收到ACK，没东西发了，所以等等。
- 第五行，收到ACK0，窗口滑动到pkt1，pkt4现在在发送窗口中了，就直接发了。第六行同理。
- 第七行，pkt2好久没收到自己的ACK，timeout了，于是发送方直接重新发送pkt2之后的所有包。

右边：

- 第三四五行，收到的pkt3，4，5都是乱序的，所以依旧回复ACK1给发送方。
- 第六行后就是正常接收啦。

PS.这里**采用数据包标号**的seq#和ACK，和TCP中的**采用字节流标号**的seq#和ACK（有加一字节减一字节的问题）不同哦，注意区分！

SR



sender

data from above :

- if next available seq # in window, send pkt

timeout(n):

- resend pkt n, restart timer

ACK(n) in [sendbase,sendbase+N]:

- mark pkt n as received
- if n=sendbase, advance window base to the unACKed pkt with smallest seq #

receiver

pkt n in [rcvbase, rcvbase+N-1]

- send ACK(n)
- out-of-order: buffer
- in-order: deliver (also deliver buffered, in-order pkts), advance window to next not-yet-received pkt

pkt n in [rcvbase-N,rcvbase-1]

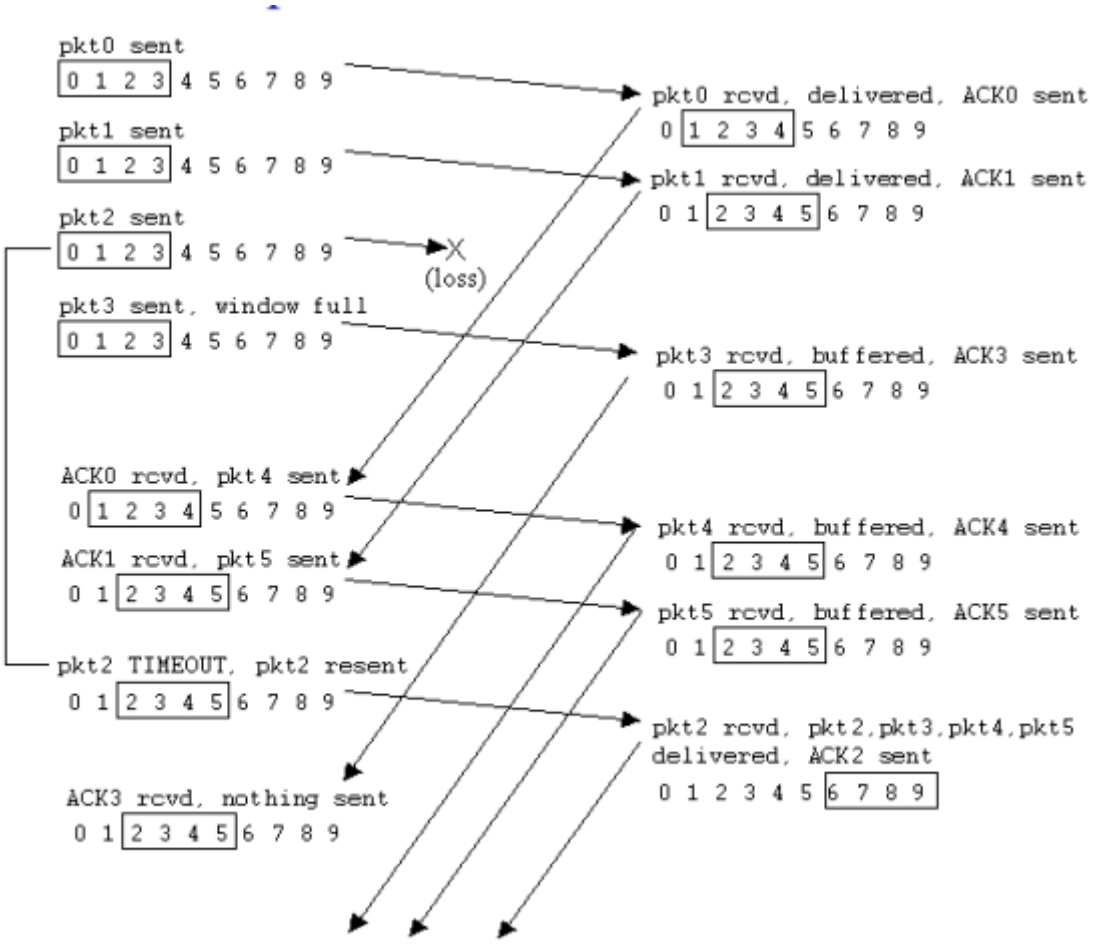
- ACK(n)

otherwise:

- ignore

- 发送方：
  - 如果下一个可发送的包在发送窗口中，则发送出去。
  - 收到timeout(n)时，重发数据包n，为其重设计时器。
  - 收到ACK(n)且n属于[sendbase,sendbase+N]时，将数据包n标记为已收到；如果n=sendbase，就滑动窗口，使saendbase=最小的未ACK的包的seq#。
- 接受方：（核心：对每个包分别单独确认）
  - 收到数据包n且n属于 [rcvbase, rcvbase+N-1]时，发送ACK(n)。
    1. 若是乱序包，则缓存此包
    2. 若是顺序包，则将所有按序且已缓存的包（若有的话）一同交付给应用层，滑动窗口到receivebase=下一个未收到的包。
  - 收到数据包且n属于 [rcvbase-N,rcvbase-1]
    1. 发送ACK(n)。
  - 收到其它位置的数据包，则丢弃。

SR过程图示



左边：

- 第五行收到ACK0后，0=sendbase，所以滑动窗口到下一个未确认的包1。第六行同理。
- 第七行pkt2超时后，直接重启timer，发送pkt2。

右边：

- 第三四五行，pkt3，4，5都属于乱序包，所以都缓存。

- 第六行，收到的pkt2是顺序包，所以交付给应用层所有的按序收到的包，即pkt2，3，4，5，然后滑动窗口到下一个未收到的包，即pkt6。

PS.这里**采用数据包标号**的seq#和ACK，和TCP中的**采用字节流标号**的seq#和ACK不同哦，注意区分！

DNS过程

（DNS使用的是UDP！）

DNS为什么采用分布式结构？（负载均衡）

- 可拓展性，可靠性，降低延迟

DNS过程：

2. DNS查询流程（以递归查询为例）

步骤1：本地DNS解析器查询

- **触发条件**：用户在浏览器输入 `www.example.com` 。
- **本地缓存检查**：
  - 操作系统和浏览器先检查本地缓存是否有该域名的IP。
  - **教材引用**：DNS缓存可减少查询延迟（《Top-Down》第7章）。

步骤2：向递归DNS服务器发起请求

- 若本地无缓存，向**本地DNS服务器**（如ISP提供的 `8.8.8.8` ）发送递归查询请求。
- **递归查询**：本地DNS服务器代表客户端完成所有查询工作（教材强调递归查询的"全权代理"特性）。

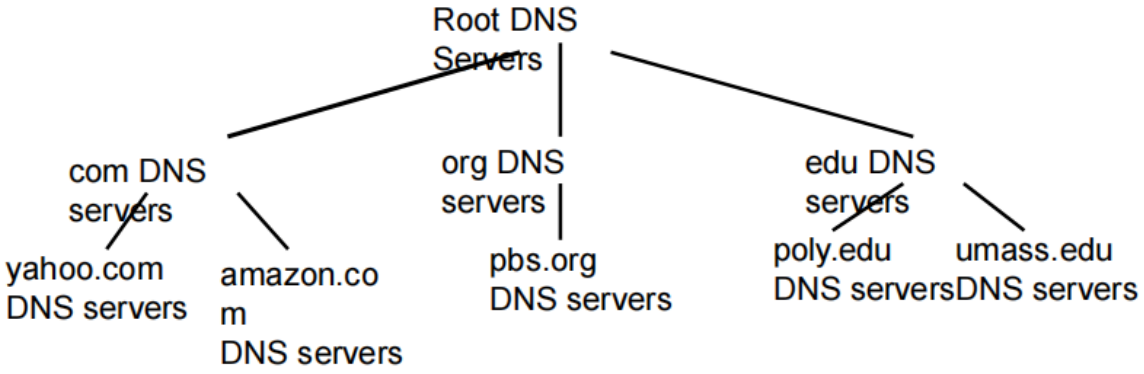
步骤3：迭代查询权威DNS服务器

1. **根DNS服务器查询**：
   - 本地DNS服务器向根服务器（全球13组）查询 `.com` 的顶级域服务器地址。
   - 根服务器返回 `.com` 的NS记录（如 `a.gtld-servers.net` ）。
2. **顶级域（TLD）服务器查询**：
   - 本地DNS服务器向 `.com` 服务器查询 `example.com` 的权威DNS服务器。
   - TLD服务器返回 `example.com` 的NS记录（如 `ns1.example.com` ）。
3. **权威DNS服务器查询**：
   - 本地DNS服务器向 `ns1.example.com` 查询 `www.example.com` 的A记录。
   - 权威服务器返回IP地址（如 `192.0.2.1` ）。

