**计算机网络和因特网**

**什么是因特网**

**构成描述/服务描述**

**构成描述**：连接数以亿计的计算设备(被称为主机/端系统) 这些设备通过通信链路和分组交换机(也叫报文交换设备:

路由器、链路层交换机)连接到一起，连接的传输速率以bps度量。端系统通过因特网服务提供商(ISP)连接在一起（所以因特网是网络的网络，是由ISP网络互联而成）。网络协议控制报文信息收发（TCP、IP等等）。因特网标准（IETF->RFC）

**服务描述**：向网络应用提供服务的基础设施、向网络应用提供编程接口

协议：规定了网络实体间消息交互的格式、顺序、收发消息后执行的动作等

**网络边沿**

端系统(=主机)被细分为客户和服务器

**接入网**：将主机连接到其边缘路由器的网络 家庭接入有(DSL)数字用户线/电缆/FTTH(光纤到户)/拨号和卫星 公司企业或家庭接入有以太网/WIFI，它们都是LAN(局域网)技术，以太网和WIFI以前只被大型组织应用，现在它也被广泛应用在家庭之中 广域无线接入4G等

**主机：数据包发送**：将消息分为小的块，称为数据包或报文，每个不超过L比特，以传输速率R将数据包发到链路上数据包传输时延 = L（bit）/ R（bit/s）

**物理介质**：双铜绞线(TP)/同轴电缆（同心的两个铜导体）/光纤/无线电波（地面微波、无线局域网、广域、卫星）

**网络核心**

**网络核心部分的两大关键功能** 路由-决定从源到目的地的网络路径，由路由算法完成 转发-将数据包从路由器的输入端口移到输出端口

**两种基本方法（分组交换和电路交换）**

**分组交换** 报文(message)被划分为小的数据块称之为分组(packet)。在源和目的地之间，分组通过通信链路和交换机（路由器、链路层交换机）传送。存储-转发 以R bps速率将L-比特大小的数据包发送到链路上，耗时L/R 秒。路进程标识包括主机的IP地址和进程在主机上分配的端口号.由器必须完整接收到一个数据包，才能将其转发给下一跳路由器。端到端时延=N×L/R，N为跳数，有N-1个路由器排队时延、丢包 排队时延是等待链路空，丢包是缓存已满（新到达的数据将被丢弃）

**电路交换** 提前预留资源，电话网采用这种方式 实现预留资源的方法-复用（频分复用FDM或时分复用TDM）例子：主机A通过TDM链路向主机B传送一个640,000bit大小的文件。TDM链路被分为24个时隙，总带宽1.536Mbps。主机之间建立端到端线路需要500 msec。问需要多长时间完成文件传输？-每条电路带宽(1.536 Mbps)/24 = 64 kbps，传输640,000bit需要(640,000 bits)/(64 kbps) = 10 sec。加上建立线路的0.5秒，共需10.5秒。

**分组交换 vs. 电路交换** 分组交换可以容纳更多的用户使用网络 是不是分组交换一定优于电路交换？No. 分组交换适用于突发数据，数据太多、资源不够时，产生拥塞。

**网络的网络** ISP们自身必须互联，才能互相发包，因此逐渐形成了“网络的网络”，由经济和国家政策因素推动演进。多个全球ISP形成商业竞争关系，它们之间互联。地区网络向接入网络提供连接到全球ISP的服务。内容提供商（例如，谷歌、微软、亚马逊）构建自己的网络，靠近用户提供服务。第一层ISP。

**分组交换网中的时延、丢包和吞吐率**

**时延** 为何发生丢包和时延-数据包在路由器缓存中排队，数据包到达速率（暂时）超过出口链路的容量时，数据包排队等待被传输。时延的四种类型：节点处理时延dproc，排队时延dqueue，传输时延dtrans，传播时延dprop。

dproc: 通常一毫秒以内完成。dqueue: 排队等待被转发链路传输,取决于队列长度（拥塞程度）

dtrans: L: 数据包大小(bits) R: 链路带宽(bps) dtrans = L/R。dprop：d: 物理链路的长度 s: 信号传播速度(大约2x108 m/sec) dprop = d/s。在每一跳上的时延dnodal = dproc + dqueue + dtrans + dprop

