1.1实体：任何可以发送或接受信息的硬件或软件进程。

1.1 什么是因特网

1.1.1 构成描述

因特网：一个世界范围的计算机网络。

主机/端系统：与因特网相连的设备（比如手机、平板、电脑等）。

端系统通过通信链路和分组交换机被连接到一起。端系统发送的分组在从发送端到接收端所历经的一系列通信链路和分组交换机被称为路径。

分组交换机(两种类型)：① 路由器：② 链路层交换机：

ISP：（Internet Service Provider）因特网服务提供商。

协议：被用来控制因特网中信息的接受和发送。

TCP传输控制协议 IP网际协议是因特网最最要的协议

1.1.3 协议

协议：定义了两个或多个通信实体之间交换的报文的格式和顺序，以及报文发送/接收一条报文或其他事件所采取的动作。

协议三要素：

- 语法：定义交换信息的格式

- 语义：定义收发双方需要完成的操作

- 同步：定义收发双方的时序关系

实体：任何可以发送或接受信息的硬件或软件进程。

1.2 网络边缘

网络的结构= 网络边缘(应用，主机)

+网络核心(路由器)

+接入网络与通信链路

网络边缘: 运行应用的端系统(端系统中的应用交互方式)

C/S 模式，特点

P2P模式，特点

利用网络的服务:

面向连接的服务

无连接的服务

1.3 网络核心

网络核心

组成:网络交换设备如:路由器+链路

功能:数据交换

数据交换方式 及比较

分组交换: 存储转发方式，统计复用

VC（虚电路）在网络层建立主机间的连接

Datagram（数据报）网络交换节点不需要维护主机间通信

线路交换

FDM

TDM

WDM

1.4 分组交换网中的时延、丢包和吞吐量

1.4.1 分组交换网中的时延和概述

节点总时延dnodal=节点处理时延dproc+排队时延dqueue+传输时延dtrans+传播时延dprop

* 处理时延dproc：路由器检查分组首部和决定将该分组导向何处所需要的时间。
* 排队时延dqueue：分组在经过网络传输时，要经过许多的路由器。所以分组在进入路由器时要在输入队列中排队等待处理；在路由器确定了转发接口后，还要在输出队列中排队等待转发。
* 传输时延dtrans：是主机或路由器发送数据帧所需要的时间，也就是从发送数据帧的第一个比特算起，到该帧的最后一个比特发送完毕所需的时间——数据帧长度(b) / 信道带宽(b/s)
* 传播时延dprop：两个路由器之间的分组运输的耗时。该传播速度取决于链路的物理媒介（光纤、双绞铜线等）。

1.4.2 排队时延和丢包

排队时延dqueue 是最复杂最有趣的，因为它对不同的分组不同。

流量强度：La/R（原则该比值≤1）

L：分组比特（ithink：就是长度）

a：分组到达队列的平均速率

R：路由器传输速率（带宽）

丢包：分组丢失。丢失比例会随流量强度增加而增加。

1.4.3 端到端时延

一般求端到端时延会假设没有排队时延。

（N为链路数，路由器有N-1台）

端到端的时延=发送主机的（传输时延）+传播时延+（N-1）台路由器的（处理时延+传输时延+到下一台设备的传播时延）+接收主机的（处理时延）

端到端的时延=N处理时延+N传输时延+N\*传播时延



1.4.4 计算机网络中的吞吐量

带宽是理想，吞吐量是现实。

两个主机之间的链路的传输速率（带宽）可能不尽相同，但两个主机的文件传输吞吐量一定=min{R1，R2,……RN}

1.5.1 分层的体系结构

目前用于教学需要掌握的为5层结构（自顶向下）： P35直接看书吧

* 应用层：网络应用程序及它们的应用层协议存留的地方，也就是这一层会通过流量和能与用户交互的程序来解决进程间的网络应用。
* 运输层：解决进程的通信问题（比如传输出错的应急情况）。有两种协议——TCP和UDP。

（写的不好，直接看书， 因特网的运输层在应用程序端点之间传送应用层报文，把运输层的分组称为报文段

* 网络层：标识各个网络与主机（比如IP编址；解决路由器如何转发分组进行路由选择的问题）

（no 网路层负责把称为数据报的网络层分组从一台主机移动到另一台主机上）

* 数据链路层：解决分组在链路上传输的问题（比如从信号标识的比特流中区分数据与地址；协调主机对总线的征用）
* 物理层：与实际传输媒体相关，可以决定用哪种信号来传输比特（比如电压、接口标准这些）。

而法律上的国际标准OSI体系结构是7层，在运输层和应用层之间增加了2个，其顺序为：物理层、数据链路层层、网络层、运输层、会话层、表示层、应用层。

1.5.2 封装

其实就是计算机网络的体系结构中，每一层向上一层提供服务时都会加一些附属信息，那肯定就要做好必要的打包工作，这就是封装。

2.1.1 网络应用程序体系结构

客户-服务器结构P56 书上是三点 需要看书

1. 客户和服务器是指通信中所涉及的两个应用进程

2. 客户是服务请求方，服务器是服务的提供方

3. 服务器总是处于运行状态，并等待客户的服务请求

4. 服务集中型（应用服务主要集中在网络中比客户计算机少得多的服务器计算机上）

对等（Peer-to-Peer，P2P）方式

1. 没有固定的服务请求者和服务提供者

2. 服务分散型

3. 突出特性：自扩展性（系统性能不会因规模的增大而降低）

4. 成本更低

### 2.1.2 进程通信

分三步：

* 标识必要的客户和服务器进程，发起通信的进程为客户，在会话开始时等待联系的进程是服务器
* 进程与计算机网络之间接口（通过套接字实现）

套接字：向网络发送报文和从网络接收报文

定义：套接字是应用程序进程和运输层协议之间的接口。

* 进程寻址（主机由IP地址标识，IP地址是一个32比特能够唯一标识主机的量）

2.1.3 可供应用程序使用的运输服务

套接字是应用程序进程和运输层协议之间的接口。

TCP用主机的IP地址加上主机上的端口号作为TCP连接的端点，这种端点就叫做套接字（socket）或插口。

套接字用（IP地址：端口号）表示。

而在开发一个应用程序时，还需要选择一种可用的运输层协议，而运输协议主要提供以下四个方面的服务（选择时可以从这些方面进行考量）：

1. 可靠数据传输

2. 吞吐量

3. 定时（有些数据交付会有时延限制的需求）

4. 安全性

补充：

端口号的范围是从1～65535。其中1～1024是被RFC 3232规定好了的，被称作“众所周知的端口”(Well Known Ports)；从1025～65535的端口被称为动态端口（Dynamic Ports），可用来建立与其它主机的会话，也可由用户自定义用途。

