***Chapter 6 Transport Layer***

大纲

传输服务

传输协议的要素

拥塞控制

因特网传输协议*UDP*

因特网传输协议*TCP*

6.1传输层协议概述

引入传输层的原因

消除网络层的不可靠性；

提供从源端主机到目的端主机的可靠的、与实际使用的网络无关的信息传输。

传输服务

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高层提供有效、可靠的服务；

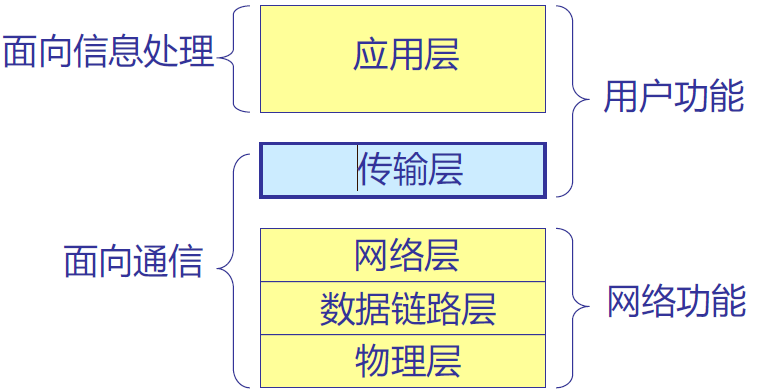
传输层提供两种服务

面向连接的传输服务：连接建立，数据传输，连接释放

无连接的传输服务。

1 ~ 4层称为传输服务提供者（transport service provider），4层以上称为传输服务用户（transport service user）。

从通信和信息处理的角度看，传输层向它上面的应用层提供通信服务，它属于面向通信部分的最高层，同时也是用户功能中的最低层。



应用进程之间的通信

两个主机进行通信实际上就是两个主机中的应用进

程互相通信。

应用进程之间的通信又称为端到端的通信。

传输层的一个很重要的功能就是复用和分用。应用层不同进程的报文通过不同的端口向下交到传输层,再往下就共用网络层提供的服务。

“传输层提供应用进程间的逻辑通信”。

“逻辑通信”的意思是：传输层之间的通信好像是沿水平方向传送数据。但事实上这两个传输层之间并没有一条水平方向的物理连接。

传输层与网络层

传输服务屏蔽不同的网络

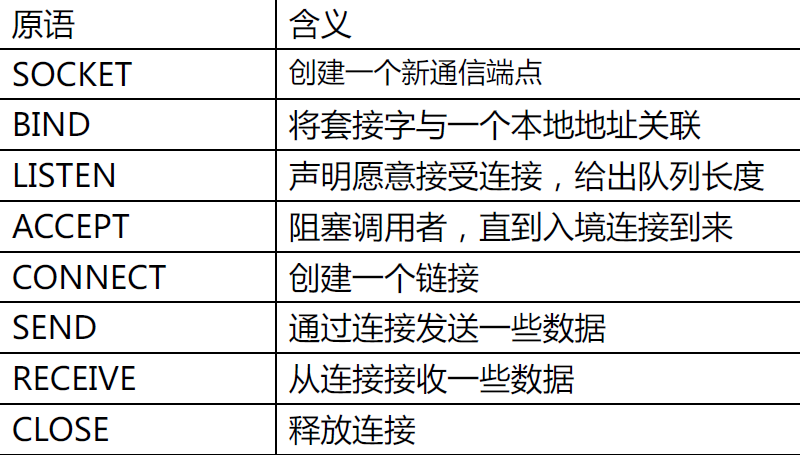
面向连接服务：不可靠的网络服务上建立可靠服务;无连接服务

服务对象：程序只能看到传输服务，无法控制网络

原语: LISTEN;CONNECT;SEND;RECEIVE; DISCONNECT

段（传输协议数据单元TPDU）—传输实体间发送的消息

TCP所用套接字(socket)原语



6.2 传输协议的要素

与数据链路层类似

相同点：处理错误控制、顺序性、流控制

不同点：指定目的地址、初始连接更为复杂、子网存储能力导致乱序接收、缓冲区数量可变

要素

1. 寻址;2. 建立连接;3. 释放连接;4. 流量控制和缓冲策略;5. 多路复用;6. 崩溃恢复

6.2.1 寻址

方法：定义传输服务访问点TSAP（Transport Service Access Point），将应用进程与这些TSAP相连。在Internet中，TSAP为（IP address, local port）；

远方客户程序如何获得服务程序的TSAP？

方法1：预先约定、广为人知的，像telnet是（IP地址，端口23）；

方法2：从名字服务器（name server）或目录服务器（directory server）获得TSAP

一个特殊的进程称为**名字服务器**或**目录服务器**（TSAP众所周知）；

用户与名字服务器建立连接，发送服务名称，获得服务进程的TSAP，释放与名称服务器的连接；

与服务进程建立连接。

当服务程序很多时，使用初始连接协议（initial connection protocol）

一个称为进程服务器（process server）的进程(inetd)同时在多个端口上监听；

远方客户程序向它实际想访问的服务程序的TSAP发出连接建立请求；

如果没有服务程序在此TSAP上监听，则远方客户和进程服务器建立连接；

进程服务器产生所请求的服务进程，并使该进程继承和远程客户的连接；

进程服务器返回继续监听；

远方客户程序与所希望的服务程序进行数据传输。 Fig. 6-9

6.2.2 建立连接

网络问题：分组可能丢失、延迟、损坏、重复出现

废弃使用过的传输地址

分配一个连接标识符

限制数据包在网络中的生存期

限制网络设计;放置跳计数器;时间戳

区分重复段

段标签：序号在周期T秒内不重复;日时钟：永远运行，时钟低位作为开始序号

日时钟（time-of-day）方案

对于任何一个序列号，在它有可能被用做初始序列号之前的T秒（分组周期上限）时间内，禁止使用该序列号——确保在同一时刻永远不会出现两个编号相同的TPDU。

防止序号进入禁止区域:发送速度不能太快;发送速率不能低于时钟速率

三次握手

时间戳：辅助扩展32位序号

防止序号回绕(PASW)：伪随机初始序号;防止攻击者预测到下一个初始序号

6.2.3 释放连接

释放

非对称;任何一方可DISCONNECT，其TPDU到达对方-连接释放;右图导致数据丢失

对称：各方单独DISCONNECT,停止发送但继续接收;双方均DISCONNECT后释放连接

释放连接的协议场景: 正常情况;超时释放; 第二次成功;重传全部失败

如果初始DR和N次重传全部丢失，则协议失败:发送端放弃发送强行释放，导致半开放连接

不允许发送端放弃:允许超时—导致永久释放操作;不允许超时—协议将被悬挂

传输实体本身无法彻底解决这一问题:传输层用户必须要参与进来决定何时断开连接

6.2.4 差错控制和流量控制

差错控制: 确保数据传输的可靠性

流量控制:防止快速发送端淹没慢速接收端

与数据链路层协议不同

数据链路层:CRC、自动重发请求、停等、滑动窗口

功能上的差异: 链路层校验和只保护在一条链路上进过的数据包，路由器错误在链

路层校验之外；相比链路层校验，端到端的传输层校验至关重要

程度上的差异:

1) 链路的时延带宽积很小，有线与光纤链路误码很低可忽略重传；

2)传输层连接的时延带宽积较大，需使用较大的滑动窗口

流量控制和缓冲策略

流量控制是发送方和接收方之间的传输速率上的匹配，为使没有得到确认的PDU在超时后的重发，通常必须在缓冲区中暂存

在数据链路层，实现的是点对点的通信，双方缓冲区的大小根据滑动窗口协议而定

而传输层实现的是端到端的通信，某一时刻，一台主机可能同时与多台主机建立了连接，多个连接必须有多组缓冲区，所以缓冲区的动态分配和管理策略与数据链路层相比

较要复杂得多

流量控制由缓冲区的容量来决定的

流控和缓存(Flow Control and Buffering)

缓存：由于网络层服务是不可靠的，传输层实体必须缓存所有连接发出的TPDU，而且为每个连接单独做缓存，以便用于错误情况下的重传。接收方的传输层实体既可以做也可以不做缓存。缓存区的设计有三种。Fig. 6-15

流控：传输层利用可变滑动窗口协议来实现流控。所谓可变滑动窗口协议，是指发送方的发送窗口大小是由接收方根据自己的实际缓存情况给出的。为了避免控制TPDU丢失导致死锁，主机应该周期性的发送TPDU。Fig. 6-16

缓冲池的组织方式

a. 段的长度都差不多——链接固定大小的缓冲区

b. 段的长度差异较大—— 链接可变大小的缓冲区—管理复杂

c. 每条连接使用一个大的循环缓冲区

6.2.5 多路复用

向上多路复用：多个传输连接使用同一网络连接

向下多路复用：一个传输连接使用多个网络连接

6.2.6 崩溃恢复

网络崩溃的恢复

主机崩溃的恢复

确认和写操作的问题

网络崩溃的恢复:

数据报子网：如果传输层对丢失的TPDU留有副本，可以通过重发来解决

虚电路子网：必须重新建立连接，并询问远端的传输实体哪些TPDU已经收到，没有收到的则必须重发

主机崩溃的恢复

客户端主机崩溃，恢复将较为简单

服务器崩溃，如能及时重新起动，则如何恢复与原客户主机的数据传输，问题比较严重

重新连接后，客户端可能处于两种状态之一：

S1—有一个未被确认的TPDU

S0—没有未被确认的TPDU

在一般情况下，远端服务器的传输实体在接收到一个客户端的TPDU后，先发送一个确认，再对应用进程执行一个写操作（如存盘或交上层）。发送一个确认和执行一个写操作，是两个不同的而又不可分的事件，但两者不能同时进行

确认和写操作的问题

先发送确认，然后再执行写操作，中间发生崩溃此时客户端将收到这个确认，当崩溃恢复声明到达时它处于状态S0，客户端将因此不再重发，因为它错以为那个TPDU已经到达服务器端，客户端的这种决定会导致丢失一个TPDU

先执行写操作，然后再发送确认，中间发生崩溃设想已经完成了写操作但在确认发出前系统发生了崩溃，此时客户端将处于状态S1并因此重传数据，从而会导致在服务器应用进程的输出流上出现一个无法检测的重复的TPDU

无论怎样对发送方和接收方的协议进行编程，总是存在协议不能正确地从故障中恢复的情况，传输层无法彻底解决该问题，将由高一层协议处理，由应用进程来解决。

崩溃恢复处理

发送主机状态:S0：没有未被确认的TPUD;S1：有一个未被确认的TPUD

接收主机策略:先确认后写：AC(W)、AWC、C(AW);先写后确认：WC(A)、WAC、C(WA)

发送主机策略:总是重发;从不重发;S0状态下重发;S1状态下重发

结论:图6-18（P408）; 崩溃恢复只能由上一层完成

6.3 拥塞控制

理想的带宽分配;调整发送速率;无线问题

理想的带宽分配: 效率与功率;功率=负载/延迟

最大-最小公平性:均等带宽比例

调整发送速率:显式拥塞协议; 加法递增乘法递减(AIMD)

无线问题: 拥塞与传输错误:TCP连接(AIMD)丢失率：0%~1%~10%;802.11正常丢失率：10%

屏蔽策略：设置两种机制作用的时间尺度不同

链路层重传在微秒到毫秒；传输层丢失毫秒到秒

6.4 因特网传输协议UDP: UDP介绍; 远过程调用; 实时传输协议

UDP介绍:用户数据报协议;UDP头部; IPv4伪头部

用户数据报UDP 有两个字段：数据字段和首部字段。首部字段有8 个字节，由4 个字段组成,每个字段都是两个字节.

在计算检验和时，临时把“伪首部”和UDP 用户数据报连接在一起。伪首部仅仅是为了计算检验和。

RPC

客户存根：封装参数（列集）;服务器存根：解封参数（散集）

6.4.3 实时传输协议

RTP(RFC3550)—广泛应用于多媒体应用程序

RTP协议:以数据包的形式传输音频和视频数据

接收端:在正确的时间播放音频和视频

RTP——是一个在应用层上实现的传输协议: 将多个实时数据流复用到一个UDP数据段中;时间戳：网络延迟变化，多个流之间同步

RTCP——实时传输控制协议:向源端提供网络特征反馈信息;延迟、抖动、带宽、拥塞等;

带有缓冲和抖动控制的播放:

播放点:最小延迟、平均延迟;等待时间：高抖动、低抖动

6.5 因特网传输协议TCP

TCP概述

TCP服务模型

TCP协议

TCP段的头

TCP连接建立

TCP连接释放

TCP连接管理模型

TCP滑动窗口

TCP计时器管理

TCP拥塞控制

TCP未来

6.5.1 TCP概述

传输控制协议TCP（Transmission Control Protocol）

面向连接的、可靠的、端到端的、基于字节流的传输协议；

RFC 793+，1122，1323，2018，2581，2873，2988等。

在不可靠的互联网上提供可靠的端到端服务

TCP实体:一个库过程; 一个用户进程;内核的一部分

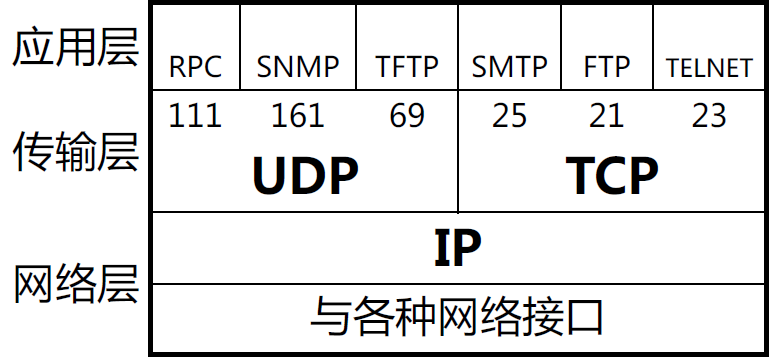
6.5.2 TCP服务模型

TCP服务由发送端和接收端创建一种称为套接字（socket）的端点获得。

套接字: 主机ip地址+16位数值的端口号

一个套接字可用于多个连接

TCP连接都是全双工的，且是点到点的:是字节流，不是消息流



端口的概念

端口就是传输层服务访问点TSAP。

端口的作用就是让应用层的各种应用进程都能将其数据通过端口向下交付给传输层，以及让传输层知道应当将其报文段中的数据向上通过端口交付给应用层相应的进程。

从这个意义上讲，端口是用来标志应用层的进程

端口标志

端口用一个16 bit 端口号进行标志。

端口号只具有本地意义，即端口号只是为了标志本计算机应用层中的各进程。在因特网中不同计算机的相同端口号是没有联系的。

有两类端口:一类是熟知端口，其数值一般为0~1023。当一种新的应用程序出现时，必须为它指派一个熟知端口。

另一类则是一般端口，用来随时分配给请求通信的客户进程。

插口(socket): TCP 使用“连接”(而不仅仅是“端口”)作为最基本的抽象，同时将TCP连接的端点称为插口(socket)，或套接字、套接口。

同一个名词socket有多种不同的意思:

应用编程接口API称为socket API, 简称为socket。

socket API 中使用的一个函数名也叫作socket。

调用socket 函数的端点称为socket。

调用socke t函数时其返回值称为socket描述符，可简称为socket。

在操作系统内核中连网协议的Berkeley 实现，称为socket 实现。

6.5.3 TCP协议

每个字节都有自己独有的32位序号

以段形式交换数据

TCP段: 20字节的头;IP有效载荷：65515B; 网络最大传输单元（MTU）;某条链路有更小MTU时——路径MTU发现:利用ICMP错误消息发现某条路径上任意一条链路的最小MTU。

使用动态窗口大小的滑动窗口协议

TCP段的结构:净载荷 65535-20-20=65495

建立连接采用三次握手方法

SYN泛洪攻击: 恶意用户发送SYN连接请求，且故意不做连接建立

6.5.6 TCP连接释放

释放连接: 发送设置了FIN标志位的TCP段;三个TCP段:FIN, ACK+FIN,ACK

TCP滑动窗口: 接收端4096B

窗口为0:紧急数据可发送;窗口探测:强制宣告窗口大小

解决方案

强制接收端有一定数量空间后再通告对方;如，缓冲区一半为空,发送端不发太小的段

6.5.9 TCP计时器管理—重发计时器

TCP在发送一个数据段的同时，启动一个数据重发计时器，如果在计时器超时前该数据段被确认，则关闭该计时器

如果在确认到达之前计时器超时，则需要重发该数据段（并且该定时器重新开始计时）

重发计时器初始值的设定

与数据链路层点对点的情况不同，在TCP层，源站点与目的站点之间的网络距离是动态变化的，数据传输和信号传播时间的离散性很大，线路的状态更是瞬息万变

所以，重发计时器的初始值不宜设定为固定大小

延时分布比较说明

对于点对点的数据链路层，延时由数据传输延时和信号传播延时两部分组成，对这两部分延时的估计基本上是准确的（即误差很小），所以计时器初始值设定为大于估计的确认返回时间即可，如图6-42(a)所示

TCP所面临的是完全不同的情况，端到端的连接可能远隔重洋，需经过很多路由器的存储转发，途经路由器的实际情况又是动态变化的，所以TCP确认返回所需时间的概率密度函数更接近于图6-42(b)所示

计算新的往返时间估计值

TCP不使用固定的重发计时器，而是根据对网络性能的不断测定，主要是对远程的确认报文作延时分析，不断调整超时间隔，自适应地修正往返时间的

估计值：RTT（Round-Trip Timer）

重传定时器算法

平滑往返时间估计值SRTT的确定

为尽可能避免因实际延时较大而进行的错误重发，同时尽可能提高系统的吞吐率，所以适应性重发机制中往返时间估计值RTT的确定将根据前一次的估计值RTT0 ，并参考实测的往返时间R作动态调整



其中：SRTT：前一次计算得到的平滑往返时间估计值

R：前一次实测得到的往返时间

a：修正因子，RTT0和M0的参考权值（通常取a = 7/8）

原来固定为βRTT

固定为往返时间估计值RTT的β倍，初期β= 2，但经验表明常量是很不灵活的，当环境发生变化时不能很好地适应

现在：往返时间变化RTTVAR

RTTVAR = β×RTTVAR + (1−β) | SRTT −R |

超时值: RTO = SRTT + 4×RTTVAR

如接收方向发送方发出一个零窗口公告，发送方当即停止发送，当接收方的上层处理了所收到的数据并释放了全部的缓冲区后，将向发送方发出一个新的窗口公告，以通知发送方可继续发送，如果此更新公告丢失，双方将相互等待，出现死锁