传输时延和传播时延的区别：收费站模型 | 排队时延：a表示分组到达队列的平均速率，R是传输速率，L是分组的大小(bit)，La/R 称为流量大小 ： La/R ~0: 平均排队时间短（队列基本为空） La/R -> 1: 平均排队时间长

La/R > 1: bit到达速率超过离开速率，平均排队时间无穷大（如果缓存无限大）。关于La/R ->1 ：如果每隔L/R秒到达一个数据包-走一个包，来一个包。队列一直为空，无排队时延。如果每隔NL/R秒到达N个数据包，即突发到达

第1个包不排队 … 第N个包排队(N-1)L/R秒。Traceroute 程序: 提供一条从发送源到目的地路径上距离所有路由器的往返时延测量结果。

**丢包** 出口链路的队列（或缓存）空间有限，数据包到达时发现队列已满，路由器丢弃数据包，被丢弃的数据包可能会被上一跳路由器重传，或被发送端重传，或根本不被重传.

**吞吐量(吞吐率)** 发送端到接收端的比特传输速率（比特/单位时间，例如，bps）。瓶颈链路-端到端路径上最小速率的链路，限制端到端吞吐率

**协议层次及其服务模型**

**协议分层** 每层通过在该层中执行某些动作或使用直接下层的服务来提供服务。各层的所有协议被称为协议栈。因特网的协议栈由5个层次构成：物理层、链路层、网络层、运输层、应用层。（书按照自顶向下处理）

应用层: 支持网络应用FTP, SMTP, HTTP应用层报文被称为消息（message）

传输层: 进程之间的数据传输TCP, UDP进程到进程 传输层报文被称为分段（segment）

网络层: 实现从源主机到目的主机的数据包路由IP, 多种路由协议 设备到设备 网络层报文被称为数据报(datagram)

链路层: 介质链路直连的相邻网元（网卡）之前数据传输 Ethernet, 802.11 (WiFi), PPP（点对点） 网卡到网卡 链路层报文被称为帧（frame）

物理层: 在各种介质链路上传输bit 介质不同，信号形态不同，脉冲、电波、高低电平

OSI参考模型：应用层、表示层、会话层、运输层、网络层、数据链路层、物理层

表示层：支持应用层对数据含义的处理,例如, 加解密、压缩解压、 机器转码等

会话层: 同步、检查点（checkpoint）、数据恢复

因特网协议栈在应用层按需实现这些功能，不是所有情况下都需要

**封装** 数据从发送端系统协议栈向下，沿着中间的链路层交换机和路由器协议栈上上下下，然后向上到达接收端系统的协议栈。每一层数据包包含头部和负载。上层的数据包是下层协议数据包的负载。链路层交换机只能实现下面两层，路由器实现下面三层。

**网络攻击和安全**

**网络安全包括**:攻击者如何攻击网络/如何防御攻击/如何设计对攻击免疫的网络体系结构（内生安全）

因特网设计之初没有（太多）安全考虑,通过“亡羊补牢”的方式增加安全性,在各层增加安全机制!

**恶意软件的攻击** 主机感染恶意软件的方式**-**病毒: 通过主动可执行对象自我复制 (例如，邮件附件)蠕虫: 被动接收到自我复制的程序恶意软件记录键盘敲击、网站访问地址等信息，上传到信息收集站点 被感染的主机可以被黑客操控为僵尸网络，用于传播垃圾信息（spam）、发起分布式拒绝服务攻击（DDoS）

**对服务器和网络基础设施的攻击**：拒绝服务攻击-攻击者通过伪造大量流量耗尽攻击目标的带宽、服务能力等资源，达到瘫痪攻击目标的目的。1. 选取目标 2. 侵入目标周围的主机（僵尸网络）3. 从被侵入的主机发送攻击数据包

**数据包嗅探** 针对广播性质的传播介质(共享以太网、无线网) 混杂模式下网络接口可以读取链路上传输的所有数据包（包括其它用户和主机的数据包）

**地址欺骗** IP地址欺骗：使用假冒地址作为源地址发送数据包

**计算机网络和因特网的发展历史**

分组交换的发展 1961 – 1972 (ARPAnet，典型的私有网络，70年代有很多其它分组私有网络，它们不互通)