一些常见的端口号及其用途如下：

TCP 21端口：FTP 文件传输服务

TCP 23端口：TELNET 终端仿真服务

TCP 25端口：SMTP 简单邮件传输服务

UDP 53端口：DNS 域名解析服务

TCP 80端口：HTTP 超文本传输服务

TCP 110端口：POP3 “邮局协议版本3”使用的端口

TCP 443端口：HTTPS 加密的超文本传输服务

TCP 1521端口：Oracle数据库服务

TCP 1863端口：MSN Messenger的文件传输功能所使用的端口

TCP 3389端口：Microsoft RDP 微软远程桌面使用的端口

TCP 5631端口：Symantec pcAnywhere 远程控制数据传输时使用的端口

UDP 5632端口：Symantec pcAnywhere 主控端扫描被控端时使用的端口

TCP 5000端口：MS SQL Server使用的端口

UDP 8000端口：腾讯QQ

2.1.4 因特网提供的运输服务

因特网为应用程序提供了两个运输层协议：

①TCP

TCP服务模型包括 面向连接服务、可靠数据传送服务。P61

- 面向连接服务：在报文流动之前，TCP让客户和服务器互相交换运输层的控制信息

（这个过程也叫做三次握手）。

- 这种连接是全双工的

即连接双方的进程可以在此连接上同时进行报文的收发。

- 该连接结束必须拆除。

- 可靠的数据传送服务：没有字节的丢失和冗余。

- TCP有拥塞控制机制，它能抑制发送进程。

②UDP

UDP服务模型仅提供最小服务、无连接没有握手过程、是不可靠数据传送服务。

- 没有拥塞控制机制

* 因特网电话与udp （唯一）
* Dns（域名系统）与udp（唯一）

2.1.5 应用层协议

应用层协议定义了运行在不同端系统上的应用程序进程如何相互传递报文

协议内容）： （ 协议就是语法语义同步P5

* 交换的报文类型
* 报文类型的语法
* 字段的语义
* 发送报文的时序和响应规则

2.2 Web和HTTP

Web是因特网应用。

2.2.1 HTTP概况

Web使用的应用层协议是超文本传输协议（HyperText Transfer Protocal，HTTP）

HTTP由两个程序实现：一个客户程序+一个服务器程序；

HTTP定义了Web客户向Web服务器请求Web页面的方式，以及服务器传回Web页面的方式；

HTTP使用Tcp协议作为它的支撑运输协议；

HTTP是一个无状态协议，因为HTTP服务器不保存客户的任何信息

它使用客户-服务器的应用体系结构，Web服务器总是打开，有固定的IP地址。

2.2.2 非持续性和持续连接

往返时间（Round-Trip Time，RTT）指一个短分组从客户到服务器然后再返回客户所花费的时间。

非持续连接：

当客户-服务器的交互经Tcp进行，每个请求/响应都经过各自的一个单独的Tcp连接发送。

总的响应时间=2\*RTT + 服务器传输HTML文件的时间

因为非持续连接每次首先要完成Tcp的标志性三次握手动作：

①客户发起Tcp连接的一个小报文段

②服务器接收到后返还一个确认和响应

③客户向服务器再返回确认，表示自己状态ok

④服务器接收到确认，开始传输文件

⑤文件传输到结束，连接中断

①②和③④各是一次完整的RTT，所以总的响应时间是（2\*RTT+文件传输时间）

以上我们也能看出非持续连接的缺点:

- 每一个请求的对象都要建立和维护一个全新的连接

- 每个连接都要经受两倍RTT的交付时延（一个创建TCP，一个用于请求和接受对象）

持续连接：所有请求/响应经过同一个Tcp连接发送。

一般来说，如果一条连接经过一定的时间间隔未被使用，HTTP就关闭该连接。

实际上，HTTP的默认模式就是带流水线的持续连接。

2.2.3 HTTP报文格式

HTTP有两种报文

①请求报文

第一行（请求行）：

方法字段/URL字段/HTTP版本字段

补充说明 方法字段的取值：GET POST HEAD PUT DELETE

GET和POST有实体的区别：

使用GET时实体为空

使用POST时实体体中包含用户 在表单字段中的输入值；

HEAD用于调试跟踪，因为HEAD请求会使服务器不返回响应对象；

PUT常与Web发行工具连用，允许用户上传资料到指定的Web路径中；

DELETE允许用户删除Web服务器上的对象。

首部行：

Host ：指明对象所在的主机

Connection ：值为close时该行告诉浏览器不要麻烦地使用持续连接

并要求服务器在发送完被请求的对象后就关闭该连接

User-agent ：用来指明用户代理（向服务器发送请求的浏览器的类型）

Accept-language：表示用户想得到该对象的语法版本

②响应报文

初始状态行：

协议版本字段/状态码和相应的状态信息

状态码相关的短语：

- 200 ok ：请求成功

- 301 Moved Permanently：对象被转移了

- 400 Bad Request：一个通用的差错代码

- 404 Not Found：请求对象不在服务器上

- 505 HTTP Version Not Supported：不支持该版本协议

首部行（有6个）：

Connection ：值为close时该行告诉客户，发送完报文后关闭Tcp连接

Date ：指示服务器产生并发送该响应报文的日期和时间

Server ：指示该报文由xxx服务器产生

Last-Modified ：指示了对象创建或者最后修改的日期和时间

Content-Length：指示了被发送对象中的字节数

Content-Type ：指示对象的文本类型

实体体（为主要部分）

2.2.4 用户与服务器的交互：cookie

上一节提到HTTP往往无状态，但服务器希望标识用户，所以HTTP使用cookie使得站点可以跟踪用户。

cookie有4个组件：

* 在HTTP请求报文的首部行加一个cookie
* 在HTTP响应报文的首部行加一个cookie首部行
* 在用户端系统中保留一个cookie文件，由用户的浏览器进行管理
* 位于Web站点的一个后端数据库

2.2.5 Web缓存

Web缓存也叫代理服务器，它是能够代表初始Web服务器来满足HTTP请求的网络实体。它会保存最近请求过的对象的副本。

（其实Web缓存就像cache，当我的客户端（cpu）想要从服务器（内存）请求对象数据时，先看Web缓存（cache）是否命中，若命中，直接把数据返回，不命中，再从服务器（内存）求取对象存到web缓存（cache）中）

要是客户端请求的对象在Web中有副本，那么它就直接给客户返回响应报文

如果没有，Web就打开一个与目的服务器的Tcp连接

（Web发送请求，服务器收到回应

Web收到后存储一个副本，并将该副本用响应报文返回给客户）

Web缓存通常由ISP购买并安装

Web存在的原因：

- 可以大大减少对客户请求响应的时间

- 可以减少接入链路到因特网的通信量，减少成本

- web缓存器能从整体上大大减低因特网上的web流量，从而改善所有应用的性能

2.2.6 条件GET方法

web缓存可以给服务器发送一份请求报文，其中使用GET方法，在对服务器的请求报文中包含一个“If-Modified-Since：”的首部行，来查询某个对象在服务器是否被更改过，以此来向客户端证明Web缓存中存储的对象是否是最新的对象。