为防止死锁，每当发送方收到一个零窗口公告，即启动一个持续定时器，如持续定时器超时，发送方将向接收方发一探测报文（仅一字节） ，接收方的应答报文将避免相互等待(这正是上面所说：即使在发送方已收到一个零窗口通告，也允许发送一个字节的数据报询问对方）

一旦建立连接，双方都将启动“ 保活”定时器，当连接长时间无数据传输，所设定的“ 保活”定时器超时，超时一方会发送一个探测报文检查对方是否存在，如没有得到响应，则终止连接

6.5.10 TCP拥塞控制

TCP作用: 在拥塞控制和可靠传输中发挥主要作用

拥塞窗口:采用AIMD(加增乘减)规则调整窗口大小

发送窗口= min(拥塞窗口，接收窗口)

确认时钟(ACK时钟)

拥塞窗口计算

例10Mbps网络：RTT=100ms

拥塞窗口= 带宽延迟积= 1Mb = 100个×1311B/个(1250？)

拥塞窗口从1开始，每RTT加1，10秒才能达到100

慢速启动

拥塞窗口(cwnd)初始值：早期为1个段，后来为4个段

发送端:按初始值发送段，每收到一个未超时确认，cwnd++

慢速启动控制：慢启动阈值(或阀值)

慢速启动阈值

慢速启动阈值初始为流量控制窗口大小

超时后，阈值变为当前拥塞窗口的一半，然后重新启动

一旦慢速启动超过阈值，TCP从慢启动切换到线性增加:每一RTT，cwnd++

方案缺陷:

等待超时

超时时间太长数据包丢失后，确认号不变，发送端因拥塞窗口已满，无法发送新包

重复确认:三个重复确认意味着丢包

快速重传:重传计时器超时前重发

TCP Tahoe

1988年发布的4.2BSD TCP Tahoe

最大段长：1KB

拥塞窗口初值：64KB

TCP Reno

恢复:个重复确认

1990年发布:4.3BSD Reno

选择确认SACK

拥塞避免（congestion avoidance）算法

若拥塞窗口大于阈值，从此时开始，拥塞窗口线形增长，一个RTT周期增加一个最大段长，直至发生丢包超时事件；

当超时事件发生后，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；

执行慢启动算法。

发送端的主机在确定发送报文段的速率时，既要根据接收端的接收能力，又要从全局考虑不要使网络发生拥塞。

因此，每一个TCP 连接需要有以下两个状态变量：

接收端窗口rwnd (receiver window) 又称为通知窗口(advertised window)。

拥塞窗口cwnd (congestion window)。

(1) 接收端窗口rwnd ，这是接收端根据其目前的接收缓存大小所许诺的最新的窗口值，是来自接收端的流量控制。接收端将此窗口值放在TCP 报文的首部中的窗口字段，传送给发送端。

(2) 拥塞窗口cwnd (congestion window) 是发送端根据自己估计的网络拥塞程度而设置的窗口值，是来自发送端的流量控制。

发送端的发送窗口的上限值应当取为接收端窗口rwnd 和拥塞窗口cwnd 这两个变量中较小的一个，

即应按以下公式确定：发送窗口的上限值= Min [rwnd, cwnd]

当rwnd < cwnd 时，是接收端的接收能力限制发送窗口的最大值。

当cwnd < rwnd 时，则是网络的拥塞限制发送窗口的最大值。

慢启动算法的原理

在主机刚刚开始发送报文段时可先将拥塞窗口cwnd 设置为一个最大报文段MSS 的数值。

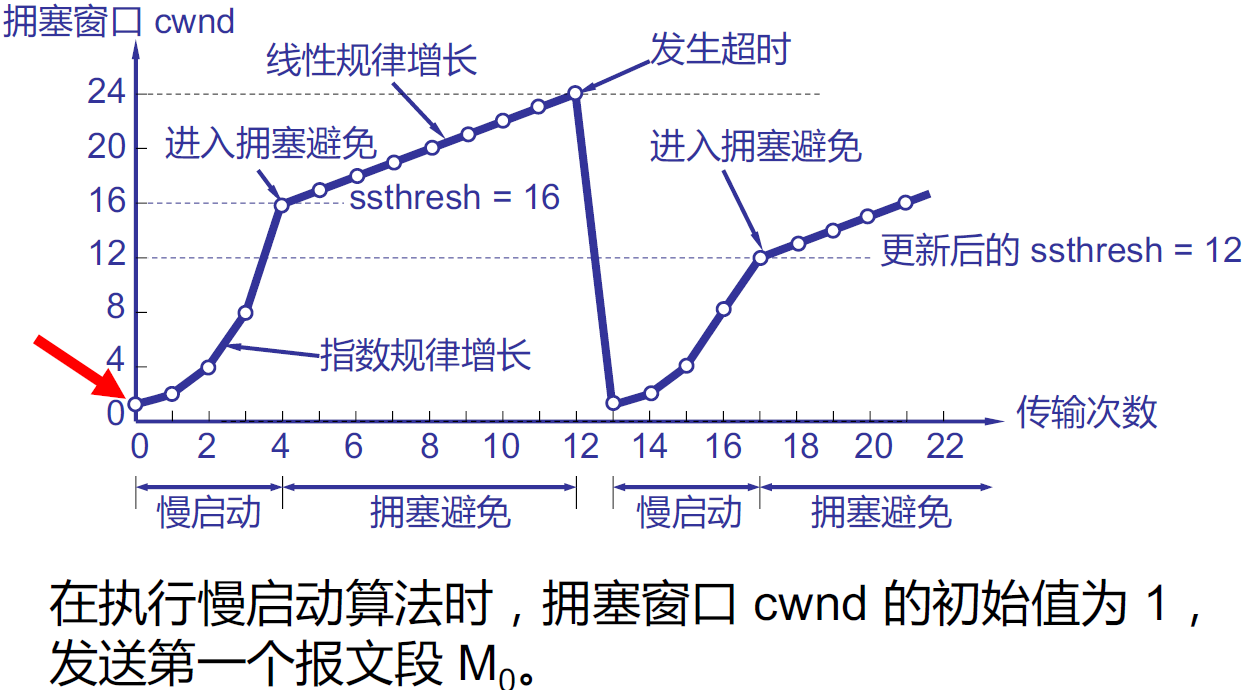
在每收到一个对新的报文段的确认后，将拥塞窗口增加至多一个MSS 的数值。

用这样的方法逐步增大发送端的拥塞窗口cwnd，可以使分组注入到网络的速率更加合理。

当TCP 连接进行初始化时，将拥塞窗口置为1。图中的窗口单位不使用字节而使用报文段。

慢启动门限的初始值设置为16个报文段,即ssthresh=16

发送端的发送窗口不能超过拥塞窗口cwnd 和接收端窗口rwnd 中的最小值。我们假定接收端窗口足够大，因此现在发送窗口的数值等于拥塞窗口的数值。



发送端收到ACK1 （确认M0，期望收到M1）后，将cwnd 从1 增大到2，于是发送端可以接着发送M1 和M2 两个报文段。

接收端发回ACK2 和ACK3。发送端每收到一个对新报文段的确认ACK，就把发送端的拥塞窗口加倍。现在发送端的cwnd 从2 增大到4，并可发送M4 ~ M6共4个报文段。

发送端每收到一个对新报文段的确认ACK，就把发送端的拥塞窗口加倍，因此拥塞窗口cwnd 随着传输次数按指数规律增长。

当拥塞窗口cwnd 增长到慢启动门限值ssthresh 时（即当cwnd = 16 时），就改为执行拥塞避免算法，拥塞窗口按线性规律增长，窗口每次增长1。

假定拥塞窗口的数值增长到24 时，网络出现超时（表明网络拥塞了）。

更新后的ssthresh 值变为12（即发送窗口数值24 的一半），拥塞窗口再重新设置为1，并执行慢启动算法。

当cwnd = 12 时改为执行拥塞避免算法，拥塞窗口按按线性规律增长，每经过一个往返时延就增加一个MSS 的大小。

乘法递减(multiplicative decrease)

“乘法递减”是指不论在慢启动阶段还是拥塞避免阶段，只要出现一次超时（即出现一次网络拥塞），就把慢启动门限值ssthresh 设置为当前的拥塞窗口值乘以0.5。

当网络频繁出现拥塞时，ssthresh值就下降得很快，以大大减少注入到网络中的分组数。

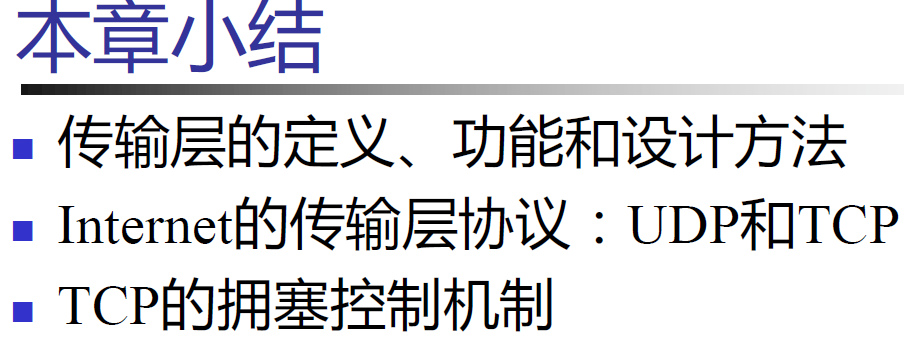
加法递增(additive increase)

