***第1章：计算机网络和因特网***

**什么是因特网：**

具体构成描述：终端（接入因特网的设备，运行网络应用程序）称**主机**或**端系统**，通过**通信链路**和**分组交换机**连接到一起。 链路的**传输速率**或**带宽**以比特每秒（bps）度量。 当端系统向另一端系统发送数据时，发送端系统将数据分段并加上首部字节形成信息包（叫做分组）。分组交换机从其一条入通信链路接收到达的分组，并从其一条出通信链路转发该分组，最著名的分组交换机是**路由器**（常用于**网络核心**）和**链路层交换机**（常用于**接入网中**）。 从发送端到接收端系统，某分组所经历的一系列通信链路和分组交换机称为通过该网络的**路径**。 端系统通过**因特网服务提供商（ISP）**接入因特网。每个ISP都是由**多个分组交换机**和**多段通信链路**组成的网络，各ISP为端系统提供不同类型的网络接入，也为内容提供者提供因特网接入服务，将Web站点接入因特网。ISP之间互联，每个ISP均独立管理且运行IP协议。 端系统、分组交换机和其他因特网部件都运行一系列**协议**，这些协议控制因特网中信息的发送和接收（终端与终端，终端与交换设备，交换设备与交换设备），TCP和IP是因特网中最重要的两个协议，因特网中的主要协议统称为TCP/IP。 **因特网标准**由**因特网工程任务组（IETF）**研发，IETF的标准文档称为**请求评论（RFC）。** 涉及多个相互交换数据的端系统的应用程序称作分布式应用程序，如QQ、多人游戏。因特网应用程序运行在端系统上而非分组交换机中。 **因特网定义一：**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，该定义对于通信功能的实现有指导作用：ISP内部和之间互联。 **因特网的几个特点：**因特网是网络的网络，不存在严格的层次结构，没有统一的管理机构。

服务描述：**因特网定义二：**为分布式应用提供通信服务的基础设施，该定义对于服务接口的定义有指导作用：有序可靠的数据交付服务&不可靠的数据交付服务。 与因特网相连的端系统提供了**套接字接口（socket interface）**，它规定了运行在一个端系统上的软件请求因特网基础设施向运行在另一个端系统上的特定目的地软件交付数据的方式，是一套发送程序必须遵循的规则集合。

什么是协议：一个**协议**定义了在两个或多个通信实体之间交换的报文格式和次序，以及报文发送和/或接收一条报文或其他事件所采取的动作。

**网络边缘：**

端系统也称为**主机**，因为它们容纳（即运行）应用程序，有时进一步被划分成两类：**客户**和**服务器**。

接入网：**接入网**是将端系统物理连接到其**边缘路由器**的网络。**边缘路由器**是端系统到任何其他远程端系统的**路径**上的**第一台路由器**。 宽带住宅接入有两种流行的类型：**数字用户线（DSL）**和**电缆**。 DSL使用已有的数字电话线（双绞铜线，每户一条线），同时（从高频到低频）承载高速下行信道（<24Mbps）、中速上行信道（<2.5Mbps）和普通的双向电话信道（频分复用），此时电话公司为用户的ISP；家庭的DSL调制解调器得到数字数据后将其转换为高频音；5~10英里内才可如此接入。 **电缆因特网接入：**利用有线电视公司现有的有线电视基础设施，同时应用了光缆和同轴电缆，称为混合光纤同轴（HFC）系统，同时还有电缆调制解调器（cable modem）和电缆调制解调器端接系统（CMTS）；500~5000户家庭公用一个电缆，一个重要特征是共享广播媒体；下行30Mbps，上行2Mbps（DOCSIS 2.0定义了下行42.8Mbps和30.7Mbps上行）。家庭接入还包括光纤到户（FTTH）、卫星链路和使用传统电话线的拨号接入。 **以太网（Ethernet）：**公司和大学校园以及有较多终端的家庭使用局域网（LAN）；以太网是最为流行的接入技术，使用双绞铜线；以太网交换机及链路构成接入网；10Mbps，100Mbps，1Gbps，10Gbps；IEEE802.11无线LAN接入（WiFi），终端与基站几十米内，11Mbps或54Mbps。 **广域无线接入（3G、4G、长期演进/LTE）：**移动通信公司提供，使用现有的蜂窝电话网络；数万米；3G 2Mbps，4G 下行100Mbps上行20Mbps。

物理媒体：每个传输器-接收器对都通过一种**物理媒体**传播电磁波或光脉冲来发送比特。物理媒体可以有各种形状和形式，并且对路径上的每个传输器-接收器对而言不必相同。 物理媒体划分为**导引型媒体（沿固体媒体传播）**和**非导引型媒体（在空气或外层空间）**。 **双绞线：**两根隔离的铜线，常用在LAN、电话线网线，10Mbps到10Gbps，速率取决于线的粗细与传输距离。 **同轴电缆：**两根同心的绝缘铜导体，有线电视电缆，能作导引型共享媒体。 **光纤：**导引光脉冲的玻璃纤维，几十~几百Gbps，抗电磁干扰，长距离传输低衰减，难窃听，低误码率。 **电磁波：**无物理线路，并具有穿透墙壁、提供与移动用户的连接以及长距离承载信号的能力。

**网络核心：**

**网络核心**：由互联因特网端系统的分组交换机和链路构成的网状网络。 **网络核心的任务**：将数据包从发送侧的边缘路由器传送到接收侧的边缘路由器。 **基本问题**：数据包如何在网络核心中高效地传递（分组传输延迟小，网络吞吐量高）？ **两大关键功能**（PPT）：**路由**（决定从源到目的地的网络路径，由路由算法完成）和**转发**（将数据包从路由器的输入端口移到输出端口，通过查询转发表）。 通过网络链路和交换机移动数据有两种基本办法：**电路交换**和**分组交换**。

**分组交换**：源将长报文划分为较小的数据块，称为**分组**。 分组以某链路的最大传输速率通过该通信链路传输，用时L（bit的分组） / R（bit/s，链路传输速率），这就是**传输时延**。 **存储转发传输机制：**在交换机能够开始向输出链路传输该分组的第一个bit之前，必须接收到整个分组（为了校验和确认目的地）。 每个分组交换机有多条链路与之相连。对于***每条***相连的链路，该分组交换机均有一个**输出缓存（输出队列）**，用于缓存路由器准备发往该链路的分组。 分组承受输出缓存引起的**排队时延（**随网络拥塞程度变化**）**。 因为缓存空间的大小是有限的，一个到达的分组可能发现输出缓存已被其他等待传输的分组充满了，在此情况下将出现**分组丢失（丢包）**：*到达的分组或已经排队的分组之一将被丢弃。*大量分组集中到达时，排队延迟和丢包较严重。 在因特网中，每个端系统均具有IP地址，发送端在发出分组的首部包含了目的地的IP地址。每台路由器均具有一个转发表，用于将目的地址（或其一部分）映射成为输出链路。当一个分组到达网络中的路由器时，路由器检查该分组的目的地址的一部分，并向一台相邻路由器转发该分组。 因特网有特殊的**路由选择协议**，用于自动设置转发表。

**电路交换**：电路交换网络中在通信期间***预留了***端系统间的通信资源（缓存，链路传输速率）（分组交换网络中这些资源按需使用，可能要排队），例子为传统的电话网。 当两台主机要通信时，该网络在两主机之间创建一条专用的**端到端连接**，该链接被称为一条**电路**，网络创建时预留了恒定的速率，发送方能**确保**以恒定的速率向接收方传送数据。区分：链路是物理媒体，也称信道，可以通过某种方式划分为若干条独立的子信道；电路是物理媒体中的一条子信道。 链路中的电路是通过**频分复用（FDM）**或**时分复用（TDM）**实现的：FDM的链路为每个连接专用一个频段（频率范围），其宽度称为**带宽（bandwidth）**；TDM中时间被划分为固定区间的**帧**，每帧被划分为固定数量的**时隙**，每个帧的每个时隙为不同连接专门使用。TDM电路的传输速率=帧速率（时长的倒数）\*1个时隙中的bit数=总带宽/时隙数。 **电路交换的缺点：静默期**转用电路空闲而效率较低；创建端到端电路和预留端到端带宽需要复杂的指令软件以协调沿端到端路径交换机的操作。 **分组交换的缺点：**分组交换不适合实时服务，因为其端到端时延是可变和不可预测的（可能丢包），并且不能保证服务质量（带宽）。 **分组交换的优点：**提供更好的带宽共享；比电路交换更简单更有效，实现成本更低，适合突发数据，可容纳更多用户。

网络的网络：端系统经接入ISP与因特网相连，而接入ISP自身必须互联。**接入ISP**；**地区ISP**；**第一层ISP**；**存在点（PoP）：**低层ISP接入高层ISP的地方；**多宿：**一个低层ISP可以接入多个高层ISP（允许ISP之一出现故障时继续发送、接收分组）；**对等：**相同层次的ISP连到一起以试图绕过上层；**因特网交换点（IXP）：**第三方公司创建，多个ISP可在这里对等；**内容提供商网络**（谷歌等大公司自己的网络，包含多个互联的数据中心，每个数据中心包括多台服务器）。

**分组交换中的时延、丢包和吞吐量：**

分组交换网中的时延概述：分组沿某路径传播时，在沿途的*每个*结点均经受了几种不同的时延：**结点处理时延**，**排队时延**，**传输时延**，**传播时延**，累加起来是**结点总时延**。 **处理时延：**检查分组首部、决定分组导向何处，检查bit级别的差错（μs或更低）。 **排队时延：**分组在链路上等待传输的时间，取决于先前到达的排队的分组数量和拥塞程度，是流量强度和性质的函数。（差异很大，μs到ms） **传输时延：**将所有比特推上链路所需要的时间。L bit的分组在R bps的链路上的传输时延是L/R。（μs到ms） **传播时延：**从该链路的起点到终点*传播*所需要的时间，速率取决于*物理媒体*，范围是2/3~1倍光速，传播时延为*距离d/传播速率v*。（ms级）。总时延d\_nodal=d\_proc（处理）+d\_queue（排队）+d\_trans（传输）+d\_prop（传播），这些成分所起的作用可能差异很大（传播时延：大学内部可忽略，但它是卫星链路延迟的主要来源）。

排队时延和丢包：不同于其他3种时延，不同分组的排队时延可能不同（先到的短，后到的长）。排队时延一般使用统计量（均值、方差、分布）表征。 a表示分组到达输出队列的平均速率（pkt/s），R传输速率bps，L分组大小bit，L\*a/R称为**流量强度**。 流量强度>1，排队时延趋向无穷大，*设计系统时流量强度不能大于1*；流量强度<=1，到达流量的性质影响排队时延，分组周期性到达平均时延低，突发一次到达多个平均时延高。 **平均排队时延和流量强度的定性关系：**随着流量强度接近于1，平均排队时延迅速增加，该强度少量的增加将导致时延大得多的增加。 一条链路前的队列只有有限的容量，尽管排队容量极大的依赖路由器的设计和成本。不过，因为输出队列的容量是有限的，随着流量强度接近1时，排队时延并不趋向无穷大。相反，路由器在队列已满后，将**丢弃**新到达的分组，该分组将**丢失**。分组丢失的份额随着流量强度的增加而增加。 度量一个结点的性能，不仅需考虑节点时延，常常还要考虑分组丢失的概率。