2.3 因特网中的电子邮件

用户代理

邮件服务器

简单邮件传输协议（Simple Mail Transfer Protocol，SMTP）

2.3.1 SMTP

SMTP是因特网电子邮件中的主要应用层协议，使用Tcp可靠传输服务。

它的报文内容中，以C：开头的是客户交给TCP套接字的那些行； S：开头的则是服务器发送给TCP套接字的那些行。

2.3.2 SMTP与HTTP的对比

* HTTP是一个拉协议，当某些人在Web服务器上装载信息，用户使用该协议从服务器拉取获得想要的信息，

TCP连接是由想要接收文件的机器发起的。

* SMTP是一个推协议，发送邮件方的服务器把信息推给接受方的服务器，这个TCP连接由想要发送文件的机器发起。

2.3.3 邮件报文格式

一个典型的报文首部包含：

- From：xxxxx

- To：xxxxx

- Subject：可以为空

2.3.4 邮件访问协议

因为SMTP是一个推的协议，假如收信方想从其他媒介查看通信内容，就需要从当前用户代理的浏览器去收到信件的邮件服务器里拉取邮件，这显然SMTP不再适用，所以新增了特殊的邮件访问协议解决这个问题：

* POP3(Post Office Protocol——Version3,POP3)第三版的邮局协议
* IMAP(Internet Mail Access Protocol,IMAP)因特网邮件访问协议：比POP3更强，用户可以创建远程文件夹并为报文指派文件夹
* HTTP

2.4 DNS：因特网的目录服务

识别主机主要有两种方式，通过主机名或IP地址。实际运用中，主机名往往不定长不完整，IP地址定长并且固定字段标识不同含义，所以我们希望得到一个主机名时能获得它的IP地址以得到主机更完整的信息。

2.4.1 DNS提供的服务

主要有三种重要服务：

- 主机别名（或者说因为有了DNS，一个主机才有规范主机名和别名的说法）

- 邮件服务器别名

- 负载分配

而我们需要一种能进行主机名到IP地址的转换的目录服务——Domain Name System,DNS

DNS：

- 本身是一个应用层协议

- 是一个由分层的DNS服务器实现的分布式数据库

- 一个使得主机能够查询分布式数据库的应用层协议

- 运行在UDP运输协议上，使用53端口

- 可以将用户提供的主机名解析为IP地址

2.4.2 DNS工作机理概述

核心为：分布式、层次数据库 + DNS缓存

为了处理扩展性问题，dns以层次方式组织，大致有三层：

- 根DNS服务器：提供TLD服务器的IP地址，根管顶级

- 顶级域DNS服务器（Top-Level-Domain,TLD）：提供权威DNS服务器的IP地址

顶级管权威

- 权威DNS服务器

除此之外还有一个本地DNS服务器

- 但它不属于层次结构它起代理作用

- 可以避免太多主机查询的通信和根服务器连接

- 请求主机需要先跟本地DNS获取，跟Web缓存的设置目的差不多。

2.4.3 DNS记录和报文

DNS分布式数据库的所有DNS服务器存储了资源记录（Resource Record，RR），RR提供了主机名到IP地址的映射。

资源记录格式：{Name,Value,Type,TTL}

TTL是该记录的生存时间

Name和Value的值取决于Type

Type=A，Name是主机名，Value是该主机名对应的IP地址

Type=NS，Name是域，Value是个知道如何获得该域中主机IP地址的权威DNS服务器的主机名

Type=CNAME，则Value是别名为Name的主机对应的规范主机名

Type=MX，Value是个别名为Name的邮件服务器的规范主机名

采访

第三章：运输层

运输层为运行在不同主机上的应用进程提供直接的通信服务起着至关重要的作用。

3.1 概述和运输层服务

运输层协议为运行在不同主机上的应用进程之间提供了逻辑通信功能。（意思就是感觉上让两个有通信需求的进程的主机直接相连了一样）

（可以对比一下：网络层提供了主机之间的逻辑通信）

3.1.1 运输层和网络层的关系

书上的例子很好，我简要描述一下放在这里

有两个家庭，每个家庭12个孩子，每周每个家庭中的小孩都要给对方家庭的小孩写信。

A家庭中的a1负责去邮局取回B家庭寄来所有的信、并寄去自家的信

B家庭中的b1负责去邮局取回A家庭寄来所有的信、并寄去自家的信

应用层报文 = 信封上的字符内容

进程 = 两个家庭里的小孩（它们是活动具体运行的单位）

主机（端系统） = 家庭（所以每个主机可以有好多进程）

运输层协议 = a1 和 b1

网络层协议 = 邮局提供的服务

①同时我们也可以注意到运输层协议的一个特性：只工作在固定的端系统中

（可以看到a1、b1肯定只为自己家服务，不会去给别人家送信）

并且我们假设一下，如果a1、b1生病了，这肯定要换人，假设换了a2、b2（他俩年纪更小）

那a2和b2完成的工作量肯定比不上a1、b1（比如a2、b2因为年纪小，拿信不专心造成丢失）

②所以计算机网络可以安排多种运输层协议，它们为应用程序提供的服务不尽相同

另外，a2和b2是受制于邮局的，他们不能想什么时候去拿就什么时候去拿，总得邮局开门吧

③所以运输协议能够提供的服务受制于底层网络层协议的服务模型

3.1.2 因特网运输层概述

这里简要描述一下网络层使用的协议：IP，它的服务模型是尽力而为交付服务——也就是它会尽力做到主机之间报文段的交付，但不保证交付成功、不保证报文段顺序正确、不保证数据完整，因此IP也称为不可靠服务。

所以运输层的协议（TCP或UDP）的最基本的任务就是：将两个主机间IP的交付服务扩展为运行在主机上的两个进程间的交付服务。它俩通过差错检查对报文段首部进行完整性检查。

TCP（传输控制协议）：它为调用它的应用程序提供了可靠的、面向连接的服务。

TCP：

- 提供进程交付和差错检查

- 提供可靠传输：确保正确地、按序地把数据从发送进程交给接收进程

- 拥塞控制

UDP（用户数据报协议）：它为调用它的应用程序提供了一种不可靠、无连接的服务。

UDP：

- 只提供进程交付和差错检查，所以UDP也不可靠

以下规范术语：

将TCP和UDP传输的分组称为报文段。

将网络层的分组称为数据报

3.2 多路复用域多路分解

实质是将由网络层提供的主机到主机交付服务延伸到为运行在主机上的应用程序提供进程到进程的交付服务。

多路分解：将运输层报文段中的数据交付到正确的套接字的工作。

多路复用：在源主机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（这将在以后用于分解）从而生成报文段，然后将报文段传递给网络层。

比如前面举的例子中

a1将取回的所有信件按照收信人的姓名交给自己对应的兄弟姐妹，这个过程就是多路分解；

a1将兄弟姐妹写好的信一起打包装好交去邮局，这个就是多路复用。

所以在运输层中，主机上的每个套接字能够分配一个端口号，当报文段到达主机，运输层检查报文段中的目的端口号，要是符合，就把它丢给对应的套接字，然后报文段中的数据通过套接字进入所连接的进程。（主机能够运行多个进程，每个进程都有独属自己的套接字和端口号，有各人回各家的感觉）

UDP套接字=目的IP地址+目的端口号

TCP套接字=源IP地址+源端口号+目的IP地址+目的端口号

3.3 无连接运输：UDP

UDP只做了运输协议能够做的最少工作，除了分解和复用它其实跟IP差不多，使用UDP差不多就是让应用程序直接跟IP打交道，UDP只是完成从应用程序拿到数据，附加上需要的源和目的端口号字段以形成报文段的工作。这一过程发送方与接收方的运输层实体并没有握手，所以UDP叫做无连接。