“加法递增”是指执行拥塞避免算法后，当收到对所有报文段的确认就将拥塞窗口cwnd增加一个MSS大小，使拥塞窗口缓慢增大，以防止网络过早出现拥塞。

必须强调指出

“拥塞避免”并非指完全能够避免了拥塞。利用以上的措施要完全避免网络拥塞还是不可能的。

“拥塞避免”是说在拥塞避免阶段把拥塞窗口控制为按线性规律增长，使网络比较不容易出现拥塞。



***Chapter 7 Application Layer***

**概述**

**DNS域名系统**

**电子邮件**

**万维网**

**网络管理**

应用程序：互相通信的分布的进程

在网络主机上的用户空间运行

互相交换消息

比如email、ftp和web

应用层协议

应用程序的一部分

定义应用程序直接交换的信息以及相应的动作

利用底层协议提供的服务

应用层术语

一个**进程**是运行于主机上的一个程序。

在同一主机上的进程利用操作系统提供的IPC (interprocess communication）进行通信。

在不同主机上运行的进程利用应用层协议进行通信

**用户代理**（user agent）是指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

应用程序编程接口API （application programming interface）

定义应用程序和传输层之间的接口

socket: Internet API:两个进程通过向socket写数据和读数据来通信

Q: 进程如何指明它要与之通信的另一个进程

IP地址指明该进程所在的主机

端口号指明该主机应该把收到的数据交给哪个当地进程

应用程序需要什么样的传输服务

数据丢失

有的应用程序(比如音频)可以容忍一定程度的数据丢失

有的应用程序(如文件传输)要求100％的可靠传输

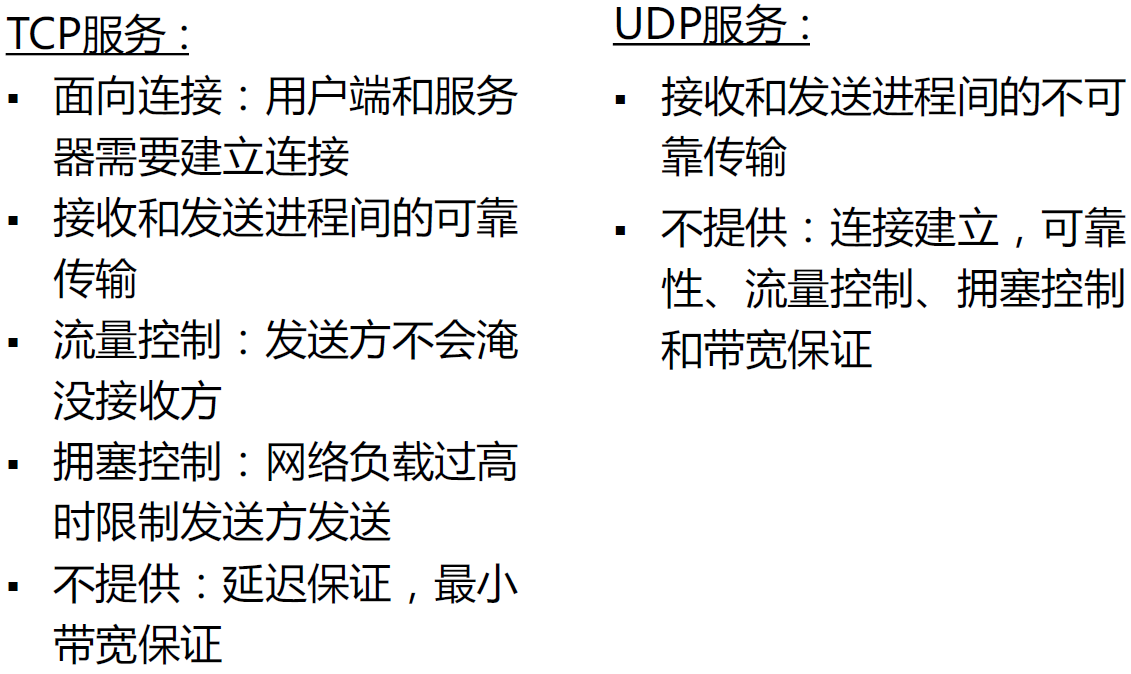
带宽

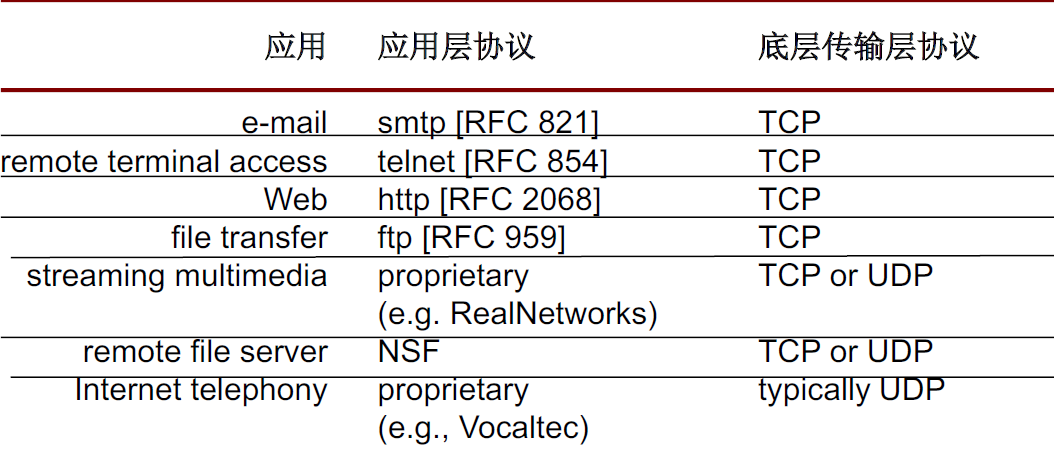
有的程序(如多媒体)需要一定的带宽才能工作

有的程序则使用它所能得到的全部带宽

延迟

有的程序要求低延迟，比如IP电话和交互游戏.





7.1 DNS域名系统

DNS名字空间

域名资源记录

域名服务器

DNS名字空间: 因特网名字与数字地址分配机构ICANN



资源记录: 资源记录格式

Domain\_name, Time\_to\_live, Class, Type, Value; 域名;记录所在域;保留时间; 类别;类型;值

7.2 电子邮件

体系结构和服务;用户代理;邮件格式(RFC 5322);邮件传送;最后传递

体系结构和服务

子系统：用户代理、传输代理

功能与协议：编写、显示、处理；简单邮件传输协议SMTP

其他：邮箱、收件人列表；信封、头部、主体

邮件传送

RFC 5322（822）头域

SMTP:简单邮件传输协议 ;端口25

最后的投递

POP3——邮局协议

IMAP——交互式电子邮件访问协议

Webmail

文件传输协议FTP

在两个主机之间传输文件

客户/服务器模式：由客户端发起文件传输(上传或下载)

ftp: RFC 959

ftp server: port 21

客户端连接到ftp服务器TCP的21号端口

建立两个并行的TCP连接:

- 控制: 在客户端和服务器之间交换命令、响应

- 数据: 传递文件数据

Ftp服务器维护状态：当前目录，身份认证

注意: 1. 数据连接可以双向使用.

2. 数据连接不必始终存在.

3. 控制连接采用的是telnet.

7.3 万维网

体系结构概述

静态Web页面

动态Web页面和Web应用

HTTP——超文本传输协议

移动Web

Web搜索

客户端

统一资源定位符URL: http://www.ruc.edu.cn:80/index.html;202.112.126.221

超文本传输协议HTTP

浏览器、插件、辅助应用程序

多用途网际邮件扩充协议MIME (Multipurpose Internet Mail Extensions)

服务器端

监听TCP 80端口

Web服务器:内存缓存技术;多线程模式

Cookie

使用:标识顾客;商品列表（购物车）;参数设置（Web门户网站）;服务器访问记数

静态Web文档

HTML——超文本标记语言

服务器端动态Web页面生成

公共网关接口CGI:包含表单的Web页面

超文本预处理器:PHP脚本

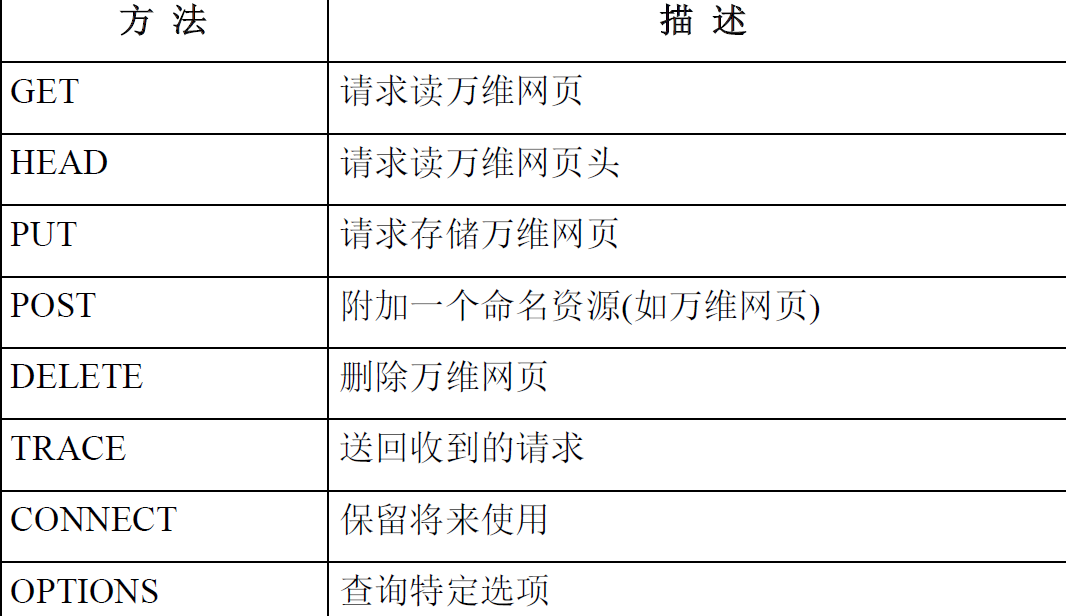
其他:JSP;ASP

动态HTML:JavaScript;VBScript

HTTP——超文本传输协议

发展趋势:HTTP像一个传输协议;SOAP：运行在HTTP之上的XML RPC

连接:HTTP1.1,持续连接a.多个连接b.一个持续连接c.流水线式请求



网络管理的基本出发点:

具有网络监视和控制两方面的能力

能够管理所有层次的协议是所有协议的集合

尽可能大的可管理范围（设备种类） 通过信息交换来实现的

尽可能小的系统开销

可管理不同厂家的设备

网络管理的五大功能

故障管理：故障检测、故障定位、故障报告

配置管理：识别网上的设备和用户，维护网上软硬件和电路的精确清单

性能管理：资源利用率分析

计费管理：对用户使用的各种资源进行跟踪，统计时间

安全管理：用户授权，控制网上用户只能访问自己权限内的资源

常用的术语

对象： 设备使用一个或多个变量描述自己的状态，这些变量称为对象（objects），通信和信息处理范畴内可标识的且拥有一定信息特性的资源，如：网络接口、主机、路由表、某种协议

被管理节点:可用管理协议进行管理和控制的对象，运行SNMP代理程序（SNMP agent），维护一个本地数据库，描述节点的状态和历史，并影响节点的运行。

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，维护管理信息库。

管理信息库MIB: 保存有关被管站点信息的数据结构，每个站点，所有的对象组成管理信息库MIB（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。

网络管理系统的组成

网管站：运行网络管理软件的计算机

网管代理（agent）：运行于被管设备

委托代理（proxy）：无法安装SNMP的设备，通过委托代理管理

管理信息库（MIB）：存放被管信息

管理协议：网管站、网管代理、委托代理之间的通信协议

网管站

运行特殊网络管理软件的普通计算机;

在它上面运行一个或多个进程

它们在网络上与代理通信：发送命令，接收应答

网管工作站对所有被管设备的管理，通常采用定时的询问机制

网管代理（agent）

运行于被管设备中，如路由器、交换机、网桥等

网络设备在完成主要职责的同时，运行一个网管代理软件，一般网管工作站定时轮询网管代理，网管代理则向网管工作站报告所询问的网络状态，或有紧急情况时网管代理向网管工作站提出紧急请求：Trap

委托代理（proxy）:

一个SNMP的委托代理负责对一个或多个被委托设备的管理功能，即把管理者的命令和请求转换为此类设备能识别的信息，同时把这些设备的信息转换为管理者所能识别的形式，进行间接管理

对象的命名

管理信息库MIB（management information base）指明了网络元素所维持的变量（即能被管理进程查询和设置的信息）

MIB给出了一个网络中所有被管对象的集合的数据结构，SNMP的管理信息库采用树型结构，称为对象命名树

SNMPv1协议只支持5类消息

get：管理者读取代理信息

get-request（询问）

get-response（得到回答）

get-next-request（继续询问）

set：管理者设置代理对象中的值:set-request

trap：在紧急情况下，代理向管理者报告信息

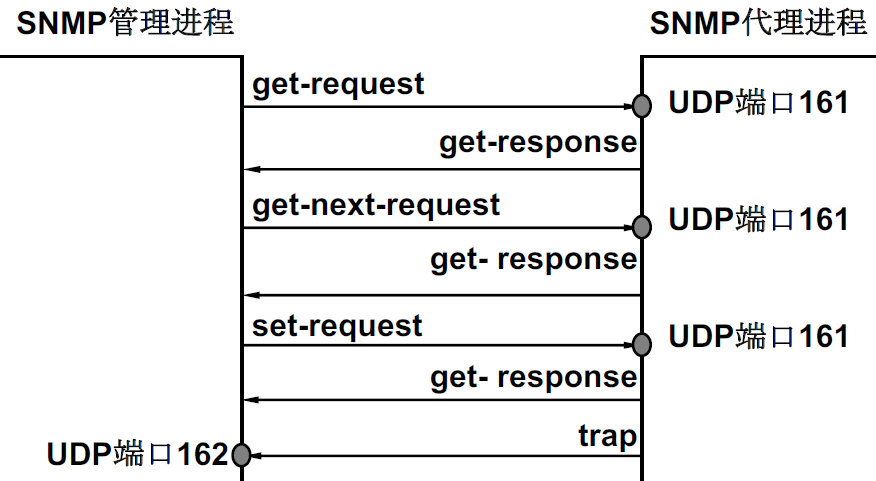
SNMP采用UDP协议

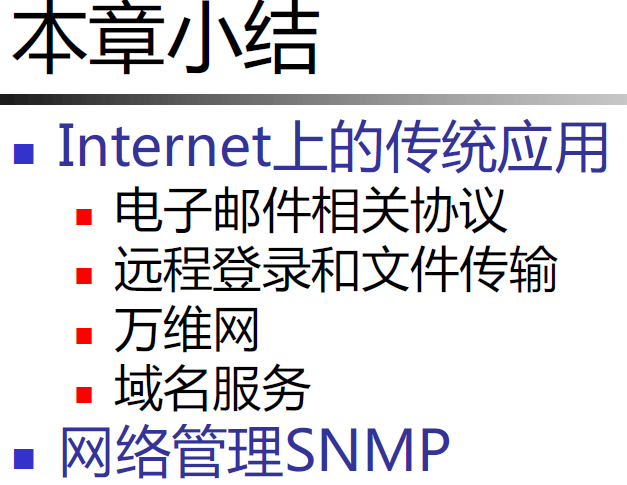
使用的端口号为161和162

161端口：接收来自管理者（网管工作站） 的全部协议数据单元（用在被管对象上）

162端口：用于代理向管理者传送事件/陷阱（用在网管站）

SNMP报文的典型操作过程





***Chapter 8 Network Security***

主要内容

8.1 网络安全概述

8.2 密码学

8.3 对称密钥算法

8.4 公开密钥算法

8.5 数字签名

8.6 公钥的管理

8.7 通信安全

8.8 计算机网络病毒

网络安全问题

保密—又称为机密，任务是确保信息不会被未授权的用户访问

认证—是指当你在展示敏感信息或者进入商务交易前你必须要确定在跟谁通话

不可否认—牵涉到签名，指你发出信息之后从技术上说不能够否认这条信息不是你发出的

完整性控制—确保信息的传输途中未遭到任何改变

**安全涉及协议栈的每一层**

物理层将传输线封装在内含高压气体的密封管线中可以对付搭线窃听

数据链路层采用点到点链路加密（link encryption, 缺点一是易受来自路由器内部的攻击;缺点二是必须对所有的应用实施，缺乏灵活性

网络层采用防火墙进行分组过滤:IPSec

传输层端到端的加密

应用层解决用户认证和不可否认性这样的问题

8.2 密码学

待加密的消息为明文（plaintext），它经过一个以密钥（key）为参数的函数变换，这个过程称为加密，输出的结果称为密文（ciphertext）

设计加密和解密方法的艺术称为密码编码学，破解密码的艺术称为密码分析学，两者统称为密码学

符号标记

明文P；密钥K；密文C

加密过程: C = Dk (P)

解密过程: P = Ek (C)

Kerckhoff原则：“让密码分析者知道加解密算法，把所有的秘密信息全放在密钥中”

密钥长度越长，破解难度越大！

密码分析方法

只有密文攻击:密码分析者只得到了一定量的密文，但是没有对应的明文

已知明文攻击:密码分析者有一些相匹配的密文和明文

选择明文攻击:密码分析者能够加密某些他自己选择的明文

密码必须能抵抗上述三种攻击！

置换密码（substitution cipher）

每个字母或者每一组字母被另一个字母或者另一组字母取代;不改变明文字母的顺序，但是把字母伪装起来

凯撒密码（Caesar cipher）:每一个字母被位于其后的第三个字母代替: 一个例子：attack -> dwwdfn

一个通用方案是允许字母表被移动k个字母而不是3个，此时k相当于字母表循环移动算法的密钥:优点：简单；缺点：极其容易破译

单字母表置换

让明文中的每一个字符都映射到其他一个字符上

例如abcdefghijklmnopqrstuvwxyz QWERTYUIOPASDFGHJKLZXCVBNM

则attack -> QZZQEA

密钥共有26！≈ 4′10^26种可能，穷举攻击计算上不可行

转置密码（transposition cipher）

重新对字母进行排序，但是不伪装明文

下图给出了一种常见的转置密码：柱形转置: MEGABUCK是密钥，作用是对矩阵列进行编号，例中第一列为A代表的那一列，以此类推;明文按水平方向的行依次填充矩阵;密文按列读出，列次序如密钥所规定