端到端时延：**端到端时延：**分组传输路径上所有结点的节点延迟之和。对端到端延迟敏感的应用：实时交互应用高度敏感（如网络电话视频会议），在线交互应用中度敏感（如网页浏览）。

计算机网络中的吞吐量：**瞬时吞吐量：**在任何瞬间，目的主机接收到数据的速率（bps）。**平均吞吐量：**文件总bit数/总时间。 链状链路的吞吐量：min { R1 , R2 , … , Rn }，即**瓶颈链路**的传输速率。公共链路吞吐量：min { Rc , Rs , R\_pub/user\_num }。吞吐量取决于数据流过链路的实际传输速率，而非固有的传输速率；还取决于干扰流量（比如说来自于其他用户的）。若许多干扰数据流通过具有高传输速率的链路传输，则该高速链路仍可能成为传输的瓶颈链路。

**协议层次及其服务模型：**

分层的体系结构：每个层次与其下面的层次结合在一起，实现了某些功能/服务。 **系统分层：**将系统按功能划分为一系列水平的层次，每一层实现一个功能（服务）。 **层次间关系：**每一层的功能实现都要依赖其下各层提供的服务。某层向它的上一层提供的**服务**就是该层所谓的**服务模型**。 **每个层次通过如下方式提供服务：**1. 在**该层中**执行了某些动作 2. 使用**直接下层**的服务。 **分层的好处：**分层提供了讨论系统组件的结构化方式。层次结构易于确定系统的各个部分及其相互关系，协议分层具有概念化和结构化的优点；模块化简化了系统的维护和升级（改变某层服务的实现方式对于其他层次没有影响），使更新系统组件更为容易。 网络设计者以**分层**的方式组织协议以及实现这些协议的网络硬件和软件，每个协议属于这些层次之一。 一个协议层能够用软件、硬件或二者的结合来实现（应用层和传输层是端系统的软件实现，物理层和数据链路层是在与链路相联系的网络接口卡中实现，网络层是软硬件实现的混合）。 一个位于某层的协议**分布**在构成该网络的端系统、分组交换机和其他组件中，即该协议的不同部分常位于这些网络组件的各部分中。 **分层的缺点：**较高层可能冗余（重复实现）较低层的功能；某层的功能可能需要仅在其他某层才出现的信息，这违反了层次分离的目标。 各层的所有协议被称为**协议栈：物理层、链路层、网络层、传输层和应用层。 应用层：网络应用程序**及它们的**应用层协议**存留的地方。**应用层协议**分布在多个端系统上，不同端系统的应用程序之间使用协议交换信息分组，位于**应用层**的信息分组称为**报文（message）**。 **传输层：**利用传输层协议（TCP/UDP）可以在应用程序与网络的接口间（进程间/应用程序端点间）传输**报文段（segment）**。 **网络层：**负责将称为**数据报（datagram）**的网络层分组从一台主机移动到另一台主机，包括IP协议和路由选择协议。 **链路层：**在相邻节点（主机/路由器）间传输**帧（frame）**。 **物理层：**将帧中的一个一个**比特**从一个节点移动到下一个节点（物理媒体上）。 **ISO/OSI：**在应用层和传输层之间插入表示层和会话层。因特网中，这两层的功能留给应用程序开发者处理。

封装：与端系统类似，路由器和链路层交换机以多层次的方式组织他们的网络硬件和软件。而路由器和链路层交换机**并不**实现协议栈中的所有层次。**链路层交换机**实现了**物理层**和**链路层**，**路由器**实现了**物理层、链路层**和**网络层**。主机实现了所有五个层次。 这体现了**封装**的概念。 一个分组具有两种类型的字段：首部字段和**有效载荷字段**。有效载荷通常来自上一层的分组。在分组跨越不同层时，通常伴随着首部字段的添加和删除。

***第2章：应用层***

**网络应用**是计算机网络**存在的理由。**

**应用层协议原理：**

网络应用程序时需要在多台端系统上运行，并能通过网络相互通信，但不需要也不能写在网络核心设备（如路由器或链路层交换机）上运行的软件，因为网络核心设备并不在应用层上起作用或运行用户应用程序。将应用软件限制在端系统的方法，促进了大量的网络应用程序的迅速研发和部署。

网络应用程序体系结构：应用程序体系结构不同于网络的体系结构。在应用程序研发者的角度看，后者是固定的，并为应用程序提供了特定的服务集合。网络**应用程序体系结构**由应用研发者设计，规定了如何在各种端系统上组织该应用程序。 现代网络应用程序的两种主流体系结构：**客户-服务器体系结构**和**对等（P2P）体系结构**。 **客户-服务器体系结构：**有一台总是打开（在线）的主机（称为**服务器**）服务来自许多其它主机（称为**客户**）的请求，用户终端上运行客户端进程，需要时与服务器进程通信，请求服务。 **特点：**服务器主机具有*固定的*、周知的地址，称为IP地址；客户之间不直接通信；客户机可以使用动态地址，通常不会总是在线。 客户-服务器架构的**资源集中**，资源（服务）只在某些固定的终端上提供。**优点**：资源发现简单。**缺点**：集中式计算带来的问题如服务器扩容压力、网络流量不均衡、响应延迟长，数据中心或可缓解压力。 **P2P体系结构**对很少（或几乎不）依赖位于数据中心的专用服务器。应用程序在间断连接的主机对之间使用直接通信，这些主机对被称为**对等方**。 **P2P特性：**自扩展性、节省成本（通常不需要庞大的服务器基础设施和服务器带宽）。 **优点：**易于扩容、均衡网络流量；**缺点：**资源高度非集中，安全性、性能和可靠性难保障。

进程通信：进行通信的实际上是**进程（端系统上运行的程序）**而非应用程序。 **进程通信的方法：**同一主机内使用OS提供的进程间通信机制。在两个不同端系统上的进程（如分布式应用），通过跨越计算机网络交换**报文**来相互通信。 对每对通信进程，将发起请求的进程标识为**客户**，另一个在会话开始时等待联系（接受请求）的进程标识为**服务器**（P2P中的每个“peer”*可以*同时是客户和服务器）。 进程通过被称为**套接字（socket）**的软件接口向网络发送报文和从网络接收报文。**套接字**是**同一台**主机内部**应用层**与**传输层**的接口,也称为应用程序和网络之间的**应用程序编程接口（API）**。 网络应用程序开发者可以在应用层端控制套接字，但是除了选择传输层协议和设定部分传输层参数外无法控制传输层。 为了标识接收分组的进程（在另一台主机上运行），需要定义两种信息：主机的地址、在目的主机中的指定接收进程的标识符。在因特网中，主机用其**IP地址**标识，接收进程由目的地**端口号**标识。一些特定端口号被分配给了特定的应用（HTTP 80，SMTP 25）。

可供应用程序使用的传输服务：不同传输层协议提供不同的服务，应用程序对服务的要求包括可靠数据传输、吞吐量、定时、安全性。 **可靠数据传输：**确保数据从一端正确、完整地交付给应用程序的另一端。**容忍丢失的应用**（如多媒体应用）文件不要求完全可靠的数据传输。 **吞吐量：**传输层协议能以某最低速率保证可用吞吐量。**带宽敏感的应用**（多媒体应用）有吞吐量要求，**弹性应用**（电子邮件、文件传输与web传送）则不然。 **定时：**如发送方注入套接字中的每个bit到达接收方套接字耗时不长于某时间。交互式实时应用程序（因特网电话、虚拟环境、电话会议和多方环境）有该限制，而非实时应用没有严格约束。 **安全性：**比如传输层协议能够加密发送进程传输的所有数据，然后在交付接收进程之前解密；也包括数据完整性和端点鉴别。

因特网提供的传输服务：因特网（更一般的是TCP/IP网络）*提供***UDP**和**TCP**两个传输层协议（它们均不保证安全）。 TCP服务模型包括**面向连接服务**和**可靠数据传输服务**：客户和服务器握手后在二者套接字间创建**TCP连接**，它是全双工的（双方可以同时进行报文收发）。结束发送时，必须拆除连接；TCP能无差错、按顺序交付所有发送的数据。TCP还有**拥塞控制**机制（不一定对通信进程带来直接好处，但能对因特网带来整体好处：公平共享网络带宽）和**流量控制**（发送端数据发送速率不会超过接收端能力）。**TCP不提供：**及时性、最低带宽保证，安全性。 TCP的加强版本**安全套接字层（SSL）**提供了TCP所能提供的一切以及进程到进程的安全性服务（加密，数据完整性和端点鉴别）。***SSL强化是在应用层实现的TCP补丁。*** UDP是不提供不必要服务的轻量级运输协议，仅提供最小服务：无连接（进程通信前无握手），不可靠数据传输（可能无法到达或乱序到达），没有拥塞控制机制（发送进程可以以任何速率向网络层注入数据），没有安全性。 **UDP还是TCP都不提供**吞吐量和定时保证，但应用程序可以容忍缺乏这种保证的传输层。 能够容忍某些丢失但要求达到一定最小速率才能有效工作的应用（如因特网电话）使用UDP以设法避开TCP的拥塞控制机制和分组开销。

应用层协议：**应用层协议**定义了运行在不同端系统上的进程如何相互传递报文，特别是交换的报文类型（请求/响应）、各种报文类型的语法（包含哪些字段、字段的顺序）、字段的语义（字段信息的含义）、进程何时以及如何发送报文、进程响应报文的规则。 开放协议由RFC文档定义，而一些别的应用层协议（如Skype和微信）是私有的。 网络应用和应用层协议不同：应用层协议（如HTTP）只是网络应用（Web应用还包含浏览器、服务器和HTML）的一部分。

**Web和HTTP：**

HTTP概况：Web是因特网应用，其**应用层协议**是**超文本传输协议（HTTP）**。 HTTP由客户程序和服务器程序实现，二者分布在不同的端系统通过HTTP报文会话。 **Web页面（也叫文档）**由对象组成，一个**对象**是单个文件（HTML/JPEG/\*.java），多数web页面含有1个**HTML基本文件**以及几个引用对象（包含HTML文本和5个JPEG图形的Web页面有6个对象：1个HTML基本文件+5个图形）。HTML基本文件通过对象的URL地址引用页面中的其他对象，每个URL地址由两部分组成：存放对象的服务器主机名（www.someSchool.edu）和对象的路径名（/someDepartment/picture.gif）。**Web浏览器**实现了HTTP的客户端，**Web服务器**实现了HTTP的服务器端（用于存储Web对象，对象均用URL寻址）。 **HTTP**定义了web客户向web服务器请求web页面和服务器向客户传送Web页面的方式，**使用TCP**。HTTP服务器**不保存**关于客户的任何信息，是一个**无状态协议**。**Web采用**客户-服务器架构，Web服务器总是打开的。