但UDP存在也有很多优点

- 关于发送什么数据以及何时发送的，应用层能控制更为精细

- 无需建立连接，意味着开销和延时小

- 无连接状态，能支持更多的活跃用户

- 分组首部开销小

3.3.1 UDP报文段结构

|<----- 32bit ----->|

源端口号 目的端口号

长度 检验和

应用数据

（报文）

3.3.2 UDP校验和

发送方的UDP对报文段的3个16bit（源端口号、目的端口号、长度）的和进行反码运算，求和时的溢出都会回卷，所得结果就放在检验和字段。然后在接收方，接收方会把4个16bit（源端口号、目的端口号、长度、检验和）加在一起，如果和是1111111111111111就没出错，只要出现0就说明出错。

UDP虽然能检验，但对恢复差错毫无办法，要么丢弃，要么将受损报文段交给应用程序并附加警告。

3.4 可靠数据传输原理（这部分必须看书）

第三小节的理解假设建立在单项数据传输的前提下。

* rdt 表示可靠数据传输协议
* \_send指示发送端正在被调用
* udt表示不可靠数据传输

3.4.1 构造可靠数据传输协议

1. 经完全可靠信道的可靠数据传输：rdt1.0

这个是理想状态，最开始的协议的基本内容设置先基于理想状态。

2. 经具有比特差错信道的可靠数据传输：rdt2.0

这种情况是为了解决比特受损。

自动重传请求协议（Automatic Repeat reQuest，ARQ）：

- 增加一些控制使得发送方知道哪些内容被正确接收、哪些内容接受有误并因此需要重传

- ACK 肯定确认

- NAK 否定确认

- 包括了停等协议，stop-and-wait

- 当发送方处于等待接收方确认状态时，它不能从上层获得更多数据

- 发送方每发送一帧后就等待一个确认帧

但是ACK和NAK的传输是有可能受损的！

解决方法：

- 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

- 接收方检查序号即可确定是否重传

所以rdt2的核心就是

- 检验和

- 序号

- ACK分组确认

- 重传

1. 经具有比特差错的丢包信道的可靠数据传输：rdt3.0(3.0也叫比特交替协议）

现在假定除了比特受损，底层信道还会丢包。

所以设定的协议的主要内容就是：

发送方愿意等待足够长的时间等接收方返回信息，以确定分组是否丢失，丢失则重传。

而这个足够长的时间 = 一个往返时延+接收方处理一个分组的时间

实际运用中，发送方会使用倒计数定时器在一定时间值内进行等待

若没有收到传回确认立即重发

（虽然引入了冗余分组，但rdt2.0中的序号设置能够解决冗余的问题）

所以rdt3.0也被称为比特交替协议

所以rdt3.0的核心就是

- 检验和

- 序号

- 定时器

- 肯定和否定确认分组

- 重传

以上，通过总结归纳，我们得到了一个可靠数据传输协议！

3.4.2 流水线可靠数据传输协议

rdt3.0虽然保障了数据的正确性，但性能问题成为隐患。

我们定义发送方的利用率为：发送方实际忙于将发送比特送进信道的那部分时间/发送时间：

Usender = ( L/R )/(L/R + RTT)

L为分组比特长度，R为带宽，RTT为一个往返时间

解决办法：不以停等协议运行，而是允许发送方发送多个分组无需确认

因为许多从发送方 向 接收方 输送的分组可以被看作是一条流水线，所以这种技术叫流水线

- 必须增加分组的序号范围

- 协议的发送方和接收方两端需要能够缓存多个分组

- 对于差错恢复选用 回退N步 或 选择重传

3.4.3 回退N步

Go-Back-N，GBN：允许发送方发送多个分组而不需要等待确认，但它也限制了流水线中未确认的分组个数不允许超过N。

GBN中序号范围的理解：

- base（基序号）：最早未确认分组的序号

- N：窗口长度，所以GBN也叫做滑动窗口协议，设置N也是为了限制发送方进而实现拥塞控制

[0,base-1]：已发送并确认的分组

[base,next-1]:已经发送但未被确认的分组

[next,base+N-1]：准备发送的分组

可能为空，这一部分其实是预留给上层应用程序录入新数据的

GBN的工作过程：

- 上层调用rdt\_send()检查发送窗口是否已满 ：

- 未满：产生一个分组准备发送

- 满了：将数据返回，并告知窗口状态，隔一会儿再尝试

- 发送方等待传回信号，这里分两种情况：

- ①收到一个ACK

表示接收方已经正确收到序号为n的分组包括n以前的所有分组

窗口向后移动到序号为n的分组的下一个分组

- ②超时

发送方重传所有已发送但未确认的分组

- 接收方的动作也分两种情况：

- ①正确接收序号为n的分组，并且n之前的分组序号顺序都正确，返回ACK

并将数据交付给上层应用程序

- ②分组失序，接收方丢弃未确认的所有分组

3.4.4 选择重传

虽然流水线提高了信道利用率，但是接收方丢弃所有未确认分组其实也比较傻，因为有些分组正确只是顺序不对。

选择重传（SR协议）就是为了优化这个问题，协议通过让发送方仅重传那些它怀疑出错的分组，所以在发送发的窗口中对分组的发送状态的划分粒度会更细。

SR发送方：

- 从上层收到数据，检查N之后发送

- 超时重传单个分组（这一步需要让每个分组都有自己的逻辑定时器）

- 收到某个分组的ACK

- 若返回的ACK是窗口的第一个，则窗口向后移动并将该分组确认

- 若返回的分组确认不是第一个分组，那就先确认但不移动

- 如果有分组超时，就只发送超时分组

SR接收方：

- 接收方接收一个分组n

- 检查n之前的是否都上传了

- 是的，说明n的顺序正确，也交付给上层，并返回ACK（累计确认）

- 不是，先缓存（最多缓存N个）

- 这就保证了当前的[base,base+N-1]内的分组一定会被强行排序）

注意：窗口长度必须小于或等于序号空间的一半（效率问题）

3.5 面向连接的运输：TCP

基于上述的铺垫，TCP的核心其实就是：差错检验、重传、累计确认、定时器、序号、确认号的首部字段。

3.5.1 Tcp连接

Tcp的特点：

- 面向连接（三次握手）

- 只在端系统运行，TCP是运输层协议

- 提供全双工服务

- 连接是点对点的（一对一）

3.5.2 TCP报文段结构

|<------------------------- 32bit ------------------------------>|

源端口号 目的端口号

序号

确认号

(首部长度 保留未用 CWR ECE URG ACK PSH RST SYN FIN) 接收窗口

因特网检验和 紧急数据指针

选项

数据

- 源端口号和目的端口号：用于多路分解和复用

- 序号+确认号：用于发送方和接收方实现可靠数据传输服务

- 首部长度（4bit）

- 接收窗口字段：设置N，用于流量控制

- 选项（可变长）：用于发送方和接收方协商最大报文段长度（MSS）

- 标志字段

- URG 指示报文段存在被发送端的上层实体置为“紧急”的数据，一般不用

- ACK 确认字段中的值是有效的

- PSH 指示接收方应立即将数据交给上层，一般不使用

- RST、SYN、FIN：用于连接建立和拆除

3.5.3 往返时间的估计和超时

EstimateRTT = （1-a）\* EstimateRTT + a \* SampleRTT

3.5.4 可靠数据传输

TCP的差错恢复机制是GBN协议和SR协议的混合体。

3.5.5 流量控制

这里重点讲一下TCP中拥塞控制和流量控制的区别。前面我们一直说TCP有拥塞控制的能力，而具体到TCP报文时，其中的窗口大小功能的设置却是为了流量控制。那二者是否是一个东西呢？拥塞控制是如何实现的呢？