转置密码破译方法

观察英文常用字母出现的频率是否与明文的常规频率吻合，是的话则可以确定是转置密码

根据特定的环境信息猜测一个可能的单词或者短语，观察单词中两字母组合的规律来确定列数

当列数较小的时候，共有K(K-1)种可能的列对，对每一个列对进行检查看它的两字母组合的频率是否与英文两字母组合频率相匹配，依此逐步确认列的顺序。

一次一密（one-time pad）

选择一个随机的位串作为密钥，将明文转变成一个位串，然后逐位计算两个串的异或值作为密文

信息论表明，在没有密钥的情况下密文无法破解

缺点:密钥量大，无法记忆，必须携带书面副本;对于丢失字符和插入字符敏感，发送方和接收方必须保持同步

例：明文：I love you.； 密钥是Pad1，若选择Pad2用来解密，得到明文结

果Elvis lives，似是而非

量子密码学：BB84协议

8.2.5 两条密码学原则

原则1：被加密的消息必须包含一定的冗余度

解密一条消息之后，接收者必须能够通过简单的检查手段或者计算来判断消息是否有效

冗余信息对理解消息是不必要的，但是能够阻止主动入侵者发送垃圾信息，并欺骗接收者解密垃圾信息并在所谓的“明文”上执行相应的处理

缺憾是冗余信息使被动入侵者也更加容易破解系统

原则2：需要采取某种方法来对抗重放攻击

确保接收到的消息能被验证是新鲜的，是最近被发出来的，而不是重放的

一种措施是在每条消息中包括一个时间戳，以便表明消息的有效时间段

其他措施后面继续讨论

8.3 对称密码算法

现代密码学仍然使用置换和转置的思想，但是加密算法尽可能地错综复杂

对称密钥算法(symmetric-key algorism)使用同样的密钥完成加密和解密操作

块密码（block cipher）:接受一个n位的明文块作为输入，利用密钥变换成n

位的密文块

乘积密码（product cipher）:P盒完成转置，S盒完成置换;一系列S盒和P盒叠加起来构成乘积密码，现代加密算法大多是乘积密码

8.3.2 数据加密标准DES（Data Encryption Standard）

DES工作流程: 明文按64位数据块的单元被加密，生成64位密文；密钥56位

共19个步骤。第一步是一个与密钥无关的转置操作，直接作用在明文上；最后一步是这次转置的逆操作；倒数第二步是交换左32位和右32位；其他16步在功

能上完全相同，但是使用原始密钥的不同函数作为参数

用同样的密钥可以完成解密，但是解密步骤与加密步骤相反

一次迭代过程（一轮）

接受两个32位输入(Li-1, Ri-1)，产生两个32位输出(Li, Ri)

左边的输出Li 等于右边的输入Ri-1

右边的输出是左边输入Li-1,与一个函数值f(Ri-1, Ki)逐位异或的结果，该函数的输入参数是右边的输入Ri-1和本轮的密钥Ki

F函数

第一步：根据一个固定的转置和复制规则，将32位的Ri-1扩展成

一个48位的数字E

第二步：E和Ki被异或在一起

第三步：异或结果被分成8个6位组，每个6位组被输入到不同的S盒中。每个S盒将6位输入映射到一个4位输出

第四步：8个4位通过一个P盒，得到32位输出

轮密钥的产生:

首先从64位密钥K中提取出56位，并执行转置操作

每一次迭代之前，密钥被分成两个28位单元，每个单元循环左移位，左移位数取决于当前的迭代号

移位之后再执行另外一个转置操作，即得到Ki

三重DES

1979年初，IBM意识到DES的密钥长度太短，与是设计了三重DES来增加密钥长度

共三步：第一步按照常规的方式用密钥K1执行DES加密；在第二步中，DES以解密方式运行，并使用K2作为密钥；第三步再次用K1执行DES加密

EDE（加密-解密-加密）模式是为了与现有的DES系统保持兼容性，并且112位密钥已经足够长了

8.3.3 高级加密标准AES

对AES的基本要求是：比三重DES快、至少与三重DES一样安全、数据分组长度为128比特、密钥长度为128/192/256比特

**Rijndael**算法

在数学上，Rijndael算法是以迦罗瓦域理论为基础的，具有一定的可以证明的安全性

该算法采用的是代替/置换网络。每一轮由三层组成：第一层是线性混合层，确保多轮之上的高度扩散；第二层是非线性层，由16个S-盒并置而成，起到混淆的作用；第三层是密钥加层，子密钥简单的异或到中间状态上

该算法明文和密文128bit,密钥可为128bit, 192bit, 256bit。迭代轮数也是可变的,分别为10轮,12轮,14轮

所有的操作都以整个字节为单位，可以用软件和硬件高效率实现

算法结构:

State数组被初始化为明文数据，rk[0]-rk[10]是根据密钥经过复杂操作产生的11个与state同样长的数组，rk[1]-rk[10]分别是10轮的轮密钥

算法流程:

将明文复制到state数组中，前四个字节复制到第0列，接着四个字节复制到第1列，以此类推

state数组与rk[0]逐字节异或，结果保存在state中

进入主循环，一共有10轮迭代，每次迭代如下：

第1步：在state上执行逐字节替换，替换过程利用一个S盒进行，是直接的单字目表置换

第2步：state数组每一行循环左移，第j 行左移j 个字节

第3步：通过矩阵乘法混合state的每一列,新列是旧列与一个常量矩阵的乘积

第4步：将本轮密钥rk[i]与state异或，结果保存在state中

最后state数组即为密文

电子代码簿模式（ECB）

例如使用DES来加密一长段明文，将它分割成连续的8字节的数据块，然后用同样的密钥逐个加密这些数据块，这种加密方式称为ECB模式

由于各个数据块是独立加密的，因此可以把两块密文互换位置，仍能够正确地解密出明文，只不过明文中相应两个数据块被颠倒了位置。如果这个数据块恰好是两个人不同的奖金数目，则事情就弄得很糟

密码块链接模式（CBC）

每个明文块在加密之前，先与上一个密文块执行异或操作

第一个明文块与一个随机选择的IV（Initialization vector，初始向量）执行异或，该IV和密文一起传输

解密同样使用异或操作将过程翻转

特点是明文根据它出现的位置不同而生成不同的密文

密码反馈模式

CBC模式只能当整个64位数据块到来之后才能够加密，对于逐个字节加密的应用，可以采用密码反馈模式（cipher feedback mode）

下面以DES算法为例说明该模式的使用

加密过程:

图中加密机处于“0-9字节已经被加密发送出去”之后的状态

DES算法作用在64位移位寄存器上并产生一个64位密文，密文最左边的字节被提取出来与明文字节10异或，异或值被传送到输出线路上

异位寄存器左移8位，C2移出，C10移入最右边。本字节加密过程结束

明文加密结果依赖于明文之前的历史；需要初始向量来启动加密过程

解密过程:

与加密过程类似，需要注意移位寄存器上仍然要使用DES加密！

密码反馈模式的一个问题是，即使密文在传输过程中只有一位发生翻转，解密时该字节处于异位寄存器中时所波及的8个字节都被破坏。但是这种影响是局部的

流密码模式

用一个密钥加密一个初始向量得到一个输出块，用同一个密钥加密这个输出块得到第二个输出块，以此方式得到输出块的序列称为密钥流

把密钥流和明文逐位异或得到密文输出

初始向量和密文一起传输

解密过程类似

流密码模式的优缺点

可以将密钥流提前计算出来以提高速度

密文中的一位传输错误仅仅导致解密明文时一位发生错误，对传输错误不敏感

不要两次使用同样的（密钥，IV）对，这样会得到同样的密钥流，使密文容易受到密钥流重用攻击，拿同一密钥流加密的两个密文异或即可消去密钥的影响

计数器模式:

除了电子代码簿模式以外的其它模式都不能随机访问密文，必须解密前面的之后才能解密后面的内容，计数器模式能够解决这个问题

初始向量再加上一个常数被加密，结果得到的密文再与明文异或输出

同样存在密钥流重用攻击问题

一些常见的对称密钥算法



8.3.6 密码分析

差分密码分析

原理：首先用一对仅在少量的位上有差异的明文块来加密，然后仔细观察在加密过程中每一次迭代时所发生的情况

线性密码分析

将明文和密文中特定的位异或在一起，然后检查各种模式的结果。当重复这样的过程时，应该一半为0，一半为1。但是通常密码算法会在某一个方向上引入偏差，不管偏差多小，都可以用来降低工作量

电子功率分析

计算机用3v表示“1”，0v表示“0”，因此处理“1”要比“0”耗费更多的电能。检测电能的变化可以破译逐位加密的算法

时间分析

密码算法中充满对位测试的if、else语句，它们的执行时间不同。减慢时钟并观察每一步的时间，可以推测出轮密钥

8.4 公开密钥算法

在对称密钥系统中，分发密钥往往是最薄弱的环节

1976年，斯坦福大学的Diffie和Hellman提出了一种全新的密钥系统

加密密钥和解密密钥不同，从加密密钥不能轻易地推出解密密钥

加密算法E和解密算法D必须满足三个要求：

D(E(P)) = P

从E推出D极其困难

用选择明文攻击不可能破解E

具备了上述条件之后，加密密钥和加密算法都可以被公开

该系统命名为公开密钥密码系统（public-key cryptography）

公开密钥密码系统要求每个用户拥有两个密钥，一个是公开密钥（公钥），另一个是私有密钥（私钥）

其他人给某个用户发送消息的时候，用该用户的公钥加密

用户解密收到的消息的时候，使用私钥解密

RSA

RSA算法由MIT的Rivest、Shamir、Adleman发现

预计算:选择两个大的素数p和q（通常1024位）;计算n = p \* q和z = (p - 1)\*(q - 1);选择一个与z互素的数，称为d ;找到e，使其满足e \* d = 1 mod z