非持续连接和持续连接：HTTP既能使用非持续连接，也能使用持续连接（默认）**。非持续连接：**每个请求/响应对经单独的TCP连接发送；**持续连接：**所有的请求/响应对经相同的TCP连接发送。HTTP 1.0使用非持续连接，HTTP 1.1默认使用持续连接。 **非持续连接例**：客户在80端口发起到服务器的TCP连接（**RTT**）→客户经连接发送HTTP请求报文→服务器接受请求，检索、封装对象，发送HTTP响应报文（**RTT**）→服务器通知TCP断开连接，实际上客户完整接收HTTP响应报文才关闭连接（**d\_trans**）。对每个对象重复一次步骤。每个TCP连接在服务器发送1个对象后关闭。 **往返时间（RTT）**指一个*短（不考虑传输时延）*分组从客户到服务器再返回客户花费的时间（包括传播、排队和处理时延） 非持续连接传输种，每个对象需要**2个RTT+传输时延（因为文件有大小）**。 **非持续连接缺点：**为每个请求对象建立、维护一个全新的TCP连接，并维护TCP缓冲和变量，服务器负担重；每个对象交付时延2RTT；浏览器要打开多个TCP连接来获取一个网页。 在采用**持续连接**的情况下，服务器再发送响应后保持该TCP连接打开，后续请求和响应报文能够通过相同的连接进行传送，一个完整的页面也可用单个TCP连接传送，甚至同一服务器发给同一客户的多个页面也可使用同一连接。若一条连接在一定时间内未使用，则HTTP服务器将关闭该链接。HTTP默认使用**带流水线**的**持续连接**。

HTTP报文格式：**ASCII**文本组成；每行用CRLF（回车+换行符），最后一行附加一个额外CRLF作为结束；第一行**请求行**：方法字段（**GET**（获取）**，POST，HEAD，PUT，DELETE，**后2个1.1专属），URL字段（请求对象的标识），HTTP版本（自解释）；后面是**首部行**：Host（对象所在的主机），Connection（close：不持续连接），User-agent（用户代理浏览器版本），Accept-language（指定语言）。 **额外的crlf后是实体体（entity body）**：GET时为空，POST时使用该部分发送用户向搜索引擎提供的关键字来提交表单（但提交表单可以放在GET的URL里）。 **HEAD**方法会让服务器相应一个报文但不返回内容（调试用）；**PUT**方法允许用户上传对象；**DELETE**允许用户删除对象。 **HTTP响应报文**包含初始**状态行**（协议版本，状态码【200 OK；301 Moved Permanently（新URL定义在响应报文的Location首部行）；400 Bad Request请求消息不被服务器理解；404 Not Found；505 HTTP Version Not Supported。】，状态信息），**首部行**（Connection同请求；Date报文产生时间；Server服务器类型；Last-Modified对象创建或最后修改时间；Content-Length、Contest-Type对象字节数和类型），**实体体**（包含请求对象本身）。

用户与服务器的交互Cookie：HTTP无状态简化了服务器的设计，允许开发高性能的Web服务器。**cookie**允许站点识别用户，将内容与用户身份联系或限制用户的访问。用户信息保存在服务端的后端数据库，返回客户的是一个ID（若第一次）或与ID相关联的信息；响应报文发回客户端时，保存在cookie文件中，ID会随请求报文发送给服务器。 **cookie技术四个组件：**HTTP响应报文和HTTP请求报文中的Cookie首部行；用户端系统浏览器中保留管理的Cookie文件；Web站点的后端数据库。例：A第一次访问某网站。HTTP请求到达网站时，网站为其创建一个ID，并在后端数据库中为该ID建立一个表项，响应报文中包含set-cookie字段，内容为该ID。A再次访问网站时，服务器根据请求报文的cookie字段执行特定cookie动作。**cookie**可标识用户，在无状态的HTTP上建立用户会话层，但收集用户信息会带来**隐私问题**。

Web缓存：**Web缓存器（代理服务器）**能在不访问源服务器的情况下满足客户端请求。Web缓存器有自己的存储空间，并在其中保存最近请求过的对象的副本。**可以让浏览器首先将HTTP请求发送给web缓存器**，若对象在其缓存中则其返回对象；反之Web缓存器与原始服务器TCP连接获取对象，本地缓存之并返回给客户。Web缓存器既可是服务器也可是客户。Web缓存器通常由ISP提供，减少客户请求响应时间、机构的接入链路到因特网的通信量（降低了费用）；降低了因特网整体的Web流量，改善了所有应用的性能。

条件GET方法：存放在Web缓存器里的对象副本有可能是陈旧的。解决方法为**条件GET**：若请求报文使用GET方法且包含If-Modified-Since:首部行。Web缓存器在缓存对象时也存储了最后修改日期。若服务器在收到条件GET请求报文后发现对象无更新，则响应报文不含该对象（避免浪费带宽），状态行为**304 Not Modified**（有更新为200）。

文件传输协议：FTP FTP和HTTP均为文件传输协议，FTP服务器维护当前目录、用户认证状态等状态，是有状态协议。FTP用2个并行TCP连接来传输文件：**控制连接（21端口）**和**数据连接（20端口）**，使用7bitASCII在控制连接上传送**命令/响应交互**（不像HTTP那样用报文交互），每一次数据连接只传送一个文件，一个文件传输结束则关闭连接。将控制连接与数据连接分开：不会混淆数据与命令/响应，简化协议设计和实现；在传输文件的过程中可以继续执行其它的操作，便于控制传输过程（如客户可以随时终止传输）。用关闭数据连接的方式结束文件传输允许动态创建文件，不需预先告知文件大小。**常见命令：**USER username；PASS password；LIST（返回响应的文件列表，经数据连接传输）；RETR filename（传给用户文件）；STOR filename（用户存储文件）。**回答：**3位数字，后跟可选信息：331 Username OK，Password required、125 Data connection already open; transfer starting、425 Can’t open data connection、452 Error writing file。

**因特网中的电子邮件：**

3个主要组成部分：**用户代理**，**邮件服务器**，**简单邮件传输协议（SMTP）**。用户代理允许用户阅读、回复、转发、保存和撰写报文。用户代理向邮件服务器发送邮件，邮件存储在**邮件服务器**的**外出报文队列**中。邮件服务器是电子邮件体系结构的核心，每个接收方在某个邮件服务器上有一个**邮箱**。**典型的邮件发送过程：**从发送方的用户代理开始，传送到发送方的邮件服务器，再传输到接收方的邮件服务器，并接着被分发到接收方的邮箱中。 发送者的邮箱必须能处理接收者邮件服务器的故障，如果不能交付邮件，则邮件放在发送者的邮件服务器的**报文队列**中，在以后尝试再次发送。

邮件传输协议SMTP：**SMTP**是因特网电子邮件中主要的**应用层**协议，**使用TCP**，**端口25**，SMTP有**两个部分**：运行在发送方邮件服务器的客户端和运行在接收方邮件服务器的服务器端（每台服务器既是客户也是服务器）。SMTP**规定**所有邮件报文的体部分（不只是首部）只能**采用简单的7bitASCII表示**，**不使用中间邮件服务器**传送邮件，邮件不会在中间某个邮件服务器存留。传输邮件步骤：握手，传输邮件消息数据、关闭。**采用命令/响应交互方式，**使用**持续连接。**

与HTTP对比：都用于从一台主机向另一台主机传送文件，都用持续连接；HTTP是一个**拉协议（请求者主动）**，而SMTP是一个**推协议（发送者主动）**。SMTP要求**所有报文都是ASCII**，非字符数据要编码成字符，HTTP无此限制。HTTP把**每个**对象封装到它***自己***的HTTP响应报文中，而SMTP则把所有报文对象放在***一个报文***中。

邮件报文格式：**首部行**和**报文体**用空行（CRLF）分割；首部行包括：From:，To:，（可以包含Subject:和其他的）。这些首部行和SMTP命令是有区别的，它们是邮件报文的一部分。主体只用ACSII编码。

邮件访问协议：**邮件访问方式：**用户在终端上安装用户代理，获取和阅读邮件。不能将用户信箱放在本地终端，因为用户终端不可能一直连在因特网上。**用户代理可以用SMTP将邮件推向邮件服务器**，但SMTP**不能**从邮件服务器把邮件拉到用户代理。必须引入邮件访问协议将邮件服务器上的报文传到本地PC：**第三版的邮局协议（POP3）**，**因特网邮件访问协议（IMAP）**，以及HTTP。 **POP3：TCP**，**110**端口；有三个**工作阶段**：特许（代理明文传输用户名和密码）、事务处理（代理取回报文，对报文做删除标记/取消报文删除标记，获取邮件统计信息）、更新（发出quit命令之后服务器删除被标记的报文）。**命令：**特许阶段user，pass；事务处理list，retr，dele，quit。**回答：**+OK（有时后面跟有数据），-ERR。**下载并删除**（其他终端不能再看到）或**下载并保留**（其他终端可以再获取）。POP3服务器保留了一些状态信息（如记录哪些用户报文被标记为删除），但POP3服务器并不在POP3会话中携带状态信息，这简化了协议的实现。**IMAP：**所有邮件保存在服务器上，允许用户将邮件组织在文件夹中和在文件夹间移动邮件，维护了会话的用户状态信息，允许用户代理获得报文的一部分。**HTTP：**用户代理也可以是Web浏览器，故也可使用**HTTP在用户代理和邮件服务器间上传和下载邮件**。

**DNS：因特网的目录服务**

主机的2种标识方法：**主机名**（www.yahoo.com）和**IP地址**（32bit）。DNS提供的服务：**域名系统（DNS）**包括由分层的**DNS服务器**实现的分布式数据库和一个使主机能够查询分布式数据库的**应用层协议**。 DNS运行在**UDP或TCP**上，端口**53。DNS在应用层上的原因：**使用客户-服务器模式运行在通信的端系统间；在通信的端系统间通过下层的端到端传输协议传送DNS报文。 **DNS与其他应用的不同之处：**不直接和用户打交道，而是为因特网上的用户应用程序以及其它软件提供核心功能。DNS给使用它的应用带来了额外的延时。 **DNS的服务：主机名到IP地址的转换；主机别名到规范主机名的映射（容易记忆）；邮件服务器别名**（允许使用域名作为邮件服务器的别名）**；负载分配**，允许一个规范主机名对应一组IP地址，将服务请求分配到功能相同的**一组**服务器上。

DNS工作机理：应用程序（如浏览器）调用本地DNS客户端（解析器），要查询的主机名作为参数传递；解析器向DNS服务器发送查询报文，包含主机名；解析器收到含IP地址的响应报文，将IP地址返回给调用者（如浏览器）。**不使用集中式DNS的原因：**单点故障；单个服务器处理全部查询（通信容量过大）；维护庞大的数据库（维护困难）；距集中式数据库较远的地方时延高；没有可扩展能力。**三种类型的DNS服务器以层次方式组织：根DNS服务器**（最高层，13个），**顶级域（TLD）DNS服务器**（次高层，.com/org/uk/fr），**权威DNS服务器**（低层，每个组织机构拥有，提供主机名到IP地址的映射，可以由组织或ISP维护）。**本地DNS服务器：**不在层次结构中，常邻近用户，起代理作用或将请求转发到DNS服务器的层次结构中。**查询示例：**主机向它的本地DNS服务器发送一个DNS查询报文，本地DNS服务器将该报文转发到根DNS服务器，得到com域的TLD DNS服务器地址；继续查询com域的TLD DNS服务器，得到amazon.com域的权威DNS服务器地址；最后查询amazon.com域的权威DNS服务器，得到 www.amazon.com的IP地址，本地DNS服务器将其返回给主机。TLD服务器一般只知道中间的某个DNS服务器，依次查询中间DNS服务器才能得到权威DNS服务器的IP。 **递归查询：**高负载。**迭代查询：**收到查询报文的服务器将下一个需要查询的服务器地址返回给查询者。实际查询时，从请求主机到本地DNS服务器的查询是递归的，其余的查询是迭代的。**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**，首先使用缓存中的信息响应查询请求。DNS缓存中的映射在一定时间后被**丢弃。**因为DNS缓存，实际的DNS查询很少访问根服务器。