-> 专用网络和网络互联 1972-1980 (TCP/UDP/IP概念完成,以太网) TCP/IP的意义 网络细腰结构

TCP/IP的局限 以主机间通信为首要目标，以设备地址作为网络的核心，不适应现在以内容、服务为核心的互联网商业模式

-> 网络的激增 1980 – 1990 万维网(WWW/Web)应用程序的出现

**应用层**

**应用层的原则**

**网络应用的体系结构** 客户端服务器结构、对等网络(P2P)结构。客户端服务器结构-服务器运行在永远在线的主机上，永久IP地址。客户端与服务器通信，断续地连接网络，可以使用动态地址，彼此之间不直接通信。对等网络结构-没有永远在线的服务器程序，任何终端系统直接通信，peer向其他peer请求服务，并且服务于其它peer，可扩展性

**进程通信** 进程: 主机上运行中的程序-通信的两个进程在同一个主机上，进程间通信, 操作系统的范畴。通信的两个进程在不同主机上，通过相互交换消息报文实现通信，网络范畴。 将发起通信的一方成为客户端进程，等待被连接的一方称为服务器进程。对等网络应用同时具有服务器进程和客户端进程。 进程与计算机网络之间的接口-套接字。进程通过其套接字（socket）收发报文消息，一个比喻：门。 标识进程：为接收报文消息，进程必须有一个标识ID

主机设备有一个32-bit的IP地址。

**应用程序需要的传输层服务** 从四个方面对应用程序服务要求进行分类-数据完整性(可靠数据传输)、吞吐率、时延管理(定时)、安全性。数据完整性：有的需要可靠数据传输，有的可以容忍数据丢失。吞吐率：带宽敏感的应用、弹性应用。时延管理：网游要求时延低于特定值。安全：加密、数据完整性

**因特网提供的运输服务** （TCP/UDP）

TCP 在收发进程间的可靠数据传输 流控制: 发送端的数据发送速率不会超过接收端能力 拥塞控制：网络过载时限制发送端的发送速率 不提供: 时延保障、最低吞吐率保障、安全（SSL加强了TCP的安全性，位于应用层，应用程序使用SSL库与TCP协议交互） 面向连接: 收发进程需要预先建立连接

UDP UDP服务: 在收发进程间不可靠的数据传输 不提供: 可靠性、流控制、拥塞控制、时延保障、吞吐率保障、安全、连接建立 （轻量级，避开TCP的拥堵机制和分组开销）

因特网运输协议不提供吞吐率和时延保障的相关服务。

**应用层协议** 应用层协议定义如下内容：消息类型、消息词法、消息语义、规则。有开放协议和私有协议。

**Web和HTTP、FTP**

**HTTP概况** Web页面由对象构成。Web页面包含一个基础的HTML文件，文件中包含对多个对象的引用，引用每个对象可用URL寻址。HTTP(超文本传输协议)是Web服务的应用层协议。Web使用了客户-服务器应用程序体系结构。HTTP如何使用TCP: 客户端进程创建套接字，向服务器进程的80端口发起连接请求；服务器进程接收来自客户端的TCP连接请求；客户端（浏览器进程）和服务器（web server进程）交换HTTP消息(应用层协议报文)；关闭TCP连接。HTTP是无状态协议（服务器不记录客户端以前的请求）

**HTTP连接(非持续连接和持续连接)**

非持久HTTP一个TCP连接最多传输一个对象 传输完成，关闭TCP连接 下载多个对象时，需要建立多个TCP连接

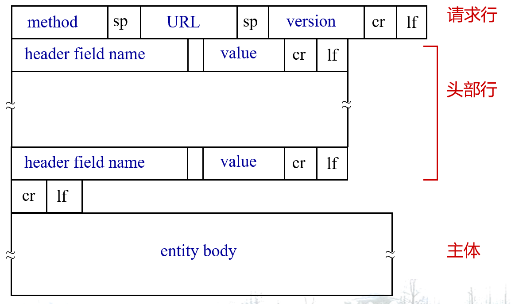
持久HTTP 多个对象可通过客户端和服务器之间的一个TCP连接传输

往返时延RTT (定义): 一个小数据包从客户端到服务器再返回客户端所需时间 HTTP响应时间: 一个RTT用于建立TCP连接 一个RTT用于发送HTTP请求并开始收到HTTP响应 文件传输时间