步骤4：结果返回与缓存

- 本地DNS服务器将IP返回给客户端，并缓存结果（根据TTL值）。
- **教材重点**：DNS缓存分层（本地→ISP→权威）显著提升效率（《Top-Down》图7.4）。

本地DNS服务器相当于代理，客户端都是先发给本地服务器，本地服务器转发到层次化DNS中。



Client wants IP for www.amazon.com; 1st approx:

- client queries a root server to find com DNS server
- client queries com DNS server to get amazon.com DNS server
- client queries amazon.com DNS server to get IP address for www.amazon.com

- 迭代查询iterated query指让本地DNS给每一层发送报文接收信息最后一起给主机
- 递归查询recursive query指让本地DNS按照上面的层次划分从上到下依次发送报文接收信息最后一起给主机

Circuit switching and Packet switching电路交换和分组交换的区别

注意和数据报网络/虚电路网络进行区分！

电路交换/分组交换出现在第一章，数据报网络/虚电路网络出现在网络层

术语	本质	典型代表
电路交换	独占物理路径，通信前建立端到端固定带宽连接（物理路径连接）	传统电话网络（PSTN）
分组交换	数据分块传输，动态占用链路，共享网络资源	互联网（IP网络）
虚电路网络	分组交换的子集，通信前建立逻辑连接（非物理路径）	ATM、帧中继（Frame Relay）
数据报网络	分组交换的子集，每个分组独立路由，无预建立连接	IP网络（传统网络，路由器转发）

- 电路交换：
  - **原理：频分复用/时分复用（FDM,TDM）**
  - 每一次呼叫都预留了固定带宽（**不考虑需求，预先分配链路的使用）**
  - 资源专用，不可共享
  - 性能有保障，**需建立连接**
- 分组交换：
  - **原理：存储转发机制**

- 数据分为分组，动态共享网络资源
- 每个分组**占用整个链路带宽**
- 可能因为排队出现拥塞

PS.虚电路和数据报网络的区别

特性	数据报网络	虚电路网络
服务类型	无连接	面向连接
路由器状态	无状态	有状态（保持连接信息）
数据包路径	可变路径	固定路径
建立连接	无需	必须
资源预留	无	可预留（带宽等）
代表技术	IP	ATM、Frame Relay、X.25

PS.虚电路网络：

- 每个分组携带**虚电路标识符（**Virtual connection identifier），而不是传统的IP地址。**
- 每条链路对应**VC编号（VC numbers）**
- **总结**：每个路由器根据入端口和VC编号查表后，将分组发送至相应出端口，并替换新的VC编号。**（Virtual connection identifier）**

router中的排队出现在哪里，为什么

- 输入端口排队：防止交换装置处理不过来。
- 输出端口排队：防止后继链路处理不过来。

CSMA/CD原理

r CSMA/CD:

- 先听再讲--载波监听
- 边讲边听—进行检测检测
- 冲突处理--发送站检测到冲突后立即停止帧的发送，并且发一个简短的堵塞信号(称强化冲突信号，Jamming signal)，通知网上各站已经发生冲突，本站及网上所有站都等待一段随机分布的时间，然后再按CSMA/CD方式重发该帧。

1. 网卡接收到网络层传来的数据报后，创建帧
2. 如果检测到信道空闲，开始发送；若信道忙，等待
3. 若成功发送完整个帧且未检测到冲突 → 完成发送
4. 若在发送过程中检测到冲突 → 停止发送，发送**堵塞信号（Jamming Signal）**

5. **二进制回退算法**（Binary Exponential Backoff）：
    1. **第 m 次冲突后，从集合 {0, 1, ..., (2^m) - 1} 中随机选择一个 K**
    2. **等待 K×512 比特时间后重新尝试**
    3. 冲突越多，退避时间越长
    4. 若冲突达到 16 次，则**放弃发送**