DNS记录和报文：所有DNS服务器存储了提供主机名到IP地址的映射：**资源记录（RR）**；每个DNS报文包含了一条或多条资源记录。RR是**四元组：（Name，Value，Type，TTL）**，**TTL**是该记录的生存时间，它决定了资源记录应当从缓存中删除的时间；若***Type=A***，Name：**主机名**，Value：**IP地址**；若***Type=NS***，Name：**域** （e.g. foo.com），Value：**该域的权威DNS服务器主机名**；若***Type=CNAME***，Value：规范主机名，Name：对应的主机别名；***Type=MX***，Value：别名为Name的**邮件服务器**的规范主机名。**权威DNS服务器**包含其所用于的主机名的**A记录**；如果**不是权威DNS**，则包含一条**NS记录**和**记录该NS记录Value字段中DNS服务器的IP地址的A记录**。 DNS报文只有两种：**查询和回答**，并且格式相同：前12字节**首部区域**，包含16bit标识符来标识该查询，16bit标志来标记报文是查询（0）还是回答（1）、回答者是权威（1）还是非权威（0）、客户希望递归查询（1）还是不希望、递归可用（1）或不可用（0），4个16bit的有关数量的字段（依次是问题数，回答RR数，权威RR数，附加RR数）；**问题区域:**名字字段（包含正在被查询的主机名）和类型字段（有关该名字的正被询问的问题类型）；**回答区域**包含了对最初请求的名字的RR（可有多条，下同）；**权威区域**包含了其他权威服务器的记录；**附加区域**包含了其他有帮助的记录（如MX的回答包含某邮件服务器的规范主机名，则附加区域包含一个提供该邮件服务器的规范主机名IP地址的A记录）**向DNS数据库中插入资源记录：①向DNS注册登记机构注册域名**（如xxx.com），提供基本（和辅助）**权威DNS服务器的名字和IP地址**（如dns1.xxx.com、1.1.1.1） **②对每个权威域名服务器**，注册机构向TLD服务器**插入NS记录**和**指示该NS记录Value字段中权威DNS服务器IP的A记录[**如（xxx.com,dns1.xxx.com,NS）和（dns1.xx.com,1.1.1.1,A）]③**建立权威DNS服务器**，插入xxx.com的A记录（xxx.com,2.2.2.2,A）和xxx.com的MX记录（xxx.com,mail.xxx.com,MX）以及相应邮件服务器的A记录（mail.xxx.com,2.2.2.3,A）。

***第3章：传输层***

**概述和传输层服务：**

传输层协议运行在端系统（而非路由器）中，为运行在不同主机上的***进程间***提供了**逻辑通信**。在发送端，传输层将从发送进程接收到的报文转换成传输层分组，称为传输层**报文段（segment）**。

传输层与网络层的关系：网络层协议提供了***主机间***的逻辑通信。传输层协议能够提供的服务常**受制于**网络层协议的服务模型（网络层没有时延和带宽保证，则传输层也没有），但传输层**也能**提供某些服务（可靠的数据传输、加密）。

因特网传输层概述：网络层的网际协议（IP）提供**尽力而为的交付服务**，但并不做任何确保（不确保交付、按需交付和数据完整性），是**不可靠服务**。 **UDP和TCP最基本的责任是**，将**两个端系统间**IP的交付扩展为运行在端系统上的**两个进程间**的交付服务，这被称为**传输层的多路复用**和**多路分解**。UDP和TCP还可以在报文段首部加上差错检查字段提供完整性检查。进程到进程的数据交付和差错检查是两种最低限度的传输层服务，**也是UDP仅能提供的服务，UDP是不可靠的**。TCP提供**可靠数据传输**，确保正确地、按顺序地将数据从发送进程交付给接收进程；TCP还提供**拥塞控制**。通过调节TCP连接的发送端发送进网络的流量速率，防止任何一条TCP连接用过多流量来淹没通信主机间的链路和交换设备，让每个通过一条拥塞网络链路的连接平等地共享网络链路带宽。

**多路复用与多路分解：**

多路复用和多路分解是所有计算机网络都需要的。一个进程有一个或多个**套接字**，它相当于网络和进程间传递数据的门户。每个传输层报文段有几个字段，在接收端，传输层检查报文中唯一的套接字标识符，将报文定向到特定套接字。将传输层报文段中的数据交付到正确的套接字的工作称为**多路分解**。在源主机上，从不同套接字中收集数据块，并为每个数据块封装上首部信息从而生成报文段，再将报文段传递到网络层的工作称为**多路复用**。传输层多路复用要求：套接字有唯一标识符；每个报文段有指示该报文段应交付到的套接字的特殊字段。这些特殊字段是**源端口号字段**和**目的端口号字段**。端口号是16bit的数（0~65535），0~1023是受限制的**周知端口号，**保留给著名的应用层协议（如HTTP）。

无连接的多路复用与多路分解：通常，应用程序的客户端让传输层自动分配端口号，而服务器端则分配特定的端口号。**UDP的多路分解：**主机A中的传输层创建一个传输层报文段，包括源端口号、目的端口号。然后，传输层将得到的报文段传递到网络层，网络层将该报文段封装到一个IP数据报中，并尽力而为地将报文段交付给接收主机。报文段到达主机B时，传输层检查报文段中的目的端口号，并将其定向到相应的套接字。UDP套接字标识为<目的IP地址，目的端口号>二元组，所以若两个UDP报文段源IP地址和/或源端口号不同但目的IP地址和目的端口号相同，则这两个报文段将通过相同的目的套接字被定向到相同的目的进程，而源端口号用于回发报文段。

面向连接的多路复用与多路分解：TCP套接字标识为<源IP地址，源端口号，目的IP地址，目的端口号>四元组。当一个TCP报文段到达主机时，所有4个字段均被用来将报文段定向（分解）到相应的套接字，因此具有不同源IP地址或源端口号的TCP报文段到达时将被定向到不同的套接字，除非TCP报文段携带了初始创建链接的请求。

Web服务器与TCP：连接套接字与进程之间并非总是一一对应。高性能Web服务器通常只使用一个进程，但为每个新的连接创建一个具有新套接字的新线程。

**无连接运输：UDP**

UDP是**无连接的：**使用UDP发送报文段前，发送方和接收方的传输层实体之间没有握手。**有许多应用更适合UDP，主要因为：**应用对发送数据时间敏感（绕过TCP拥塞控制，并在应用层实现某些UDP不提供的服务）；无需连接建立（不引入连接建立的时延，DNS运行在UDP上的可能原因）；无连接状态（维护连接需要接收和发送缓存，拥塞控制参数、需要与确认号，运行在UDP上可以节省资源支持更多用户）；分组首部开销小。UDP没有拥塞控制，这使得在UDP上运行多媒体是有争议的。使用UDP的应用程序可以自身建立可靠性机制来达到可靠传输。当分组丢包率低时，出于安全原因，某些机构将阻塞UDP流量，故有时流式媒体传输也采用TCP。

UDP报文段结构：**UDP首部4字段8字节：源端口号，目的端口号，长度**（UDP报文段的字节数，包括首部和数据）**，校验和。后面是UDP数据字段。**

UDP校验和：UDP提供差错检测功能。**校验和计算：**第一次运算前校验和字段填入0，对UDP报文段及伪头部（16bit源IP，16bit目的IP、8bit前导0+8bit protocol字段，16bitUDP总长度字段）的所有16bit字求和（求和时遇到的任何溢出进位都回加到LSB）再进行反码运算，得到的结果放在UDP报文段中的校验和字段。**接收方**将全部16bit字（包括校验和）加在一起，**若结果不为全1则一定有错**，**全1不说明没错**。**UDP仅支持检出部分错误，不支持差错恢复**。UDP校验的实现：丢弃受损报文段或者将受损报文段交给应用程序并警告。（PPT）**很多链路层也提供差错校验，为何UDP还计算校验和？**不保证源到目的路径上的所