非持久HTTP响应时间=2RTT+文件传输时间

持久HTTP响应时间 非流水线 RTT+N×RTT 流水线RTT+RTT

**HTTP报文（消息）格式** 两种HTTP消息类型:请求、响应

 请求消息：格式 上传输入：POST方法（输入被放在实体即主体中上传）、URL方法 （使用GET命令，输入框输入被放在请求行的URL字段上传） 其他命令：HEAD（类似GET，但仅返回消息头部，不返回实体对象） PUT（上传）DELETE（删除）

Host: 存放对象的主机 Connection:

close: 关闭持久连接 keep-alive: 指定时间内维持TCP连接

User-agent: 浏览器名称和版本

Accept-language:..

响应消息：初始状态行（协议版本字段 状态码 响应状态信息）、首部行、实体体

状态代码和短语: 200 OK 请求成功，被请求对象在此消息的数据部分 301 Moved Permanently 被请求的对象移走了，它的新位置（URL）在消息的Location头部行 400 Bad Request 请求消息不被服务器理解 404 Not Found在服务器上没有找到所请求的对象 505 HTTP Version Not Supported

**用户在服务器的状态——cookie** cookie由四部分构成：1) HTTP响应消息中包含cookie头部行 2) 下一个HTTP请求消息中包含cookie头部行 3) 用户在主机的浏览器中上持有一个cookie文件 4) 网站的后端数据库存储用户的cookie信息 作用：认证、购物车、推荐、保持用户会话状态。也会产生隐私问题。

**Web缓存（也叫代理服务器）** 目的： 在不访问源服务器的情况下满足客户端请求 用户设置浏览器通过代理缓存访问web 浏览器将所有HTTP请求发给缓存 若对象在缓存中，直接返回给客户端。否则向源服务器请求对象并存档返回。代理服务器同时扮演客户端和服务器角色。作用：缩短客户端请求的响应时间 降低机构（大学、企业）接入链路的流量 缓存大量部署在因特网上：有助于用户访问那些无法部署大量源服务器的内容提供者发布的内容(P2P文件共享有类似效果) 条件GET：检查缓存中对象是否过时，“If-modified-since:<date>”

**FTP(文件传输协议)** 客户端服务器模式 ftp服务器工作在TCP 21端口上**。**FTP客户端在21端口与FTP服务器建立TCP连接，该连接是控制连接，客户端通过控制连接认证，客户端通过控制连接发送命令，浏览远程文件目录 当服务器收到文件传输命令， 建立第二条TCP连接用于文件传输，该连接是数据连接，服务器端口号是20当完成一个文件的传输，服务器关闭数据连接。（这是一种“带外控制”，HTTP是“带内控制”：一条TCP连接同时是控制和数据）

**电子邮件**

三部分构成：用户代理软件、邮件服务器、简单邮件传输协议SMTP 用户代理：用于书写、编辑、阅读邮件消息 邮件服务器：邮箱存放收到的用户邮件消息 消息队列-待发出的邮件消息 邮箱服务器之间使用SMTP协议传输邮件消息 客户端-发送邮件的服务器 服务器端-接受邮件的服务器

**SMTP** 使用TCP在25端口号上从客户端到服务器端可靠地传输邮消息 服务器到服务器直接传输 传输邮件的三个步骤：握手件-传输邮件消息数据-关闭 通过命令/响应交互(类似HTTP) 命令: ASCII文本 响应: 状态码和短语 邮件消息（包括标题、正文、附件）使用7-bit ASCII 编码

**与HTTP对比** 都使用持续连接 都是用ASCII编码的命令和响应消息，都有状态码 HTTP-拉协议 每个对象封装在一条响应消息里发送 SMTP-推协议 多个对象在一条包含多个部分的消息中发送。

**邮件消息格式** （区分SMTP握手协议中的命令）“头部行 空行 主体” 头部行To: From: Subject: 主体：邮件内容（只用ASCII编码

**访问邮件的协议** SMTP是推协议，接收方需要通过别的协议将邮件从邮件服务器传送到接收方的用户代理。POP3/IMAP/HTTP POP：认证阶段-事务阶段-更新 两种模式“下载并删除/下载并保持” 无状态，因此非常简化。IMAP： 所有邮件消息存放在邮件服务器，允许用户使用文件夹在服务器上组织邮件，在多个会话间保持状态：文件夹的名称和邮件与文件夹的映射关系在多个客户端上保持一致。HTTP：基于Web的电子邮件广泛使用HTTP来和邮件服务器接发邮件（邮件服务器之间仍然使用SMTP）