Principles of reliable data transfer (mechanisms: checksum, timer/timeout, sequence number, acknowledgement, pipelining)

发送端：

1. 应用信息分段 → 分配Seq# → 计算Checksum
2. 分组加入发送窗口 → 启动定时器（若未运行）
3. 窗口内分组全部发出 → 等待ACK
4. 收到ACK：
    1. 若ACK=期待值 → 窗口滑动，重置定时器
    2. 若ACK>期待值（冗余ACK） →三次重复后不等待超时，快速重传（Fast Retransmit）
5. 定时器超时 → 重传最早未确认分组

接收端：

1. 校验Checksum：失败 → 静默丢弃
2. 校验通过：
    1. 若Seq#=期待值 → 交付应用，回送ACK(期待值+1)
    2. 若Seq#>期待值（乱序） → 缓存分组，回送ACK(期待值) # TCP累计确认
    3. 若Seq#<期待值（重复） → 丢弃，回送ACK(期待值)

> ***“TCP如何实现可靠传输？至少列出4种机制并简述原理。”***
>
> 1. **校验和**：检测数据错误，错误则丢弃。
> 2. **序列号**：标识数据段，解决乱序/去重。
> 3. **累积确认**：ACK=N 表示所有≤N-1字节已接收。
> 4. **超时重传**：基于RTT估计启动定时器，超时重发。
> 5. **快速重传**：收到3次重复ACK立即重传（不等待超时）。

> ***如何设计一个可靠数据传输协议?***
>
> 数据完整性+传输可靠性（+安全性）
>
> 1. 数据完整性：
>   1. 校验和（比特正确）
>   2. 序列号，数据分片与重组（顺序正确）
> 2. 传输可靠性：
>   1. 确认应答（ACK）、超时重传等（不漏传）

> 2. 流量控制和拥塞控制（不丢包）
> 3. (安全性)
>   1. 加密算法，身份认证等（不被攻击）

简述四种时延

1. **四种延迟**
   1. **nodal processing delay（定性）**
      - **（检查位错、决定输出链路）**
   2. **queuing delay（定性，波动会很大）**
      - **输出链路拥堵时等待传输，波动大，由路由器拥塞情况决定。**
      - 场景：跨国传输，这个影响最大。
   3. **transmission delay（定量）**
      - **将一个数据包推送到链路上所需时间 = L/R，包长度（bits）除以链路带宽（bps）**
      - 场景：对于慢传输速率的网络影响最大。
      - $$\begin{aligned} d_{trans} &= \text{transmission delay} \\ &= L/R, \text{ significant for low-speed links} \end{aligned}$$
   4. **Propagation delay（定量）**
      - **传播距离/速度**
      - 场景：提到卫星链路时，这个影响最大。

hub，switch，router的不同

- hub属于物理层，将收到的信号**广播**到所有端口，所有设备在同一个冲突域中，所有端口都可能冲突，无CSMA/CD，无帧缓存，已经被switch取代。

router：

- 属于网络层
- 核心功能：
  - **Routing： 运行路由算法与协议**（如 RIP、OSPF、BGP）
  - **forwarding****：将接收到的数据报从输入链路转发到输出链路**
  - 隔离广播域（不转发广播帧），不隔离冲突域
- 寻址方式：
  - 基于IP地址

switch：

- 属于链路层
- 核心功能：
  - **Forwarding**：（存储与转发帧）基于 **switch table**选择性的将帧发送到目标端口，是透明的，主机感觉不到交换机的存在。
  - Use CSMA/CD to access segment

    - Switch table的结构是：
    - MAC地址+对应的interface+TTL
    - **Filtering**：仅将帧发送给目标设备（非广播）。
    - **Self-learning**：自动构建switch table（也可可以叫forwarding table）。
    - **Plug-and-play**：无需配置，上电自动学习。
    - 隔离广播域和冲突域，每个端口独享链路，无碰撞，全双工
    - *“全双工模式下，交换机的端口无需竞争信道，彻底消除冲突（Collision）。”*
- 寻址方式：
    - 基于MAC地址

switch交换机的forwarding和filtering功能的讲解：

Hierarchical Routing： Intra-AS/Inter-AS routing and hot-potato routing.