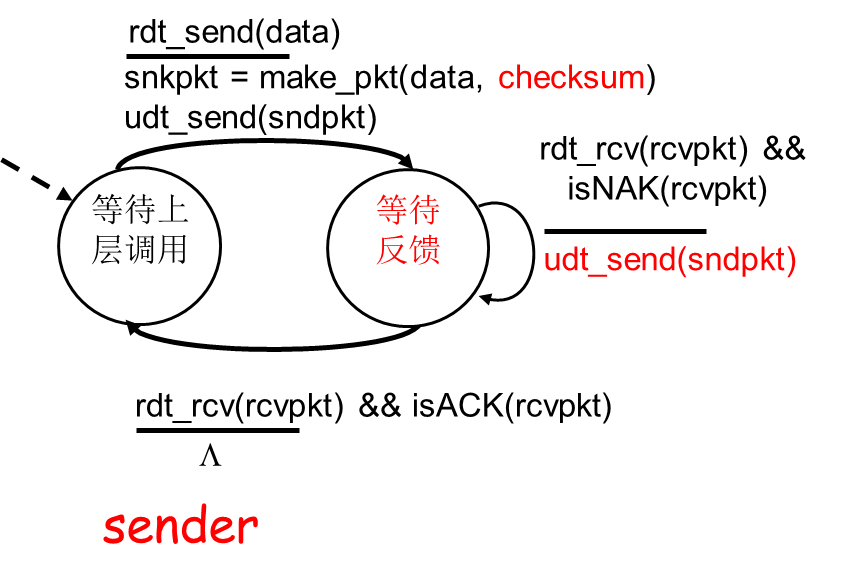
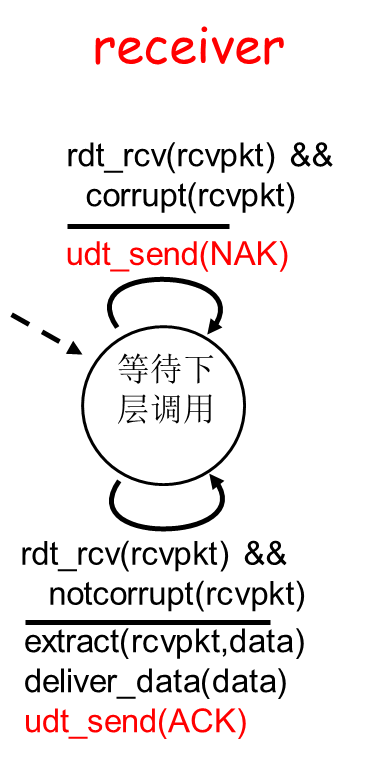
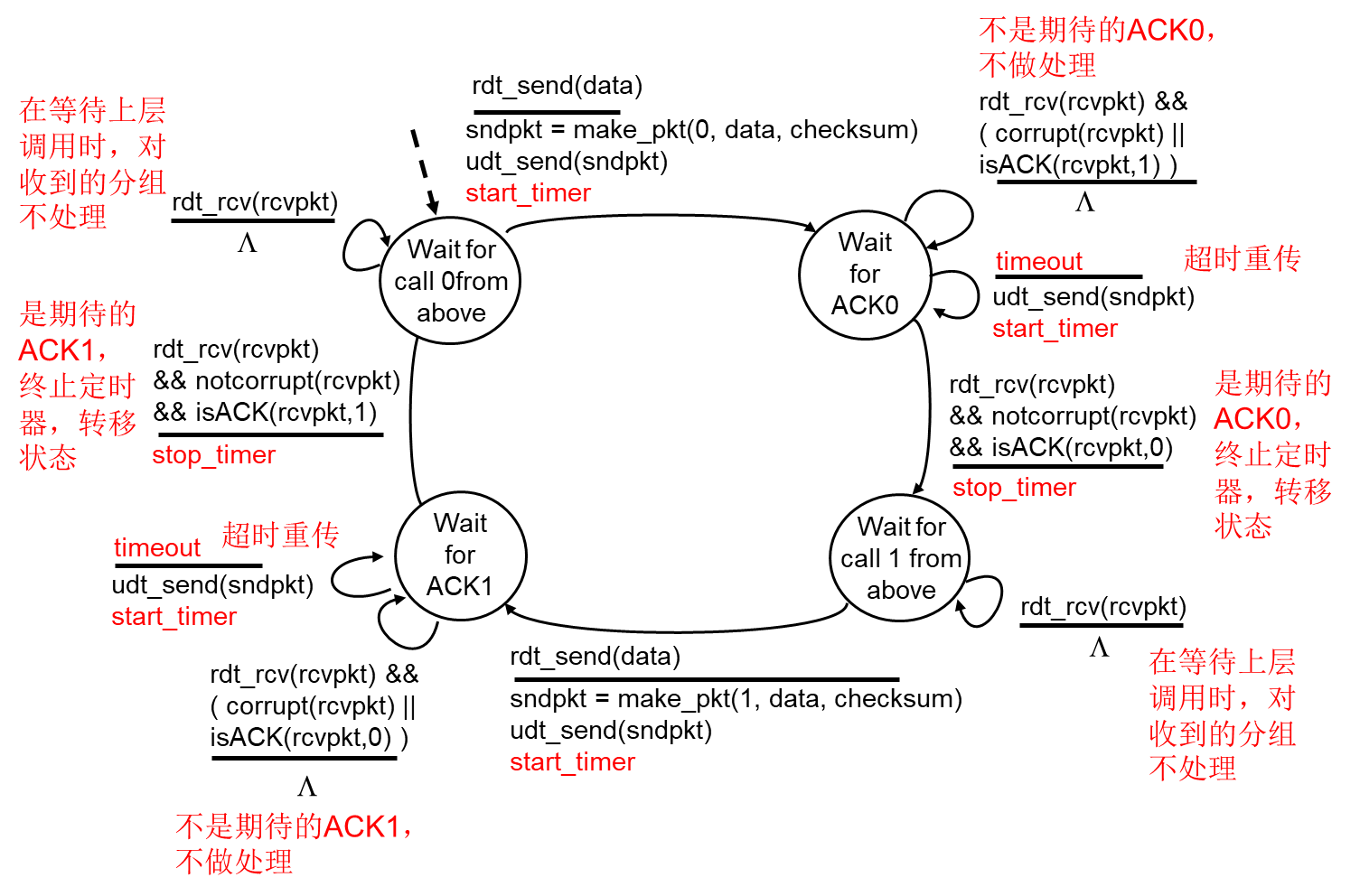
有链路都实现差错校验；链路层实现位置仅在收发端口，路由器内存中也可能发生bit翻转。这是“端到端原则”的例子。

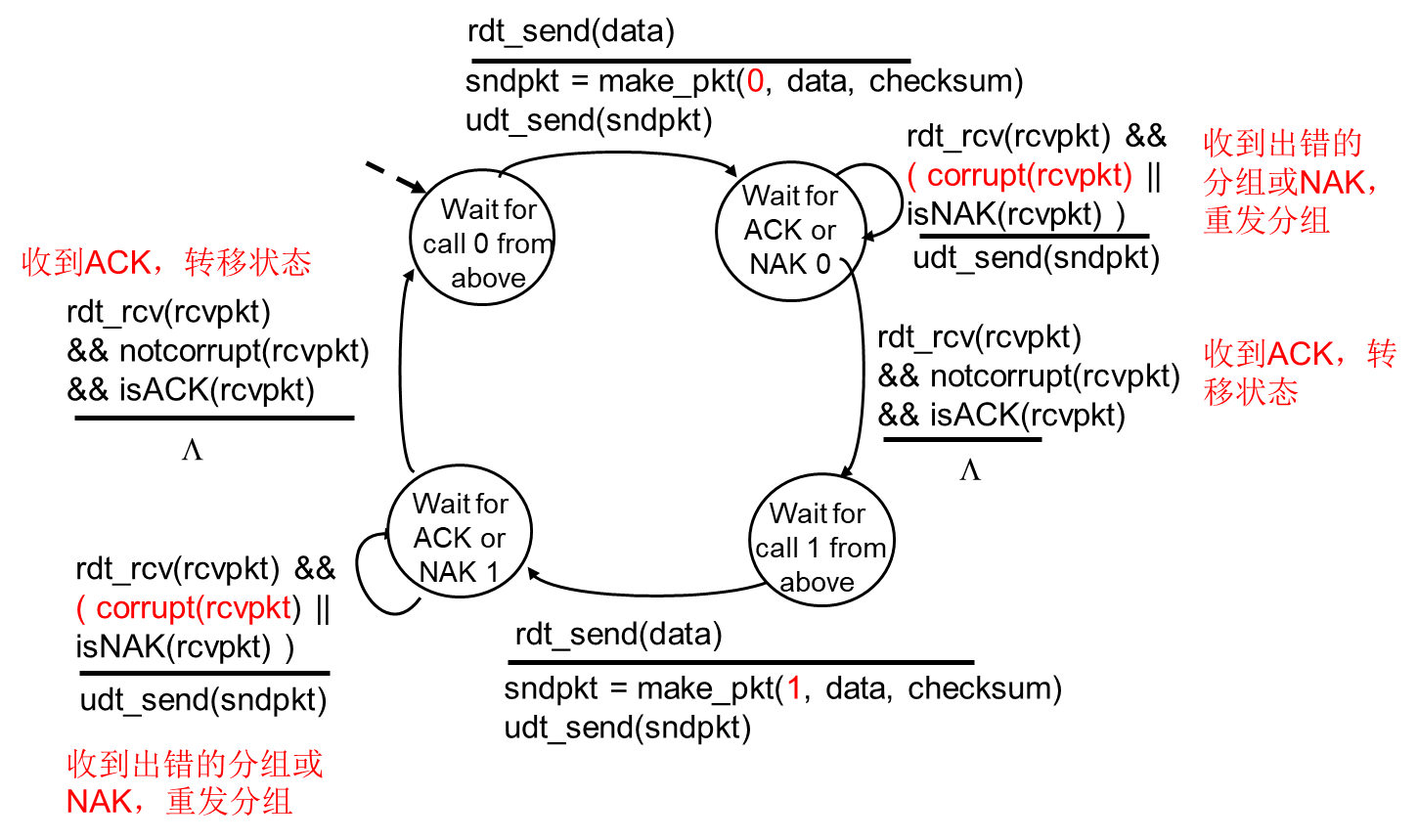
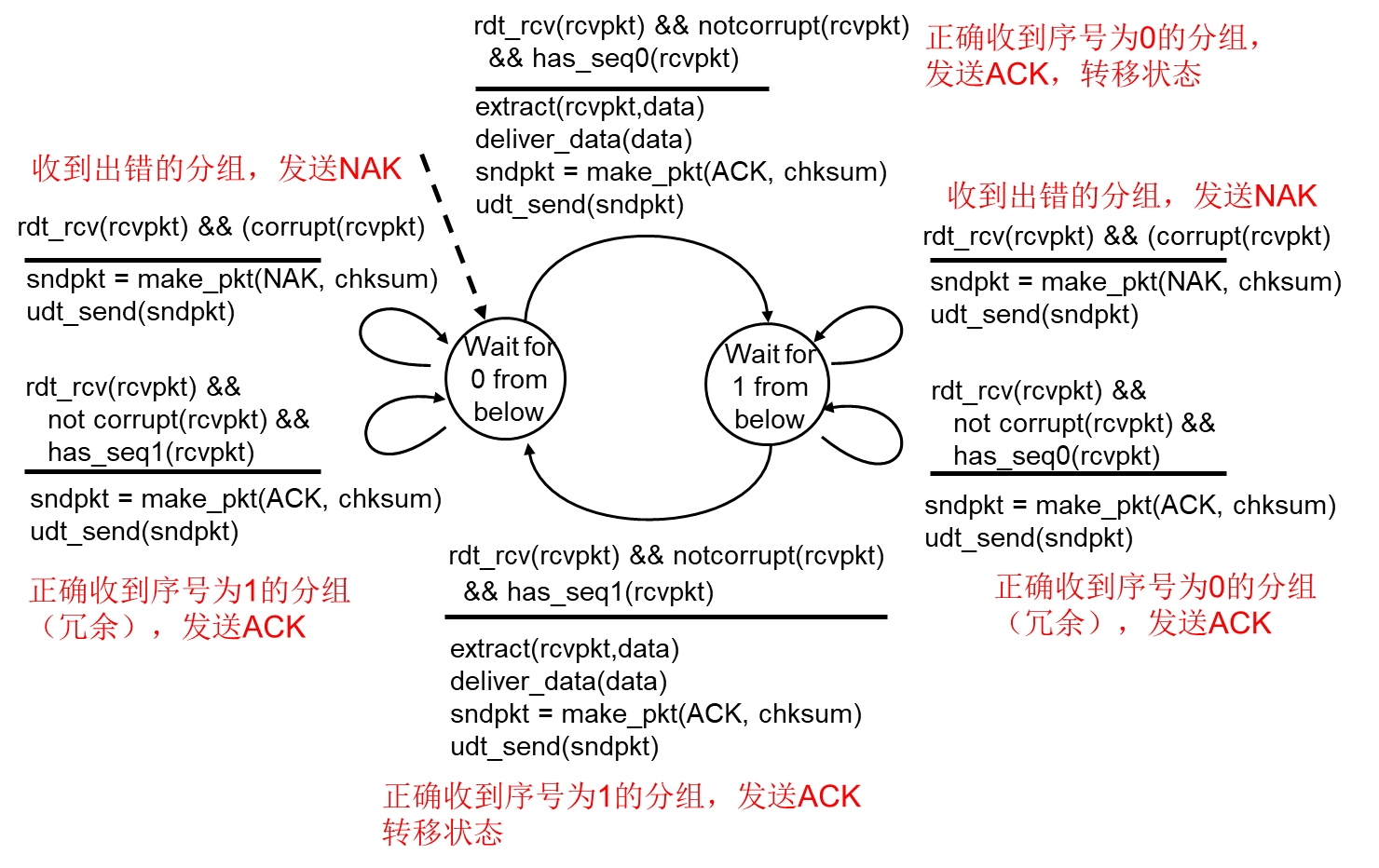
**图示