**DNS（domain name system）：域名系统，因特网的目录服务**

**DNS提供的服务** （UDP协议传输，端口53）将主机名翻译为IP地址/映射主机别名/邮件服务器别名/负载均衡

**DNS工作机理** 集中式DNS会导致单点故障、流量大、距离远解析时间长、难以维护->不具有可扩展性。因此DNS采用了分布式的设计方案（分布式分层数据库）- 根域名服务器/顶级域名(TLD)服务器/权威域名服务器 本地域名服务器并不属于上述层次结构 迭代查询：被询问的服务器返回下一步需要询问的服务器名 递归查询：解析的通信负载在域名服务器上 层次越高，负载越大。实际解析过程：主机到本地域名服务器是递归的，剩余过程是迭代的。

**DNS缓存** 域名服务器缓存所有获取的名字映射记录 一段时间后过期删除。如果主机更改了IP地址，则需要等待所有相关缓存过期，该主机才能被整个因特网访问。

**DNS映射记录** RR（分布式数据库存储资源记录）格式(name, value, type, ttl) type = A name是主机名，value是ip地址 NS name是域名 value是这个域的权威域名服务器的主机名 CNAME name是主机别名 value是规范名 MX value是name表示的邮件服务器的规范名

**DNS协议和消息** 两类消息：查询（query）和回复（reply），格式相同。将RR添加到DNS：在注册服务商注册域名，注册服务商向.com顶级域名服务器添加两条RR 一条NS一条A。然后机构在权威域名服务器上创建A和MX RR

**对DNS的攻击** DDoS攻击 重定向攻击

**UDP和TCP套接字编程**

套接字（socket）：应用程序进程之间端到端通信的“门”

两类套接字对应两类传输层服务: UDP: 不可靠的数据报 TCP: 可靠的byte流

UDP：客户端与服务器之间无连接，传输之前无需握手，发送每个数据包时明确IP和端口号。数据包可能丢失或乱序从应用程序角度: UDP 提供了一种不可靠的客户端和服务器之间的数据传输方式（以一个个数据报datagram的形式）

TCP：从应用程序角度，TCP提供了客户端和服务器端之间一种可靠、顺序的字节流管道。三次握手。服务器段有欢迎套接字（用于握手）和连接套接字（用于数据传输）

**传输层**

**传输层提供的服务**

传输层协议在不同主机上的应用程序进程之间提供逻辑的信道。传输层协议（软件实现）运行在终端系统上-发送端：将应用程序的数据划分为分段（segment），交给网络层 接收端：重组分段形成数据，交给应用层。因特网传输层：TCP/UDP 传输层和网络层关系：网络层协议提供主机间的逻辑信道，传输层提供进程之间的逻辑信道，基于网络层服务来实现。（网络层是邮政服务，传输层是Ann和Bill）

**复用和解复用**

在发送端复用（处理多个套接字传来的数据，添加传输层协议头部（用于解复用）），接收端解复用（基于传输层协议头部信息将分段交给相应的套接字）解复用工作原理：主机使用IP地址+端口号将分段送到相应的套接字。无连接解复用（UDP）创建套接字时绑定本地端口号、构造报文由UDP套接字发送时，必须指定目的IP地址和目的端口号，具有相同目的端口号，但来自不同源IP地址的报文被送到目的主机上相同的套接字。面向连接的解复用（TCP）：用一个四元组（源IP地址，源端口号，目的IP地址，目的端口号）来标识，来自不同IP地址的同一端口号的报文进入不同套接字。

**无连接的传输层协议：UDP**

朴素，尽力而为，可能丢失乱序，无连接。应用：流媒体、DNS、SNMP（简单网管协议）。可由应用层提供可靠传输。

为何需要UDP: 无需建立连接（建立连接产生时延） 简单：收发双方无需维护状态 协议头部小 没有拥塞控制（快）

**分段头部** 32byte 源端口号、目的端口号、长度（包括头部）、校验和

**校验和** 目的：检测收到分段中的错误 发送端对分段内容求和取反（注意回卷，即把进位1加到最后一位去） 接收端基于分段内容求和+校验和 如果全’1’-未检查出错误（仍然可能有错）包含‘0’-检测到错误 伪头部：UDP计算校验和时，包含一部分IP头部的字段。 很多链路层也提供了差错校验，为何在UDP计算校验和？不保证源到目的路径上的所有链路都实现差错校验、在路由器内部也可能发生bit翻转 –端到端原则