明文分块:将明文化分成k位的块，k是满足2^k < n的最大整数

加密:加密消息P，只需要计算C = P^e (mod n)即可

解密:解密密文C，只需要计算P = C^d (mod n)即可

公钥对是（e，n），私钥对是（d，n）

RSA算法的安全性建立在大数分解的难度基础上，如果能够分解大数n，则算法自动破解。但是数学家探索了大数分解300年也进展不大

算法的缺点是速度太慢，一般用来分发密钥，而不直接用来加密数据

其它公开密钥算法

背包算法（Merkle and Hellman, 1978）: 算法思想是，某人拥有大量的物品，每个物品的重量各不相同。为了编码一个消息，这个人秘密选择一组物品并将物品放在一个背包中。背包中物品的总重量公开，所有可能的物品也被公开列出，但是背包中物品的细则是保密的。“根据给定的总重量找出可能的物品明细列表”被认为是一个计算上不可行的问题

El Gamal算法: 建立在计算离散对数的困难度基础之上

椭圆曲线算法

8.5 数字签名

数字签名消息的三个要求

接收方可以验证发送方所宣称的身份

发送方以后不能否认该消息的内容

接收方不可能自己编造这样的消息

对称密钥签名系统存在的结构性问题是每个人都必须信任一个中心机构，但是现实中很难协调出一个各方都信任的权威机构

MD5:

MD5算法是Ronald Rivest设计的一系列消息摘要算法中的第5个

预处理过程: 将原始的明文消息填补到448位，然后追加一个64位整数来表示消息的原始长度，因而整个输入长度是512的倍数;将一个128位的缓冲区初始化为一个固定的值

计算过程:每一轮取出一个512字节的输入块，将它与128位的缓冲区彻底

地混合起来，这里使用了一个正弦函数表格;对每一个输入块执行4轮，直至所有的输入块都执行完毕，最后，128位缓冲区中的内容即是消息摘要

8.6 公钥的管理

X.509是ITU设计的专门针对证书格式的标准，其核心是一种描述证书的格式



问题由来

由一个CA来颁发全世界的证书是不切实际的，它将不堪重负而且成为单一失败点

一个解决方案是采用多个由同一组织运行的CA，它们用同一个私钥来签名证书 密钥泄漏问题！ ;如何选择这样一个世界公认的组织？

公开密钥基础设施PKI(Public Key Infrastructure)。PKI有多个部件，包括用户、CA、证书和目录

CA层次

最顶级的CA，即层次的根，它的责任是证明第二级的CA（称为RA，区域权威机构）

RA的责任是证明下一级CA的真实性，这些下级CA真正为组织和个人颁发证书

根CA授权一个新的RA时，它生成一个X.509证书，声明它批准了这个RA并在证书中包含该RA的公钥。根CA用私钥签名之后，将证书颁发给该RA

8.7 通信安全

安全标准IPSec(IP security，IP安全)

IPSec是一个多服务、多算法和多粒度的框架

多服务：保密性、数据完整性、重放攻击保护

多算法：与具体算法无关，支持多个算法

多粒度：既能保护单个TCP连接，又能保护一对主机或者一对安全路由器之间的所有流量

IPSec是面向连接的，一个连接称为一个SA(security association，安全关联)，是IPSec的基础和最重要的概念。SA是两个端点之间的单工连接，它定义了两端之间通信的安全措施，如是否采用加密和认证

IPSec框架

IPSec主要包括两部分，第一部分是两个新的头，AH(认证头)和ESP(安全封装净荷)；第二部分是ISAKMP(Internet安全关联和密钥管理协议)以及IKE(Internet密钥交换协议)，解决的是密钥管理和交换的问题

IPSec传输模式

用于两个主机之间的通信，两个通信端点同时都是加密端点

IPSec头插入到IP头和上层协议之间，只保护IP包中的上层协议部分

IPSec隧道模式

当一个加密端点是安全网关时，就必须使用隧道模式

对整个IP包进行封装和保护。其中：外部IP头是IPSec处理以后添加上去的新IP头，其目的地址指向IPSec隧道终点的安全网关，源地址是本安全网关。在这种保护模式中，通信终点是由受保护的内部IP头指定的地址，而通信数据受保护的终点是由外部IP头指定的地址。

认证头AH:

提供完整性检查和防重放攻击

在IPv4中，AH被放在IP头(包括所有可能选项)和TCP头之间；在IPv6中，AH只是一个扩展头

AH各字段的解释

下一个头：原始分组中的protocol域已经被替换成51代表后面紧跟一个AH头，AH中的下一个头域的值是传输层协议号，一般是6，代表TCP

安全参数索引：是连接标识符，用以指定接收方数据库中一条特定的记录，该记录包括了这个连接上所使用的共享密钥以及加密算法等其他参数

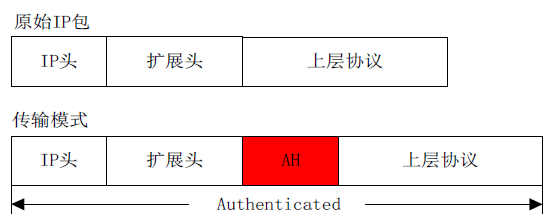
序列号：用来对一个SA上发送的所有分组进行编号，每个分组（包括重传分组）有唯一的序列号，检测重放攻击。若2^32个序列号全部用完，则SA必须重新协商，而不是回绕

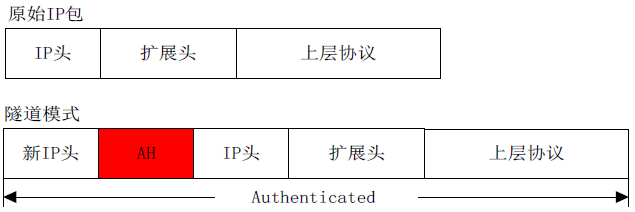
认证数据：可变长度，包含了净荷数据的数字签名。签名算法由SA建立时确定。一个简单方法是在分组和共享密钥串接起来的位串上计算散列值，称为HMAC(Hashed Message Authentication Code，散列消息验证码)

AH的两种模式:

AH的完整性检查覆盖了IP头部的一些不随路由器变化的字段，可以对IP头部做完整性控制

AH不支持数据加密，因此应用范围有限





安全封装净荷

ESP头由安全参数索引和序列号组成

ESP支持对IP头部之后的部分进行完整性检查，HMAC放在消息尾部，以便于一边发送消息一边计算HMAC

ESP同时支持加密和完整性控制，很有可能完全取代AH

8.7.2 防火墙

大多数公司有大量的保密信息放在LAN上，这些信息暴露给竞争对手将会带来可怕的后果

病毒、蠕虫和其他恶意代码泄漏进内部LAN之后也会对内部数据造成破坏

需要一些机制来管理和控制哪些信息能够出去，哪些信息能够进来—防火墙

典型的防火墙包括两个分组过滤路由器和一个应用网关。两个分组过滤路由器工作在网络层，分别负责监控流入流量和流出流量；应用层网关工作在应用层，负责监控具体的应用协议，比如电子邮件

分组过滤

一般通过ACL（Access Control List，访问控制列表）进行

ACL的每一个表项称为一条过滤规则，一般包括（源IP地址，目的IP地址，源端口，目的端口，上层协议）五元组以及一个处理动作。

对于到达的报文，首先检查该五元组，若有匹配表项，则按该表项的动作处理，若没有匹配项，则按照默认的处理规则处理

过滤规则目前是由网络管理员手工添加的

应用网关

是建立在应用层上的协议过滤机制，针对特别的网络应用服务协议的数据进行过滤

针对每个不同的应用使用不同的处理方法

优点是易于记录并控制所有的进出通信，并对Internet的访问做到内容级的过滤，控制灵活而全面，安全性高。具有日志、统计、审计和用户认证功能

缺点是需要针对每一种应用编写不同的代码，维护比较困难，速度比较慢

防火墙是不是万能的？

利用IP地址欺骗技术可以绕过防火墙

防火墙允许穿过它建立连接，外部攻击者可以向内部服务器发送建立TCP连接请求，服务器以为是正常请求于是为该请求建立一个状态并分配一定的资源。如果攻击者同时发送大量的连接请求，就会耗尽服务器的资源而无法继续提供服务，致使合法用户也不能享用服务，这种攻击称为拒绝服务攻击（Denial ofService，DoS）。更糟糕的是攻击者可以利用系统漏洞控制成千上万台合法主机来进行DoS攻击，这种攻击危害力大而且难以预防，称为分布式拒绝服务攻击（DDoS）