描述已自动生成可靠数据传输原理：**

**可靠传输协议（Rdt）**的责任（TCP提供的服务模型）：实现“数据可通过**可靠的信道传输**，且借助该信道，数据传输时**不会损坏或丢失**，所有数据均**按其发送顺序交付**”的服务抽象。可靠数据传输协议的*下层*协议或许是不可靠的，这决定了Rdt有其复杂性。

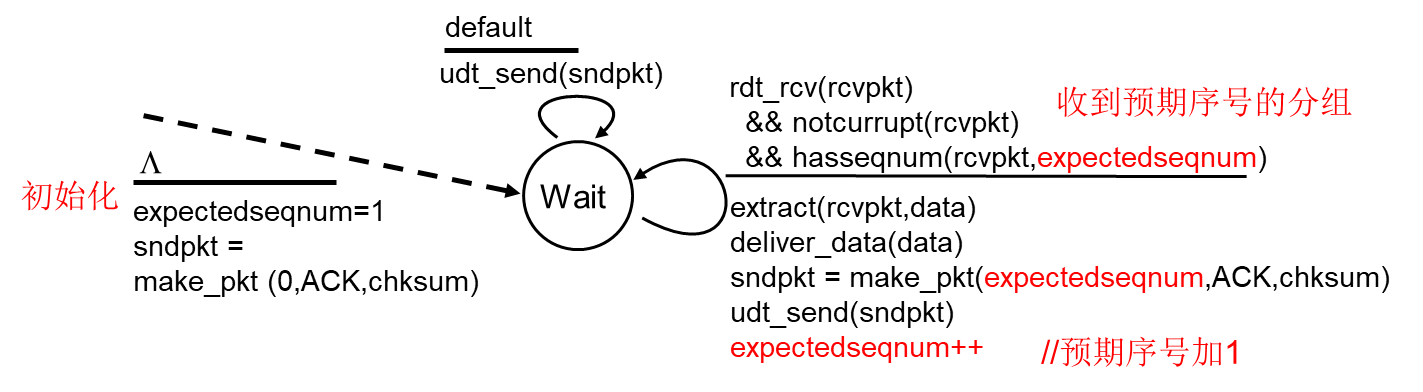
构造可靠数据传输协议（本节FSM箭头旁横线上方为条件，下方为动作）：**Rdt1.0: 可靠信道上的可靠传输：**认为下层信道是完全可靠的（不会重排分组，没有比特错误，没有分组丢失，发送能力≤接收能力）；发送方FSM：上层调用Rdt发送数据时，Rdt将data封装成分组送入下层信道；接收方FSM：从下层信道接收分组，取出数据交给上层。

**Rdt2.0: 可能产生比特错误的下层信道：**下层信道仍不会重排分组，但可能使分组中的比特产生翻转，可以通过检错码（如checksum）检测比特错误。如何检测错误？**肯定确认**（**ACK，**接收方显式地告诉发送方，收到的分组正确）或**否定确认**（**NAK，**接收方显式地告诉发送方，收到的分组有错，发送方收到NAK后重传出错的分组）。Rdt2.0中需要三种新的机制：接收方**检错**，接收方**反馈**，发送方**重传分组**。Rdt2.0这样的协议称为**停等**协议（发送端发一个包后必须等待接收端的反馈。得到反馈前，上层应用不能发新的数据包）。Rdt2.0存在致命的缺陷：没有考虑ACK和NAK受损的可能性。可以在收到一个模糊的ACK/NAK时重传该分组，但这种方法在信道中引入了**冗余分组**：接收方不知道上次发送的ACK/NAK是否被正确收到，进而无法知道收到的分组是新的还是重传。解决这个新问题的方法是在数据分组中添加一个新字段，让发送方对其数据分组编号，将分组的**序号**放在该字段，接收方检测序号以确定是否是重传分组。对于停等协议，1bit序号就够了（0/1）。

******Rdt 2.1：发送方：构造分组、加入序号**；等待反馈：收到NAK或者出错的反馈，重发分组；收到ACK则转移状态。**接收方：**收到出错的分组：发送NAK；收到冗余的分组：发送ACK，不交付数据，不转移状态；收到正确的新分组：交付数据，发送ACK，转移状态。

Rdt2.1

发送端FSM

****图示

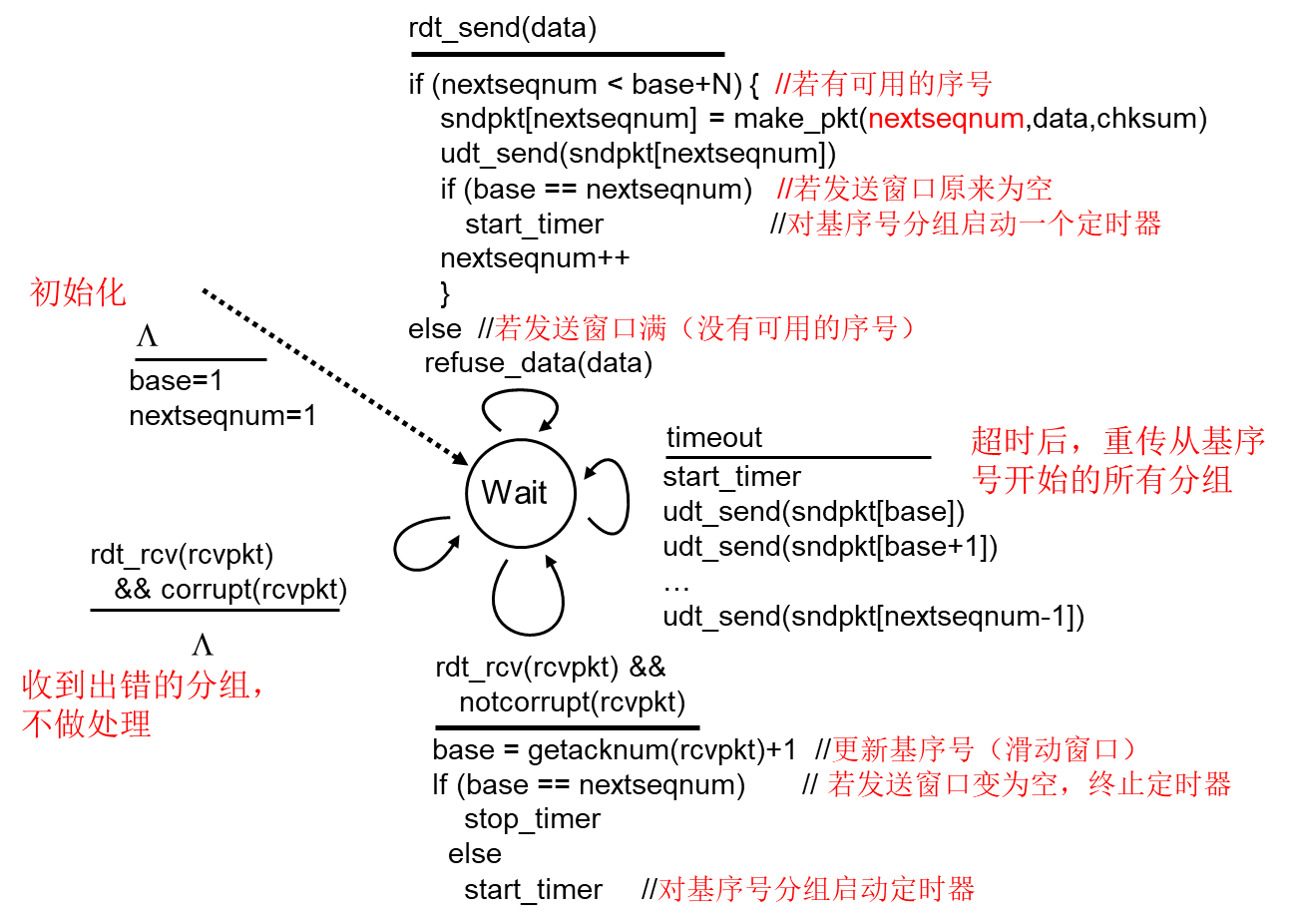
描述已自动生成**Rdt2.2:** 功能同Rdt2.1，但不用NAK。**接收方：**只对正确接收的分组发送ACK，ACK携带**所确认分组的序号**；若收到出错的分组，**重发最近一次的ACK**。**发送方：**收到期待序号的ACK：允许发送下一个分组；其它情况（接收到**冗余ACK**）重发当前分组。

**Rdt3.0：可能产生比特错误和*丢包*的下层信道**（有时称为**比特交替协议**）**：**需要两项新技术：检测丢包（包括**数据及ACK**）和从丢包中恢复。**方法：**检测丢包：若发送方在“合理的”时间内未收到ACK（需要定时器），认为丢包。从丢包中恢复：发送方重发当前分组。**发送方：**发送分组后启动一个定时器；收到期待序号的ACK，终止定时器，转移状态；收到非期待序号的ACK，不做处理；定时器超时（ACK丢失或者延迟过大），重发分组。**接收方：**ACK丢失：接收方感知不到丢包，该事件对接收方FSM无影响；接收方收到数据分组即利用序号检测，若重复则重发前一次ACK，不重复则返回对应的ACK。**接收方FSM同Rdt2.2。**

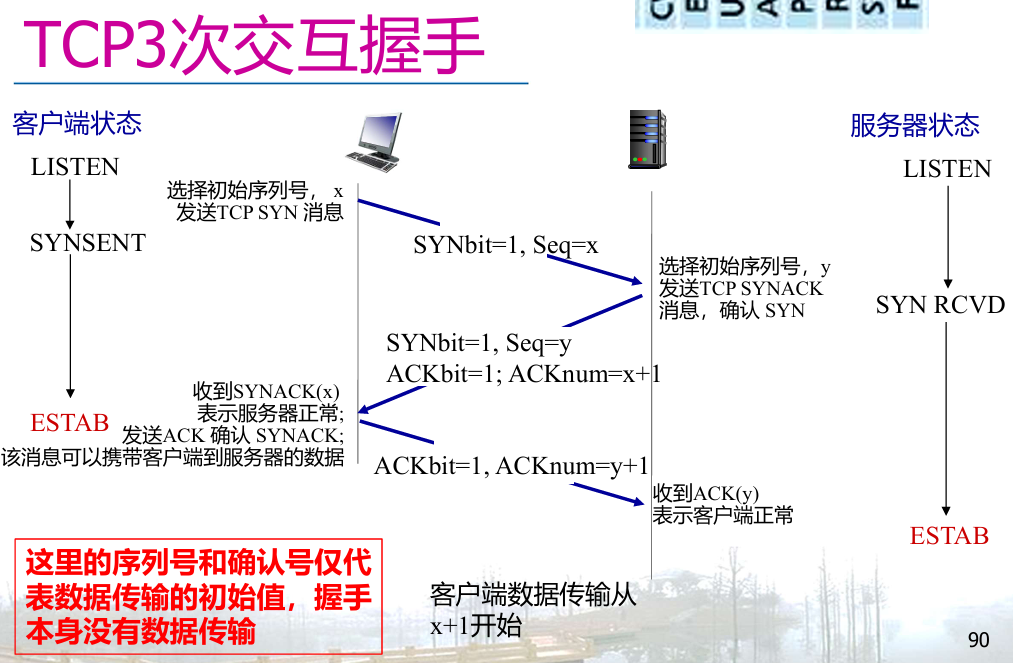
Rdt3.0

发送端FSM

流水线可靠数据传输协议：Rdt3.0是**停等协议**。**发送方（信道）利用率U\_sender：**发送方将数据包送进信道的时间与数据包传输总时间之比。若忽略ACK的**传输**时延，则停等协议的**U\_sender** = (L/R) /(RTT + L/R)，太低了。**解决方案：**采用**流水线**技术，允许发送方发送多个分组而无需等待确认。流水线技术为可靠数据传输协议带来2个**影响：**必须**增加序号范围**；协议的发送方和接收方或必须**缓存多个分组**。所需序号的范围和对缓冲的要求**取决于数据传输协议如何处理丢失、损坏及延时过大的分组**。解决流水线的差错恢复有2种基本办法：**回退N步（GBN）**和**选择重传（SR）**。