我们可以先这样对二者理解：

* 拥塞控制是为了避免网络资源耗尽的问题，通过拥塞窗口实现（它的发展对因特网整个有好处，有全局性）
* 流量控制是着眼于TCP传输的本质要求，它目的是解决发送方和接收方速率不匹配的问题，通过滑动窗口实现（控制已发送但未被ACK的包的数量）

接着我们来看二者如何实现以及运行过程：

拥塞控制

- A与B之间的网络发生堵塞导致传输过慢或者丢包，来不及传输。

- 通过拥塞窗口实现

- 拥塞窗口指 发送端 在RTT内可以发送的最大数据包数

- 拥塞窗口根据网络情况一直动态变化

- 实现过程经历两个阶段：

- 1. 慢启动（指从1开始指数增长到限定大小的过程）

- 2. 拥塞避免（超过限定大小后出现丢包，将拥塞窗口改为1，限定大小减半）

流量控制

- A给B发数据，A发送的太快导致B没法接收(B缓冲窗口过小或者处理过慢)

- 通过滑动窗口实现，有发送端窗口和接收端窗口之分

- 窗口的大小在 接收端 指能够接收的最大字节数！

- 窗口的大小在 发送端 指能够发送的数据的大小

- 窗口大小在TCP报文中设置

- 窗口大小可以根据设置动态变化，但在某通信过程中，确定了就不变了

3.5.6 TCP连接管理

1. 客户端的TCP首先向服务器端发送一个特殊的TCP报文段:

SYN=1，seq = client\_isn

2. 当TCP SYN报文段的IP数据报到达服务器主机，服务器会为该TCP连接分配缓存和变量

然后服务器发回:

SYN=1, seq = server\_isn, ack=client\_isn+1

3. 客户接受后也要准备分配缓存和变量，准备好之后，给服务器发，此时标志连接建立:

SYN=0, seq = client\_isn+1, ack=server\_isn+1

4. 可以开始传输数据

3.6 拥塞控制原理

拥塞网络的代价

* 当分组的到达速率接近链路容量，会有巨大的排队时延
* 发送方必须执行重传以补偿因为缓存溢出而丢弃的分组
* 当一个分组沿一条路径被丢弃，每个上游路由器用于转发该分组到丢弃该分组而使用的传输容量最终被浪费掉了

接收端窗口（Reciver Window)是接收端根据目前的接收缓存大小所许诺的最新窗口值，是来自接收端的流量控制。

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。

前面我们知道拥塞控制是全局性的，那拥塞发生在何时呢？

在实际运用中我们发现

如果B给A发送零窗口报文段不久后，B的接收缓存又有了一些存储空间

于是B再次向A发送空余信号，但这个信号很可能丢失！那么就会出现死锁状态！

- A收到B的零窗口通知之后停止发送，并一直等待B传来新的通知

- B认为自己发送了更新信号，一直等待A发来数据

为了解决这样的问题，需要在A设置一个启动持续计时器：

- A在收到零窗口通知后开始倒计时

- 倒计时结束发送一个零窗口探测报文，确认B当前的窗口状态并采取相应的措施应对拥塞：

TCP采用了两种控制机制：

①慢开始和拥塞避免

②快重传和快恢复

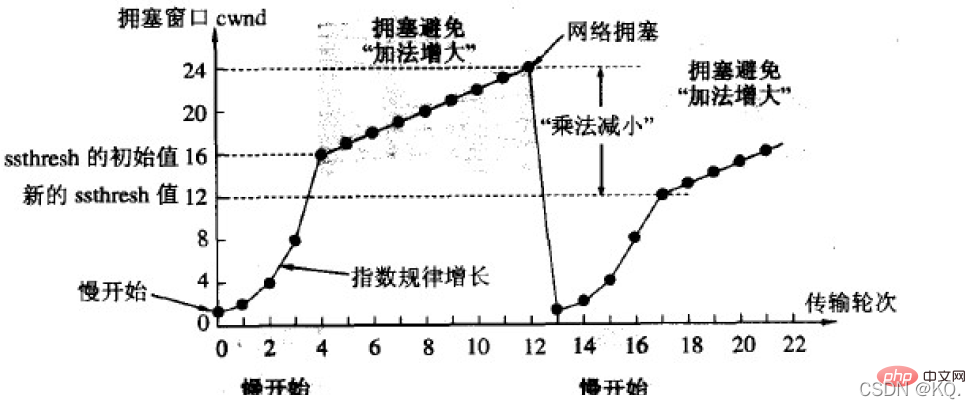
拥塞控制机制①：慢开始和拥塞避免

TCP初始化，拥塞窗口设置为1：主机开始发送数据时，因为不清楚网络的情况，会先试探，所以拥塞控制窗口的数值设置的小

执行慢开始算法，cwind按规律增长：每收到一个对新报文段的ACK确认后，就将拥塞窗口的限制值cwind变大，能比之前传输更多一点的数据

当cwind==ssthress执行拥塞避免：此时已达到ssthresh的初始设定值，但网络还未堵塞，说明ssthress可以增大一些，放出更多数据

网络出现阻塞，把ssthresh值更新为拥塞前ssthresh值的一半，cwnd重新设置为1



拥塞控制机制②：快重传和快恢复

因为等待重传计时器计数会造成资源浪费

* 快重传算法：

目的就是尽可能在更早时间重传丢失报文（比如收到三个重复确认的ACK就断定分组丢失，开始重传）

快重传算法首先要求接收方每收到一个失序的报文段后就立即发出重复确认（为的是使发送方及早知道有报文段没有到达对方）

* 快恢复算法：

当发送方连续收到三个重复确认时，就执行“乘法减小”算法，把慢开始门限ssthresh减半，这是为了预防网络发生拥塞。

由于发送方现在认为网络很可能没有发生拥塞，因此现在不执行慢开始算法（会让cwind从1开始），而快恢复是把cwnd值设置为当前更新后的ssthresh的一半，然后开始执行拥塞避免算法，使拥塞窗口的线性增大。