防火墙不是万能的！

产生计算机病毒的原因，大致可以分为以下几种：

计算机爱好者出于好奇或兴趣，也有的是为了满足自己的表现欲或制作恶作剧

产生于个别人的报复、破坏心理

来源于游戏软件

来源于软件加密

用于研究或实验而设计的“有用”的程序

出于政治、经济和军事等特殊目的，一些组织或个人也会编写一些程序用于进攻对方的计算机

狭义定义:

1994年2月18日，我国正式颁布实施了《中华人民共和国计算机信息系统安全保护条例》，在《条例》第二十八条中明确指出：“计算机病毒是指编制或者在计算机程序中插入的破坏计算机功能或者破坏数据，影响计算机使用并且能够自我复制的一组计算机指令或者程序代码”

广义定义:

计算机病毒(Computer Virus)，是一种具有自我复制能力的计算机程序，它不仅能破坏计算机系统，而且还能传播、感染到其他的系统，能影响计算机软件、硬件的正常运行，破坏数据的正确与完整。

类似于生物病毒，它能把自身附着在各种类型的文件上或寄生在存储媒介中，能对计算机系统和网络进行各种破坏；

有独特的复制能力和传染性，能够自我复制——主动传染，另一方面，当文件被复制或在网络中从一个用户传送到另一个用户时——被动传染，它们就随同文件一起蔓延开来

病毒有以下几个主要特点。

1. 传染性(基本特征)2. 夺取系统控制权3. 隐蔽性4. 破坏性5. 潜伏性6. 可触发性7. 不可预见性

有了条件病毒是如何传染的呢？

（1）驻入内存：病毒要达到传染的目的必须驻留于内存之中，所以病毒传染的第一步是驻留内存；

（2）寻找传染机会：病毒驻入内存之后，首先寻找可进行攻击的对象，并判定这一对象是否可被传染（有些病毒的传染是无条件的）；

（3）进行传染：当病毒寻找到传染的对象并判定可进行传染之后，通过INT 13H这一磁盘中断服务程序达到传染磁盘的目的，并将其写入磁盘系统；

分类：

按破坏性分类

良性病毒：是指那些只是为了表现自己而并不破坏系统数据，只占用系统CPU资源或干扰系统工作的一类计算机病毒。

恶性病毒：是指病毒制造者在主观上故意要对被感染的计算机实施破坏，这类病毒一旦发作就破坏系统的数据、删除文件、加密磁盘或格式化操作系统盘，使系统处于瘫痪状态。

按传染方式分类

系统引导型：感染引导区，系统引导时病毒装入内存，同时获得对系统的控制权，对外传播病毒，并且在一定条件下发作，实施破坏。

文件型病毒：一般只传染磁盘上的可执行文件(COM，EXE)。将自身包围在系统可执行文件的周围、对原文件不作修改，运行可执行文件时，病毒程序首先被执行，进入到系统中获得对系统的控制权。

混合型病毒：兼有以上两种病毒的特点，既染引导区又染文件，因此扩大了这种病毒的传染途径。

按连接方式分类

源码型病毒：源码病毒较为少见，亦难以编写。因为它要攻击高级语言编写的源程序，在源程序编译之前插入其中，并随源程序一起编译、连接成可执行文件。此时刚刚生成的可执行文件便已经带毒了。在源被编译之前，插入到源程序中，经编译之后，成为合法程序的一部分。

入侵型病毒：将自身入侵到现有程序之中，使其变成合法程序的一部分。

操作系统病毒：可用其自身部分加入或替代操作系统的部分功能。因其直接感染操作系统，这类病毒的危害性也较大。

外壳病毒：将自身附在正常程序的开头或结尾，相当于给正常程序加了个外壳。大部份的文件型病毒都属于这一类。

按广义病毒概念分类

蠕虫(worm)：监测IP地址，网络传播

逻辑炸弹(logic bomb)：条件触发，定时器

特洛伊木马(Trojan Horse)： 隐含在合法程序上的一段非法程序。

陷门：在某个系统或者某个文件中设置机关，使得当提供特定的输入数据时，允许违反安全策略。

病毒、蠕虫和木马的区别

1．病毒的特点

计算机病毒是编写的一段程序，它可以在未经用户许可，甚至在用户不知情的情况下改变计算机的运行方式。病毒必须满足两个条件：

自行执行：它通常将自己的代码置于另一个程序的执行路径中。

自我复制：病毒代码的明确目的是自我复制。如：可以用受毒感染的文件副本替换其他可执行文件。

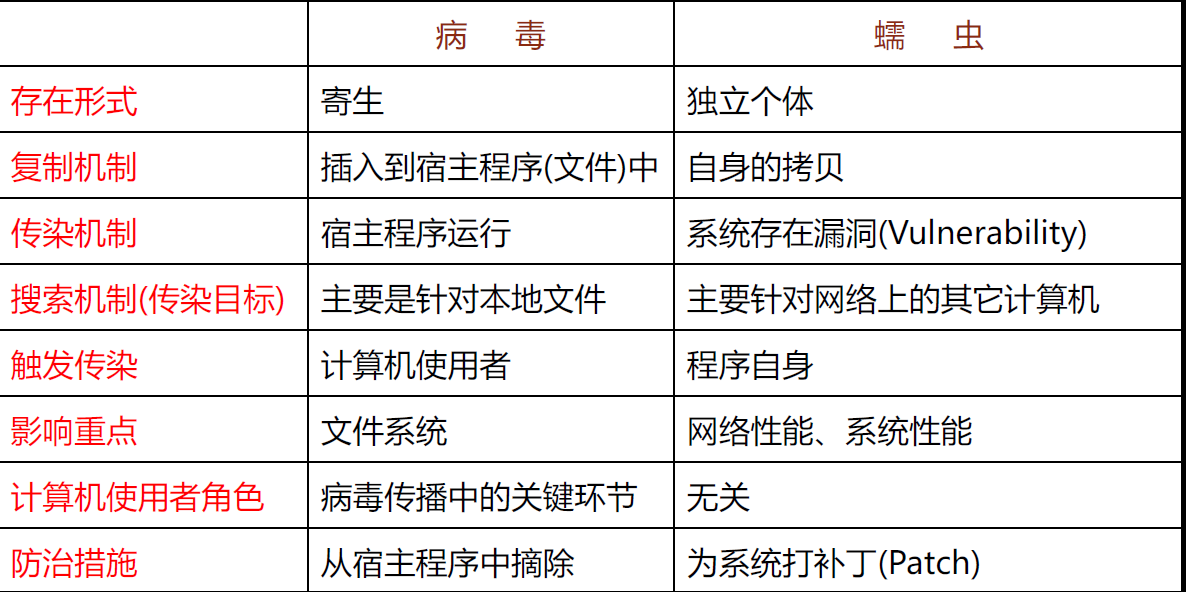
与蠕虫相比，病毒可破坏计算机硬件、软件和数据。

蠕虫的特点

蠕虫属于计算机病毒的子类，所以也称为“蠕虫病毒”。通常，蠕虫的传播无需人为干预，并可通过网络进行自我复制，在复制过程中可能有改动。与病毒

相比，蠕虫可消耗内存或网络带宽，并导致计算机停止响应。与病毒类似，蠕虫也在计算机与计算机之间自我复制，但蠕虫可自动完成复制过程，因为它接管了计算机中传输文件或信息的功能。

蠕虫与病毒之间的区别及联系



特洛伊木马

特洛伊木马(Trojan Horse)，简称木马，是一种恶意程序，是一种基于远程控制的黑客工具，一旦侵入用户的计算机，就悄悄地在宿主计算机上运行，在用户毫无察觉的情况下，让攻击者获得远程访问和控制系统的权限，进而在用户的计算机中修改文件、修改注册表、控制鼠标、监视/控制键盘，或窃取用户信息

古希腊特洛伊之战中利用木马攻陷特洛伊城；现代网络攻击者利用木马，采用伪装、欺骗(哄骗，Spoofing)等手段进入被攻击的计算机系统中，窃取信息，实施远程监控

木马的特点

木马(trojan)与病毒的重大区别是木马并不像病毒那样复制自身。

.木马的特点：

不感染其他的文件

不破坏计算机系统

不进行自我复制

木马的主要作用：作为远程控制、窃取密码的工具，它是一个具有内外连接功能的后门程序。

木马的两种主要传播途径：电子邮件和文件下载。

木马的组成

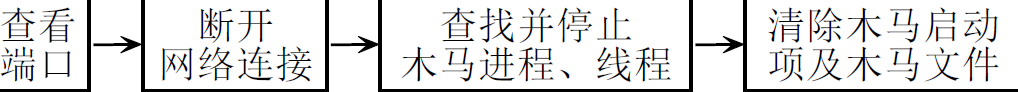
一般的木马程序包括客户端和服务器端两个程序，其中：

客户端：用于攻击者远程控制植入木马的计算机（即服务器端）

服务器端：是植入木马程序的远程计算机。

当木马程序或带有木马的其他程序执行后，木马首先会在系统中潜伏下来，并修改系统的配置参数，每次启动系统时都能够实现木马程序的自动加载。

检测和清除木马的方法;



特洛伊木马入侵的一个明显证据是受害计算机上意外地打开了某个端口

恶意程序:

未经授权便干扰或破坏计算系统/网络的程序或代码称之为恶意程序/恶意代码

恶意程序大致可以分为两类：

依赖于主机程序的恶意程序: 不能独立于应用程序或系统程序，即存在宿主

独立于主机程序的恶意程序: 能在操作系统上运行的、独立的程序