回退N步（GBN）协议：允许发送方发送多个分组而无需等待确认，但未确认的分组数**不能超过N个**。定义**基序号**（base，最早的未确认分组的序号）和**下一个序号**（nextseqnum，下一个待发的分组序号），可将序号范围分割成4段：[ 0 , base – 1 ]：已发送并被确认；[ base , nextseqnum – 1 ]：已发送但未确认，[ nextseqnum , base + N – 1 ]：可发送但尚未发送的分组； [base + N, +\inf]：在未被确认的分组被确认之前，均不可用。已被发送但未被确认的分组的序号范围可被视作长度为N的序号窗口，随着协议的运行，该窗口在序号空间向前滑动，故N被称为**窗口长度**，GBN协议也称**滑动窗口协议**。**流量控制**是对发送方施加限制的原因之一。**GBN协议的发送方：**最多允许N个已发送未确认的分组，只对最早的已发送未确认的分组使用一个定时器；若定时器超时，**重传所有已发送未确认的分组**；收到**按序的ACK，更新base**；收到**失序的ACK不做处理**；若**发送窗口从空变为不空**或者**基序号更新**，则**重启定时器。GBN的接收方：**收到一个按序分组时发送一个ACK分组，ACK**携带序号**；接收方**丢弃所有失序分组**，并在丢弃后发送一个**最近按序接收**分组的ACK（接收缓存简单，实现简单；但浪费带宽；之后的重传可能出错，导致更多重传）；使用累积确认：若GBN的ACK包含序号q，则表明“序号截止至q的分组**均正确按序收到**”。**FSM如下：**

图示

描述已自动生成选择重传（SR）：SR中仍存在滑动窗口（设长度为N），但发送方仅重传它认为出错（未收到ACK）的分组，以避免不必要的重传。这种按需重传的策略要求接收方**逐个地确认**正确接收的分组。 **SR的发送方**为每个已发送未确认的分组都设置一个定时器；发送方收到ACK时，若ACK在窗口内，则将分组标记为已发送已接收；如果分组序号还恰好为send\_base，则发送端滑动窗口的基序号“滑动“到序号最小的未确认分组处。**SR接收方**永远确认正确接收的分组，不论其是否按序到达。乱序到达的分组将被缓存，直到所有丢失分组皆被收到为止，这时再将它们一起交给上层。序号在 [ rcv\_base , rcv\_base + N – 1 ]的分组被接收，回发一个选择ACK；序号在 [ rcv\_base – N , rcv\_base – 1 ] 内的分组被接收**也必须产生一个ACK**（如果正确收到分组然而ACK丢失，必须确认发送方重传的分组，否则发送方窗口无法前进）；其他情况忽略分组。对于哪些分组被接收了，哪些没有，发送和接受方看到的结果**并不总相同**，SR协议内的发送方和接收方窗口并不总是一致的。对于SR协议而言，窗口长度必须**小于等于序号空间大小的一半**，否则接收方**可能将重发的分组当作新分组**（**例：**发送方发送了窗口[ 0 , N – 1 ] 中的分组；接收方均正确接收并回发了ACK，并滑动接收窗口至 [ N , 2N – 1 ] ，但N个ACK全部丢失，发送端超时后逐个重发这N个分组。为使接收端不会将重发的分组当成新的分组，窗口[ 0 , N – 1 ]和窗口[ N , 2N – 1 ]不能有重叠。）**SR特点：**出错后重传代价小，接受端需要大量缓存，实现复杂（多个计时器）。

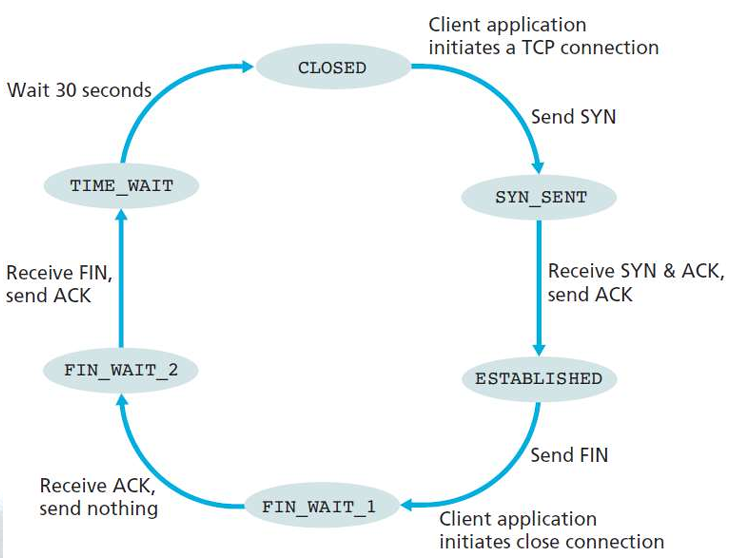
Rdt2.1

接收端

FSM

**面向连接的运输：TCP**

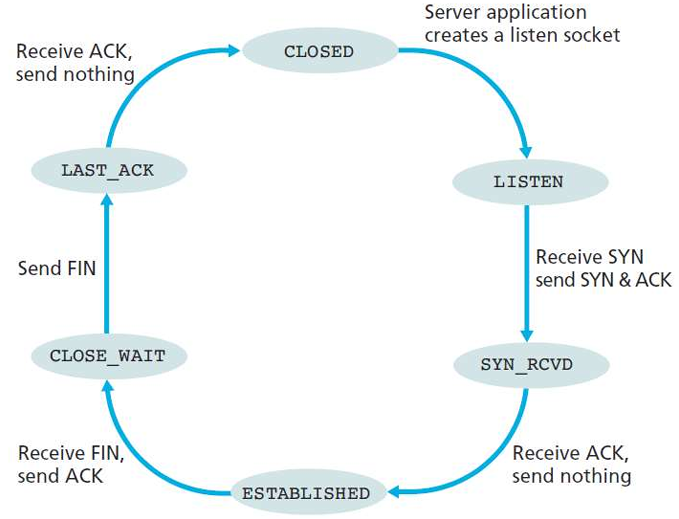
TCP连接：TCP被认为是**面向连接的**，因为在进程A向另一台主机上的进程B发送数据前，二者必须先发送某些预备报文段（三次握手），以建立确保数据传输的参数和状态（套接字、缓存、变量）。不同于电路交换网络，TCP的连接状态仅保留在端系统（的TCP程序）中，是一套逻辑链接，中间的路由器看不到TCP连接。TCP提供**全双工服务：**数据可以在不同主机上的进程A、B间同时双向流动。TCP是**点对点的**，仅支持单个发送方与单个接收方间的连接。发起连接的称为**客户进程**，另一个为**服务器进程**。TCP将客户进程通过套接字的数据存到该链接的**发送缓存**里，在三次握手初期设立。TCP可从缓存中取出并放入报文段的数据数量受限于**最大报文段长度（MSS）**（MSS指报文段中**应用层数据**的最大长度，而非包括TCP首部的TCP报文段的最大长度），MSS常根据最初确定的由本地发送主机发送的最大链路层帧长度（最大传输单元MTU.MTU对以太网而言为1500B，分别减去20B的IPv4 Header和TCP Header剩1460B，即为MSS）来设置。**TCP连接的组成：**发送方和接受方所在主机上的缓存、变量和进程连接的套接字（路由器、交换机和中继器没有为该连接分配任何缓存或变量）。

TCP报文段结构：TCP报文段由**首部字段**和一个**数据字段**组成。数据字段包含一块应用数据，长度受限于MSS。 TCP的**首部**一般20字节，分别是：16bit**源、目的端口号**；32bit**序号**；32bit**确认号**；4bit首部长度（指示以32bit为单位的首部长度，**间接指示TCP选项字段的长度**）；4+2bit保留未用；6bit标志字段**URG**（紧急数据，实际未用）、**ACK**（指示确认字段的值是否有效，1为有效）、**PSH**（推送数据，实际未用）、**RST**、**SYN**、**FIN**；16bit**接收窗口rwnd字段**（用于流量控制，指示接收方愿意接受的字节数量）；16bit因特网校验和；16bit紧急数据指针（实际未用）；可选与变长的**选项字段：**用于发送方与接收方协商MSS，或在高速网络环境下用作窗口调节因子；下面是数据字段。TCP把数据看成一个无结构的、有序的字节流，故**一个报文段的序号**是该报文段数据**首字节**的**字节流编号**。主机A填充进报文段的**确认号**是主机A**期望**从主机B**收到**的**下一个字节的序号**。 TCP**流水式**发送报文段，提供**累计确认**（保证ACK及其之前的报文段无错误），实践中的TCP接收方保留失序的字节，并等待缺少的字节填补间隔。 为了避免“撞车”，一条TCP连接的双方均可以随机选择初始序号。 对客户发送到服务器数据的确认，位于服务器发送到客户的报文段中，这种确认是被**“捎带”**在服务器发送到客户的据报文段。

TCP

client

lifecycle

往返时间的估计与超时：超时间隔必须大于RTT，否则会造成不必要的重传，如何估计超时间隔？报文段的**样本RTT（SampleRTT）**是从某报文段被发出到对该报文段的确认被收到之间的时间。**TCP不对重传的报文段测量RTT（原因：**TCP对接收到的数据而非对携带数据的报文段进行确认，因此TCP的确认是有二义性的，对重传报文段的RTT估计不准确。解决方法是，忽略有二义性的确认，只对**一次就发送成功的**报文段测量SampleRTT ，并据此更新EstimtedRTT。当TCP重传一个段时，停止测量SampleRTT**）**，在任意时刻仅为一个已发送但目前尚未被确认的报文段估计SampleRTT，不是每个都估计。 这样即可计算**EstimatedRTT** = （ 1 – α ）\*EstimatedRTT + α\*SampleRTT（α参考值0.125），是SampleRTT的**指数加权移动平均（EWMA）**。 **DevRTT(RTT偏差)**用于估算SampleRTT偏离EstimatedRTT的程度：DevRTT = （1 – β）\*DevRTT + β\*abs（SampleRTT – EstimatedRTT）（β推荐值0.25）。**最后设置超时值TimeoutInterval =** EstimatedRTT + 4\*DevRTT，初始TimeoutInterval推荐1s，**当出现超时后，TimeoutInterval乘2**（而非上述公式给出的估计值）；一旦报文段**ACK被收到**并更新EstimatedRTT后，**重新用上述公式计算TimeoutInterval。**TimeoutInterval提供了一个形式受限的**拥塞控制**。