**可靠数据传输的原理**

可靠数据传输的实现在应用层、传输层、链路层都非常重要。网络领域十大问题之一。仅考虑数据单向传输（控制信息可以双向传输）用FSM描述发送端和接收端的状态行为。

rdt1.0 信道完全可靠 发送端和接收端各自用一个FSM描述

rdt2.0 有bit翻转的信道 ACK和NAK（停等协议） 有缺陷：当ACK/NAK发生bit翻转 发送端不知道接收端是否正确收到数据包 不能简单地重传，可能导致重复发包 解决办法：当ACK/NAK错误时发送端重传数据包 发送端为每个数据包添加一个序列号 接收端忽略丢弃重复收到的数据包（相同序列号的包已收到过了） -> rdt2.1 问题：两倍的状态数量 -> rdt2.2 实现2.1相同功能，但不用NAK：当收到包含错误的数据包，接收端重复发送上个正确接收的数据包的ACK

rdt3.0：信道包含差错和丢包 应对措施: 在2.2基础上，发送端等ACK一段“合理”的时间, 如果超时未收到ACK，重传数据包 其发送端FSM引入计时器，接收端FSM与2.2相同。 性能：停等协议功能正确，但是性能差。信道利用率低：(L/R)/RTT+(L/R) -> 流水线传输

流水线传输: 发送端允许多个数据包“在路上”“未确认”, 增加序列号的空间，在发送端/接收端暂存数据包 处理差错的方法：回退N步/选择重传 回退N步-包头部携带k-bit的序列号。N是窗口长度，GBN协议也被称为滑动窗口协议(base在已被确认的下一个，nextseqnum发送还未被确认的下一个，且大于等于base+N的序号不可用) 接收方对于乱序到达的包丢弃，并重复确认收到当前顺序到达的最大的包。 选择重传-两难困境：无法区分重传数据包和新数据包 解决方案：窗口大小不应超过序列号空间的一半 比较：回N-优点：发送端仅维持一个计时器，接收端不用缓存乱序到达的数据包，实现起来简单。缺点：浪费带宽 选择性重传优点：不浪费带宽 缺点：发送端维持多个计时器，接收端需缓存乱序到达的数据包，实现复杂。

**面向连接的传输层协议：TCP**

端到端，可靠顺序，流水线传输，全双工，面向连接，流控制（发送速率不会压倒接收端速率）

**分段结构** 首部一般是20字节：源端口号 目的端口号 序列号 确认号 首部长度 保留未用 CWR ECE URG ACK PSH RST SYN FIN 接收窗口 校验和 紧急数据指针 选项 数据

序列号：分段中第一个byte在整个byte流的次序编号 确认号：期望收到的下一个byte的序列号（累计确认）如何处理乱序到达的分段？标准没有规定，取决于具体时间，多数时候缓存并等待缺失数据。

**TCP往返时延和超时** 设置超时时长：比RTT长 设置过短:-分段到达前超时，导致不必要的重传 设置过长-分段丢失时反应迟缓 测算RTT：SampleRTT，忽略重传，平均 TimeoutInterval = EstimatedRTT + 4\*DevRTT

**可靠数据传输** 导致重传的事件：超时、重复确认。发送方有3个与发送和重传相关的主要事件：从上层应用程序接收数据、超时、收到ACK。 快速重传：按之前的方案，超时需要等较长时间，重发的时延因此很长。现在，如果发送端收到3个相同确认号的ACK，立刻重发尚未确认的最小序列号的分段，无需等到定时器超时。TCP的差错恢复机制是GBN和SR的混合体。