第四章：网络层（数据平面）

网络层是实现主机到主机的通信服务的。

网络层的数据平面功能指的是路由器的功能，它能决定到达路由器输入链路之一的数据报（网络层的分组）是如何转发到该路由器的输出链路之一的。

4.1 网络层概述

网络层的作用是将分组从一台发送主机移动到一台接受主机，使用的两个重要功能：①转发；②路由选择

分组交换机：指一台通用分组交换设备，根据分组首部字段中的值，从输入链路接口到输出链路接口转移分组。

路由器：基于网络层数据报中的首部字段值做出转发决定。

4.2 路由器工作原理

首先明晰路由器的组成：

- 输入端口：含有转发表

- 交换结构

- 输出端口

- 路由选择处理器

4.2.1 输入端口处理和基于目的地转发

分组通过端口进入路由器，然后根据转发表设置的地址范围，被送到相应的输出端口。

值得一提的是，转发表是：目的地址范围——链路接口；而目的地址范围匹配采用的是最长前缀匹配规则。

4.2.2 交换方式

* 内存交换
* 总线交换
* 互联网络交换

4.2.3 输出端口处理

输出端口处理包括：排队（缓存管理）——> 数据链路处理（协议、封装）——> 线路端接

4.2.4 何处出现排队

输入排队

输出排队

4.2.5 分组调度

* 先进先出
* 优先权排队
* 循环和加权公平排队（循环排队+优先权）

4.3 网际协议：IPv4、寻址、IPv6及其他

如今有两个版本的IP正在使用：①IPv4；②IPv6

掌握IP编址就是掌握了因特网的网络层！

4.3.1 IPv4数据报格式

|<------------------ 32bit ------------------->|

版本 首部长度 服务类型 数据报长度（字符）

16bit标识 标志 13比特片偏移

寿命 上层协议 首部检验和

32比特源IP地址

32比特目的ip地址

选项（如果有的话）

----------------------------总共20字节（160bits的报头）------------

数据

- 版本号：规定了数据报的IP协议版本，路由器才能知道按照哪个版本解释

- 首部长度：用来确定IP数据报中载荷

大多数IP数据报不包含选项，所以一般的IP数据报具有20字节的首部

- 服务类型：用于区别不同的IP数据报

- 数据报长度：IP数据报的总长度（首部+数据）

- 标识、标志、片偏移：与IP分片有关

- 寿命：Time-To-Live，TTL

每一台路由器处理数据报该值减1，为0是该数据报必须丢弃

- 协议：指明数据报的数据部分交给哪个特定的运输层协议

协议号可以理解为网络层和运输层的粘合剂

端口号是运输层和应用层的粘合剂

- 首部检验和：检测收到的IP数据报中的比特错误

- 选项：用于扩展IP首部

一般IP数据报首部是20位，但TCP报文段的首部则为40字节（20IP+20TCP）

4.3.2 IPv4数据报分片

并不是所有链路层协议都能承载相同长度的网络层分组。

最大传送单元（Maximum Transmission Unit,MTU）：一个链路层 帧 所能承载的最大数据量。

解决方法：将IP数据报中的数据分片成两个或更多个较小的IP数据报，并用单独的链路层帧封装这些较小的IP数据报。

而IPv4的设计者出于对路由器性能的考量，将数据报的重新组装工作放到了端系统中，而不是放到网络路由中。所以主机会根据IP数据报中的标志、标识、片偏移对分片的IP数据报进行组装。

4.3.3 IPv4编址

IP地址长度为32bit（等价为4字节）

- 采用点分十进制技法：每一个字节的二进制转化为十进制表示，并且各字节间用句号隔开

- 每台主机和路由器上的每一个接口都必须唯一标识

- 互联主机接口和路由器接口的网络称为子网，子网的地址编址中/24记法称为子网掩码

CIDR无类别域间路由选择（Classless Interdomain Routing,CIDR）

- 形如：a.b.c.d/x 的地址

- x指示了地址的第一部分中的比特数，剩下的32-x比特被用于区分组织内部设备

- eg：a.n.c.d/21

前21比特定义了该组织的网络前缀，该组织中所有主机的IP前21位都相同

剩余11位用于划分子网

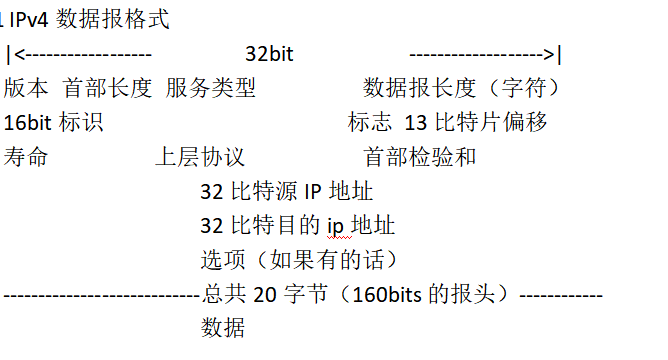
IP广播地址（255.255.255.255）

4.3.4 网络地址转换

子网的飞速发展使得原先的地址编码不适用，出现了新的方法：网络地址转换

4.3.5 IPv6

是为了解决新的子网和IP节点以惊人的增长率连到因特网上造成32bit的IP空间即将耗尽的问题。



|<------------------ 32bit ------------------->|

版本 流量类型 流标签

有效载荷长度 下一个首部 跳限制

源地址（128bit）

目的地址（128bit）

数据

- 扩大了地址容量

- 很多IPv4被弃用或作为了选项

- 流标签：是IPv6最难以捉摸的，用于给属于特殊流的分组打上标签，能用于优先权

- 流量类型：也就是服务类型，用于区别不同的IP数据报

- 下一个首部：表示数据报中的内容需要交付给哪个协议，隐含了“选项”的功能

- 跳限制：每一台路由器处理数据报该值减1，为0是该数据报必须丢弃

第五章：网络层（控制平面）

控制平面作为一种网络范围的逻辑，不仅控制沿着从源主机到目的主机的端到端路径间的路由器如何转发数据报，而且控制网络层组件和服务如何配置和管理。

5.1 概述

转发表和流表是链接网络层的数据平面和控制平面的首要元素。

5.2 路由选择算法

按照算法是集中式还是分散式来划分：

- 集中式路由选择算法（其中具有全局状态信息的算法被称作链路状态算法——LS）

- 分散式路由选择算法

按照算法是静态还是动态来划分：

- 静态路由选择算法

- 动态路由选择算法

按照负载敏感或是负载迟钝进行划分：

- 负载敏感算法

- 负载迟钝算法

5.2.1 链路状态路由选择算法

Dijkstra算法：计算从某节点（源节点设为u）到网络中所有其他节点的最低开销路径。它是迭代算法。

整理一下：

* RIP 路由信息协议，一种分布式的基*于距离向量*的路由选择协议记忆法：RIP等于dv
* 自治系统AS
* 自治系统内部路由选择协议：e.g.开放最短路优先OSPF 记忆法：ospf等于ls
* 自治系统间路由选择协议： e.g.边界网关协议BGP