Hierarchical

- 分层路由是为了解决 **互联网规模太大、每个路由器不可能维护全球所有路由信息**的问题而设计的。它将互联网划分为多个自治系统（**AS，Autonomous Systems**），在每个系统内和系统间分别采用不同的路由机制。

Intra-AS Routing（自治系统内路由）

- OSPF和RIP
    - 寻找 **AS 内部**某个目的地的最优路径
    - 向 AS 中所有路由器 **传播网络拓扑信息**
    - 保证系统内路由的 **收敛性、一致性、效率**

Inter-AS Routing（自治系统间路由）

- BGP
    - 控制 AS 之间的 **前缀传播和可达性信息**
    - 支持 **策略性路由**（如选择某个商业关系更有利的路径）
    - 每个 AS 可决定：
        - 向哪些邻居通告哪些前缀
        - 选择从哪些邻居学习的前缀加入路由表

Hot-potato Routing（热土豆路由）

- 一种 Inter-AS 路由策略 —— **尽早把流量“扔出去”给别的 AS，让对方去负责后续转发。**
    - **路径选择时不考虑全局最短路径，只看如何最早脱手**
    - 通常选用最近的出口路由器（egress point）
    - 可减少本 AS 的内部资源消耗（带宽、负载）

五层协议是什么，分别有什么作用

- **应用层（Application Layer）**
  - 直接为用户应用程序（如浏览器、电子邮件、FTP等）提供网络服务。
- **传输层（Transport Layer）**
  - **端到端通信**：确保数据在两台主机的进程间可靠传输（如TCP）或高效传输（如UDP）。
- **网络层（Network Layer）**

路由选择：决定数据包从源到目标的路径

- **数据链路层（Data Link Layer）**
  - **物理寻址**：数据链路层负责通过链路将数据报从一个节点传送到相邻的节点。
- **物理层（Physical Layer）**

比特流传输：将数据链路层的帧转换为物理信号

为什么IPv4地址要被IPv6替换？为什么还没有被替换完?

- 为什么要替换?
  - IPv4使用 **32位地址**，理论上有约 **43亿（2^{32}）** 个地址，但实际可用地址更少
  - 随着智能手机、物联网（IoT）等设备的爆炸式增长，IPv4地址早已不够用
- 为什么没替换完?
  - 旧设备兼容性低，过渡成本高
  - NAT技术和CIDR技术缓解了IPv4短缺。

中国本地没有根域名服务器，如果美国禁止我们访问，会怎么样？我们应该如何解决这个问题?

根域名服务器负责管理全球的顶级域名服务器（LTD）。

未缓存的国际域名解析将会失败，部分海外网站无法访问。

解决方法：（源自AI）

1. 减少依赖
   1. 部署根服务器镜像（如F、I、J、L根的国内镜像）。
   2. 完善国内DNS基础设施，增强冗余机制。
2. 加强本地网络自主性
   1. 自主建设本地根域名服务器，推进IPV6根服务器的建设。

TCP/IP and ISO OSI model Protocol layers/stacks and service models

(分层架构，各个层次的基本功能,依赖下层服务，为上层提供服务的思想)

> **网络通信系统被设计为一组“分层的模块”，每一层通过“抽象服务接口”与上下层交互。**

核心理念：

- 每一层 **依赖下层提供的服务**；
- 同时向上层 **提供更高级的抽象服务**；
- 各层间 **只关心接口和功能，不关心具体实现。**

为什么分层?

模块化：有助于管理复杂系统，各层相对独立，专注于解决自己的问题

抽象化：上一层不关心下一层的具体实现，通过抽象服务接口交互。

标准化：层之间使用明确定义的接口和协议

举例是哪五层然后讲一下每层的功能应该就够了

第五章最后a day in the life of a web request个人感觉挺适合出个简答题的。

插网后访问www.example.com过程为：

1. 客户端广播DHCP discover（使用UDP，目标IP 255.255.255.255），服务器回复DHCP ack，携带客户端自己IP，本地DNS服务器IP，下一跳路由器IP信息。
2. 客户端广播ARP query（目标MAC FF.FF.FF.FF.FF.FF）获得DNS本地服务器MAC地址
3. 本地DNS服务器迭代查询获得目标网站的IP地址
4. 客户端打开80端口socket，使用三次握手建立TCP连接
5. 进入页面请求阶段，客户端发送HTTP request，服务器回复HTTP reply。
6. 传输完整个页面内容后四次挥手关闭TCP连接。