**流控制** 接收端控制发送端，使其发送速率不会太快，不会造成接收端缓冲区溢出。具体实现：接收端在从接收端到发送端的TCP分段头部rwnd字段里，告知发送端其当前空闲的缓冲区空间大小。发送端控制窗口，使得已发出未确认的分段（即“在路上”的分段）不超过rwnd值。接收端维持两个变量 LastByteRead: 最后一个提交给应用进程的byte序列号 LastByteRcvd: 最后一个到达的byte序列号 rwnd = RcvBuffer - [LastByteRcvd - LastByteRead] 发送端维持两个变量 LastByteSent：最后一个发出去的byte序列号 LastByteAcked：最新确认的byte的序列号 确保 LastByteSent – LastByteAcked ≤ rwnd 当接收端套接字缓冲区满 告知发送端rwnd=0 发送端停止发数据产生的问题：接收端缓冲区空了以后，没有数据需要确认，无法通知发送端继续发数据 解决办法：当接收端rwnd=0，发送端继续发送1 byte大小的分段

**连接管理** 三次握手 发送SYN Seq(初始序列号) ACK ACKnum(传过来的序列号+1) 关闭连接 FIN比特位置1 两边互发互ACK 拒绝连接（套接字不对）RST置1

**拥塞控制原理**

**拥塞的原因与代价** 3种复杂性越来越高的情况 理想无重传 -> 重传（只重传丢失的包/超时即重传）-> 4个发送端多跳路径。 拥塞的两个代价： 1. 为实现给定的有效吞吐率（goodput），需要做额外的重传。不必要的重传，导致一个数据包的多个副本被送到接收端降低goodput 2. 当拥塞导致丢包，数据包传输消耗的上游带宽被浪费了。

**TCP的拥塞控制**

cwnd（拥塞窗口）加性增，乘性减->锯齿状变化。TCP发送端控制LastByteSent-LastByteAcked<min(rwnd,cwnd) rate≈cwnd/RTT bytes/sec

**两种TCP拥塞控制算法 Tahoe（旧）和Reno（新）**

**慢启动** 指数增大cwnd直到丢包发生（如何实现？每收到一个ACK，将cwd增加1个MSS）一开始速率很低，但是指数增长，慢启动并不慢。检测丢包：超时（则cwnd降为1）/3个重复ACK（cwnd降为一半然后线性增长，因为能收到重复ACK说明网络仍有一定带宽，即后面的包都收到了，这是Reno） 然而Tahoe在两种情况下都降为1 。

**由慢启动切换到拥塞避免** 到达一个门限（ssthresh = cwnd/2 ），然后线性增长（拥塞避免状态）。

**拥塞避免** 每个RTT增加cwnd一个MSS，具体实现-每收到一个新的ACK，发送端cwnd增加MSS\*(MSS/cwnd)，等价于每个RTT增加cwnd一个MSS（线性增长）。结束：ACK超时，则ssthresh = cwnd / 2, cwnd = 1 MSS，进入慢启动状态（Tahoe和Reno）。收到三个重复的ACK，则ssthresh = cwnd/2, cwnd = ssthresh+3，进入快速恢复状态 (Reno)

和ACK超时一样(Tahoe)

**快速恢复** 继续收到重复的ACK，每次增加cwnd一个MSS。收到新的ACK，设置cwnd = ssthresh ，进入拥塞避免状态。ACK超时，ssthresh = cwnd / 2, cwnd = 1 MSS，进入慢启动状态。

**TCP吞吐率** TCP的平均吞吐率由窗口大小和RTT决定（忽略慢启动，假设应用层一直有数据待传输）。平均TCP吞吐率 = 3/4 \* W/RTT bytes/sec （W是丢包时的窗口大小）

**在“长、肥”管道上的TCP吞吐率** 吞吐率取决于传输路径的丢包率L TCP吞吐率 = 1.22 \* MSS / RTT\*根号L MSS为最大报文段长度

**TCP的公平性** 公平性目标: K个TCP会话共享一个瓶颈链路，带宽为R，则每个连接的平均吞吐率应为R/K。为何公平：两个TCP会话竞争-加性增导致当两个连接的吞吐率增加，沿斜率1上升，乘性减导致x和y坐标减半。 UDP和公平性、多个并发TCP连接的公平性。显式拥塞通知（ECN）：网络辅助拥塞控制:发生拥塞的路由器设置IP头部两个bit (ToS字段)，接收端收到携带指示的IP报文，接收端在ACK分段中设置 ECE比特位，通知发送端路径上存在拥塞，发送端将cwnd减半，在下一个分段设置CWR比特位