5.2.2 距离向量路由选择算法

距离向量算法（Distance-Vector,DV）是一种迭代的、异步的、分布式的算法。

5.3 因特网中自治系统内部的路由选择：OSPF

开放最短路径优先（OSPF）：是一种链路状态协议。每台路由器在本地运行Dijkstra的最短路径算法，以确定一个以自身为根节点到所有子网的最短路径树。

OSPF的优点：

- 安全：因为它能够鉴别OSPF路由器之间的交换（如链路状态更新）

- 多条相同开销路径：当到达某目的地的多条路径具有相同的开销时，OSPF允许使用多条路径

- 对单播和多播路由选择的综合支持

- 支持在单个自治系统中的层次结构

5.4 ISP之间的路由选择：BGP

AS-自治系统

AS间路由选择协议涉及多个AS之间的协调，所以AS通信必须运行相同的AS路由选择协议（Broder Gateway Protocol，BGP），BGP将因特网中数以千计的ISP粘合起来，是一种分布式和异步的协议。

5.6 ICMP:因特网控制报文协议

ICMP被主机和路由器用来彼此沟通网络层的信息。ICMP的最典型的用途是差错报告。

ICMP通常被认为是IP的一部分，但从体系结构上讲它位于IP之上，因为它承载在IP分组中

第六章：链路层和局域网

链路层信道有两种：

- 广播信道

- 用于连接有线局域网、卫星网和混合光纤同轴电缆接入网的主机

- 需要媒体访问协议来协调帧的传输

- 也可以使用中心控制器来协调传输

- 点对点通信链路

- 用于在长距离链路连接的两台路由器之间

- 用于用户办公室计算机与它们所连接的邻近以太网交换机之间等场合

6.1 链路层概述

先规范术语：

- 节点：运行链路层协议的设备（包括主机、路由器、交换机和WIFI接入点）

- 链路：沿着通信路径连接相邻节点的通信信道

- 链路层帧：传输节点将数据报封装成的对象

6.1.1 链路层提供的服务

* 成帧：每个网络层数据报在传送之前，都会被链路层协议使用链路层帧 封装起来。帧 = 数据字段 + 若干首部字段 ；其中网络层数据报就插在数据字段中。
* 链路接入：媒体访问控制（Medium Access Control,MAC）协议规定了帧在链路上的传输规则。
* 可靠交付：链路层的可靠交付是通过确认和重传实现的。
* 比特级差错检查和纠正：当帧中的一个比特为1传输时，接收方节点中的链路层硬件可能不正确地将其判断为0。这种比特差是信号衰减和电磁噪声导致的。

6.1.2 链路层在何处实现

链路层的主体是在网络适配器（也称网络接口卡）中实现的。

而网络适配器的核心是链路层服务器 （一种芯片）。

由此说明，链路层是硬件和软件的结合体，链路层是协议栈中软硬件交接的地方。

6.2 差错检测和纠正技术

差错检测和纠正比特（Error-Detection and- Correction,EDC）技术可以增强数据D。注意，差错检测比特技术也存在未检出比特差错的风险。

目前常用的3种差错检查技术：

* 奇偶校验：一般用来描述差错检查和纠正背后隐含的基本思想
* 检验和方法：通常用在运输层
* 循环冗余检测CRC：通常用在适配器中的链路层

6.2.1 奇偶校验

偶校验

- 在有d个比特的发送信息末尾附加一个比特

- 选择它的值

- 使得这总共d+1比特的数据中1的个数为偶数。

- 接收方收到数据后检验这d+1个比特中1的个数是否为偶数

所以我们不难发现，无论是奇校验还是偶校验，接收方最多只能检测出1个错误，而这并不能表示实际错误只错了1个，甚至偶数个错误将无法检出。

6.2.2 检验和方法

把d比特数据作为一个k比特整数的序列处理，将这k比特整数加起来，得到的和作为差错检测比特。

TCP和UDP报文中的检验和用的就是这个原理。

6.2.3 循环冗余检测

循环冗余检测编码（Cyclic Redundancy Check,CRC）

CRC内容：

- 被发送的数据段D有d个比特

- D的末尾附加r比特的序列，这r个比特的编码称为R

- 收发双发约定一个固定的多项式G（生成多项式），长度为r+1比特

- CRC的过程：

-发送方：

①拿到准备发送的数据段D

②与接收方约定一个固定的多项式G

③ R = 结果的余数[（D \* 2^r）/ G]

④ 将DR发送给接收方

- 接收方

①用接受到的DR共d+r个比特去除G，观察余数

不为0，发生错误

除数为0，无错误

举例：

- D = 101001

- 双方约定G = x^3 + x^2 + 1（对应的比特为：1101），得到r=3

- 所以发送方：

D’ = D \* 2^3 =101001000(r有几个，就在D后面加几个0就好了)

R = remainder[D’ / G]