TCP

server

lifecycle

可靠数据传输：每个报文段维护一个定时器（如SR）需要相当大的开销，TCP定时器管理过程**仅对第一个已发送未确认的报文段计时**。 若定时器超时，**仅重传最早未确认的报文段**并重新计时。收到ACK时，只要确认序号**大于**基序号，就推进发送窗口，若**仍有已发出未确认报文段**则重新计时。 **超时重传问题之一是：**超时周期可能相对较长，报文段丢失时这种长周期迫使发送方延迟重传丢失的分组，因而增加了端到端时延。 **冗余ACK**就是再此确认某个报文段的ACK，而发送方先前已经收到对它的确认。一旦收到**3个冗余ACK**，TCP执行**快速重传**，在定时器过期之前重传**尚未确认**的**最小序列号**报文段。 TCP的ACK4条生成策略：TCP允许接收端**推迟确认**：对另一个按序报文段的到达最多等待500ms。如果下一个按序报文段在500ms内没有到达，则发送一个ACK。**缺点：**若延迟太大会导致不必要的重传且造成RTT估计不准确； 另外，当收到期望的按序报文段时，若接收方的前一个按序报文段正在等待ACK的传输，则立刻**同时确认两个报文段**。 当**失序**或**重复**报文段到达时，则**立即发送冗余ACK**。 **若可部分或完全填充接收数据间隔的报文段到达**且该报文段**起始于间隔的低端**，则立即发送ACK。 TCP的差错恢复机制是GBN和SR的混合体：TCP缓存乱序到达的分段、仅重传超时的分段（SR）且仅维护一个计数器、累计确认（GBN）。

流量控制：TCP提供**流量控制服务**以保证发送方A不会使接收方B缓存溢出，**双方处理报文段的速率相当**。 TCP**发送方**维护一个**接收窗口rwnd**来提供流量控制，它告诉发送方“接收方还有多少可用的缓存空间”。定义变量**LastByteRead**（B的进程从缓存读出的数据流的最后一个字节的编号）和**LastByteRcvd**（B的接收缓存放入的最后一个字节的编号），**则B的接收窗口rwnd** = RcvBuffer – [ LastByteRcvd - LastByteRead ]，主机A必须满足**LaskByteSent** – **LastByteAcked** <= rwnd才能保证B不溢出。 **该方法的小问题：对零窗口的处理。**当接收窗口为0时，A必须停止发送；当接收窗口变为非0时，B应**通告**增大的接收窗口。**TCP协议规定：**发送方收到“零窗口通告”后，可以发送只有1Byte的“零窗口探测”报文段，以触发一个包含接收窗口的响应报文段。**零窗口探测的实现：**发送端A收到零窗口通告时，启动定时器；定时器超时后，发送端发送一个零窗口探测报文段（序号为上一个报文段中最后一个字节的序号）；接收端在响应的报文段中通告当前rwnd的大小；若发送端仍收到零窗口通告，重新启动定时器。**UDP不提供流量控制，接收缓存可能溢出并丢失报文段。**

TCP连接管理：3次握⼿能防⽌历史连接的建⽴，减少双⽅不必要的资源开销，帮助双⽅同步初始化序列号，保证数据包不重复、不丢失和按序传输，不能少于3次。**TCP连接方式（3次握手）：①**客户端TCP向服务器发送SYN为1、不包含应用数据的**SYN报文段，其序号字段是**随机选择的初始序号client\_isn。**②**服务器收到①中报文后，为TCP连接分配TCP缓存和变量，向客户发送一个允许连接的**SYNACK报文段**：SYN和ACK置1，确认号字段ACKnum为client\_isn + 1，服务器生成自己的server\_isn，放入序号字段。**③**客户收到②中报文后，为自己分配缓存和变量，向服务器发送另一报文段： ACK = 1，SYN = 0，确认字段为server\_isn + 1。不同于①和②，③的**报文段可以负载数据**。此后的每个报文段中SYN都为0。 **TCP关闭连接：**客户应用进程可以发出一个关闭连接命令，这会引起客户TCP向服务器进程发送一个特殊的TCP报文段：FIN置1，Seq = x），服务器收到后，向发送方回送一个确认报文段（ACK=1，ACKnum=x+1），然后服务器发送它自己的终止报文段（FIN=1，Seq=y），最后客户对这个服务器的中止报文段进行确认（ACK=1，ACKnum=y+1）。此时连接的所有资源都被释放。假如服务器想拒绝连接请求，可以向链接发起端发送拒绝连接分段（RST = 1）。

**拥塞控制原理：**

拥塞原因与代价：**拥塞造成：**丢包（缓冲区溢出），延迟增大（链路接近满载，排队时间长）；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：**网络负载很重，但网络吞吐量很低。

拥塞控制方法：**网络辅助的拥塞控制：**路由器向端系统提供反馈（拥塞指示比特，发送速率指示），ATM采用此类方法。 **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推断拥塞的发生，**TCP采用此类方法**（因为IP层不提供显式反馈）。

**TCP拥塞控制：**

运行在**发送方**的TCP拥塞控制跟踪变量**拥塞窗口（congestion window，cwnd）**来控制发送方速率，特别是 LastByteSent – LastByteAcked <= min { cwnd , rwnd }。只考虑cwnd时，粗略的讲，在每个RTT的起始点，限制条件允许发送方发送cwnd个字节的数据，在该RTT结束时发送方接收对数据的确认报文，发送方的**速率大概是 cwnd/RTT Bytes/sec**。**本节讨论两个TCP拥塞控制算法：Tahoe和Reno。**总的指导思想：加性增、乘性减(AIMD)。

**Tahoe：①慢启动：**cwnd = 1 MSS（此时发送速度= MSS/RTT）。**每收到一个ACK**，cwnd就**增加1个MSS**，即**每过1个RTT**，**cwnd就翻倍**。若出现超时丢包/收到3个重复ACK，则cwnd设为1，ssthresh设为原cwnd的一半，**重新开始慢启动**。当cwnd达到ssthresh值时，进入**拥塞避免**状态。 **②拥塞避免：**每收到一个新的ACK，发送端cwnd增加MSS\*(MSS/cwnd)，即**每个RTT增加cwnd一个MSS**（线性增长）。若出现超时丢包/收到3个重复ACK，则cwnd变为1，ssthresh变为原cwnd的一半，回到慢启动状态。

**Reno：①慢启动：**cwnd = 1 MSS。**每收到一个ACK**，cwnd就**增加1个MSS**，即**每过1个RTT**，**cwnd就翻倍**。若出现超时丢包，则cwnd设为1，ssthresh设为原cwnd的一半，**重新开始慢启动**。若cwnd达到ssthresh值，则进入**拥塞避免**状态。若收到3个重复ACK，则将cwnd减半再+3，进入快速恢复状态。 **②拥塞避免：**每收到一个新的ACK，发送端cwnd增加MSS\*(MSS/cwnd)，即**每个RTT增加cwnd一个MSS**（线性增长）。若出现超时丢包，则cwnd变为1，ssthresh变为原cwnd的一半，回到慢启动状态。**③快速恢复状态：**线性增长，3个重复ACK转到拥塞避免，超时转到慢启动（详情请看FSM）。

**TCP的吞吐量的宏观描述**（忽略慢启动）：令W=发生丢包时的cwnd，此时吞吐量= W/RTT；发生丢包后令cwnd=W/2，则此时吞吐量=W/2RTT；假设在TCP连接持续期间内，RTT和W几乎不变，则由于总是在极值之间线性变化，有**平均吞吐量=0.75 W/RTT。 经高带宽路径的TCP：**吞吐量与丢包率L的关系：吞吐量=1.22\*MSS / （RTT\*Sqrt[ L ]）；针对高速网络需要新的TCP拥塞控制算法。

**公平性：公平性的目标:** 如果K条TCP连接共享某条带宽为R的瓶颈链路，每条连接平均吞吐率应为R/K。 **TCP是公平的。** 若相互竞争的TCP连接具有不同的参数（RTT、MSS等），**不能保证公平性**；若应用（如web）可以建立多条**并行**TCP连接，不能保证带宽在应用之间公平分配（比如，一条速率为R的链路上有9条连接；若新应用建立一条TCP连接，获得速率 R/10，若新应用建立11条TCP，可以获得速率R/2）。【到这里了】

***小测题和易错点：***

**易错点：**kbps = 1000bps。 平均**传输**次数=平均**重传**次数+1

**小测答案：**

常用的通信传输介质有哪些？它们之间的主要区别？（1）有线：双绞线、同轴电缆、光纤；无线 （2）区别：带宽、误码率、传输距离、价格、频谱及复用方式、是否支持移动通信等。无连接分组交换与面向连接（虚电路）分组交换的区别？（1）分组格式:前者完全源、目的地址；后者虚电路号 （2）路由表：前者面向整个网络拓扑，转发时顺序查找路由表；后者面向特定路径或源路由，转发基于索引查找路由表。 （3）可靠性、顺序性：前者无；后者有 （4）建立、维护连接：前者无；后者有假定要传送的报文共有x（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（或信令）长度、在各节点的排队时延忽略不计；分组交换使用的分组头、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 数据传播时间：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，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 …若使用一个256-kbps 的无差错卫星信道（往返传播时延为512-msec）一个方向上发送512-byte 数据帧，而在另一个方向上返回很短的确认帧。则对于窗口大小为1, 15, 127的最大吞吐量是多少?512\*8/256k=16ms （1）k=1,16/（16+512）\*256=7.75 （2）k=15,7.75\*15=116.36 （3）k=127,256滑动窗口协议中,退后N帧与选择性重传利用链路缓冲能力连续发送多个帧,令帧的传输时间（transmission time）=1（归一化）、传播时间（propagation time）=a，则链路的缓冲能力为？a（单向）或2a（双向）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 TCP连接的目标 （1）实现进程间通信 （2）实现可靠传送 （3）实现按序传送 （4）进行流量控制 （5）进行拥塞控制在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100字节数据的报文段，然后关闭连接。试问下面从客户端发送的各个报文段的序号分别是多少？ （1）SYN报文段；（2）数据报文段；3）FIN报文段。（1）215；（2）216；（3）316在一条新建的TCP连接上发送一个长度为32KB的文件。发送端每次都发送一个最大长度的段（MSS），MSS的长度为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门限设为16KB。假设发送端尽可能快地传输数据，即只要发送窗口允许，发送端就发送一个MSS。（1）已知发生第一次超时后，发送端将拥塞窗口门限调整为4KB。请问发生超时的时候，发送端的拥塞窗口是多大？此时发送端共发送了多少数据？其中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） 第一次超时发生时，发送端拥塞窗口大小 = 4KB\*2 = 8KB 在新建立的TCP连接上，发送端采用慢启动开始发送，因此当第一次超时发生时，发送端已发送的数据量 = 1KB + 2KB + 4KB + 8KB = 15KB。此时，除最后一批8个TCP段未获确认外，之前发送的TCP段都被确认，因此成功确认的数据量为7KB。（2） 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；收到全部6个MSS的确认后，拥塞窗口增至7KB；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7KB。