= 001

DR = 101001 001

发送

- 接收方：

DR =101001 001

已知G =1101

DR / G = 110101……0

余数为0，说明传输没有出错

6.3 多路访问链路和协议

首先，先了解两种类型的网络链路

* 点对点链路：由链路一端的单个发送方和链路另一端的单个接收方组成。
* 广播链路：它能够让多个发送和接收节点都连接到相同的、单一的、共享的广播信道上。

多路访问：就是协调多个发送点和接受点对于一个共享广播信道的访问。

碰撞：指多个节点同时传输帧，造成接收方处的信息矛盾。碰撞帧的信号会纠缠在一起，所有的帧都会丢失。

多路访问协议：节点通过这些协议来规范自己在共享的广播信道上的传输行为。

现在多用三种协议：①信道划分协议；②随机接入协议；③轮流协议。

6.3.1 信道划分协议

主要讨论3种

* 时分多路复用（TDM）：分配时隙 用于传输数字信号
* 频分多路复用（FDM）：分配频率 用于传输模拟信号
* 码分多址（CDMA）：分配编码

TDM

- 将时间划分为时间帧

- 并把每个时间帧再划分为N个时隙

- 然后把每个时隙分配给N个节点中的一个

- 这样当节点有数据分组想要发送的时候，他只会在被分配到的时隙里进行操作

- TDM的特点：

- 公平并且能够消除碰撞

- 它会限制节点的平均速率为R/N bps

- 节点总要等待自己的轮次

FDM

- 将传输速率为R bps 的信道划分为不同的频段

- 每个频段的带宽为R/N

- 然后把不同的频率分配给N个节点

- FDM特点

- 避免了碰撞，公平划分了带宽

- 限制了节点的带宽

CDMA

- 为每个节点分配不同的编码

- 然后节点用它唯一的编码来对它发送的数据进行编码

- 能做到不同节点同时传输，因为接收方能够通过数据中的编码对这些数据流进行区分

- CDMA贵，常用于军用系统，抗干扰性很强

6.3.2 随机接入协议

随机接入协议中，一个传输节点总是以信道的全部速率（R bps）进行发送，有碰撞时，涉及碰撞的节点等待一个随机时延进行反复重发，直到没有碰撞为止。

主要了解4种随机接入协议：

①时隙ALOHA

②ALOHA

③载波监听多路访问（CSMA）

④具有碰撞检测的载波侦听多路访问（CSMA/CD）

时隙ALOHA

- 前置条件：

所有帧都是L比特

时间划分为L/R个时隙（一个时隙传输一帧）

每个节点设置一个概率p

要求所有节点同步它们的传输

- 时隙ALOHA的运作流程：

- 某个节点有一个帧要发送，它等到下一个时隙开始传输整个帧

- 若没有碰撞，正常传送

- 有碰撞，以概率p在后续的时隙中重传

- 时隙ALOHA是对信道划分的改进，它使得在固定时间段，该帧的传输速率是完整的R bps

- 但碰撞和概率等待会造成浪费

ALOHA

- 非时隙、完全分散的协议

- 当一个帧首次到达，节点立即将帧完整传输进广播信道

- 如果碰撞，先传输完与它碰撞的帧，然后立即以概率p重传，否则等待一个帧

- 等待后，再次以p概率重传，否则以1-p在另一个帧等待

载波监听多路访问（CSMA）

- 说话之前先听（载波侦听）：一个节点在传输前先听信道，直到没有传输再发送

- 如果与他人同时开始说话，停止说话（碰撞检测）：再重复载波侦听等待一段随机时间

- CSMA的缺陷在于它无法识别发送方正处于传输时延

它会误以为信道空，而实际上对方正在准备发送数据还没来得及占领信道

具有碰撞检测的载波侦听多路访问（CSMA/CD）

- 1.适配器从网络层一条获得数据报，准备链路层帧，并放入帧适配器缓存

- 2.适配器查看信道状态

- 若适配器帧听到信道空闲，开始传输

- 若信道忙，它等待

- 3.传输中，适配器监视其他适配器的信号能量（这一步是对CSMA的优化）

- 没有检测到其他信号能量，传输完自己的任务

- 检测到，立即停止，等待一个随机量，然后返回步骤2

- 关于CSMA/CD效率的定义：

1

效率 = ——————————————————————————

1 + 5\*传播时延/传输时延

传播时延为0，效率为1，说明碰撞的节点会立即终止而不会浪费信道

传输时延很大，效率会趋近1，说明帧取得了信道时，它会使信道处于很长的有效状态

Csma/cd 与 csma/ca区别

这里补充一起说一下CSMA/CA：全称是带冲突避免的载波侦听多址接入协议

ca主要用于wlan无线局域网，在后面会讲到，ca是对csma/cd的优化

两者最重要的区别就在于

- csma/cd是发生冲突后及时检测

- 而csma/ca是发送信号前采取措施避免冲突

csma/cd是通过检测物理信道上信号电平的值来判定信道上是否有信号在发送：

- 假设一个用户站发送数据时，信道上的电平范围在0～3v

- 当有多个用户站同时发送信号，信道上的各信号就会叠加使电平增大从而大于3v

- 一旦监测信道的用户站发现信道上的电平大于正常值的话就判定发生了冲突，立即停止发送

- 等待一个随机过程再对信道进行监听。

csma/ca适用于无线环境，因为无线信道存在隐蔽站和暴露站的问题：

- 工作站a如果要给c发送数据

- a会首先激励c

- 使其广播一个短信号告诉周围的用户站自己要接收信号数据

- 这时收到信号的用户站就知道c站正忙，不再向它发送数据，从而避免了冲突。

6.3.3 轮流协议

它的核心其实是 信道划分 + 随机接入，也就是既要实现 只有一个节点活跃时，该节点吞吐量达到R bps，又要有多个节点活跃时，每个活跃节点的吞吐量接近 R/M bps。

轮流协议分两种：

①轮询协议

②令牌传递协议

轮询协议

- 要求节点之一为主节点，主节点循环轮询每个节点告诉它们能传输的帧的最多数量。

- 轮询协议消除了碰撞和空时隙，但缺点：

引入了轮询时延（就是通知时延）；

主节点瘫痪整个信道玩儿完。

令牌传递协议

- 一个称为令牌 的小的特殊帧 以某种固定的次序 在节点间交换

- 持有令牌的帧能够发送最大数目帧数，发送完把令牌转交

- 缺点：一个节点故障也会使信道崩溃

6.4 交换局域网

因为某些交换机运行在链路层，所以它们交换链路层帧，不识别网络地址，不使用RIP和OSPF这样的路由选择算法来确定路径。

6.4.1 链路层寻址和ARP

主机和路由器也具有链路层地址，这里要注意区分网络层中的IP地址。二者有明显不同：

链路层地址指的是主机和路由器的适配器（网络接口）的地址，因为一个主机可以有多个网络接口

IP地址是因为一个路由器能够关联多个主机

这样的原因是因为链路层交换机的任务是透明的在主机和路由器之间承载数据报。

链路层地址有多种称呼：①LAN地址；②物理地址；③MAC地址

MAC地址用的多一点，所以以下我们用MAC地址代表所有的链路地址。

MAC的特点是：

- 适配器的MAC地址各不相同

- MAC地址不会随适配器的移动而变化，IP地址如果转换了区域就会变化

- 适配器会检查帧的目的MAC地址是否与自己的匹配：

匹配则提取数据报向网络层传递

不匹配就丢弃

现在我们知道了存在MAC地址和IP地址，二者不尽相同，那么当我们需要二者之间进行转换时，就用到了地址解析协议（Address Resolution Protocol,ARP）。

发送主机中的ARP模块将取在相同局域网上的任何IP地址作为输入

然后返回相应的MAC地址

* IP->MAC ARP
* 域名->IP DNS

所以可以看出每个节点都维护着一张ARP映射表

- ARP只为在同一个子网上的主机和路由器接口解析IP地址

- DNS可以为因特网上任何地方的主机解析IP地址

6.4.2 以太网

以太网是有线局域网。以太网技术向网络层提供的是不可靠服务，虽然有CRC，但是无论传输成功与否，接收方没有ACK确认。

以太网的优点：

- 第一个广泛部署的高速局域网

- 以太网成本便宜，实现简单

- 数据效率高

以太网帧结构：

前同步码 目的地址 源地址 类型 数据 …… CRC

- 前同步码（8字节）：前7个字节用于唤醒接收适配器并同步时钟，且都是10101010

最后一个字节为10101011

- 数据字段（46~1500字节）：承载了IP数据报

如果数据小于46，就用IP数据报首部填充

- 目的地址（6字节）：该字段包含目的MAC地址 BB-BB-BB-BB-BB-BB

- 源地址（6字节）：传输该帧到局域网上的适配器的MAC地址

- 类型字段（2字节）：该类型允许主机复用多种网络层协议

- CRC（4字节）：循环冗余检测码

6.4.3 链路层交换机

交换机的任务是接收入链路层帧并将它们转发到出链路。交换机自身对于子网中的主机和路由器是透明的（不可见的，黑盒复用）。

交换机运行原理：

- 交换机的转发和过滤（借助交换机表实现）

- 过滤：决定一个帧应该转发到某个接口还是应当将其丢弃的交换机功能

- 转发：决定一个帧应该被导向哪个接口

————假设有一个从接口x到达的帧：

表中没有该帧MAC地址对应的表项，交换机广播该帧

该帧MAC跟x匹配，交换机丢弃该数据完成过滤操作

（这意味着该帧已经在包含目的地的局域网网段广播过了）

该帧MAC地址跟另一个接口y匹配，执行转发

- 交换机表是自动、动态建立的（自学习）。

- 交换机表初始为空

- 每次接收到帧自动记录：

- 记录该帧源地址MAC

- 该帧到达接口

- 当前时间

- 经历一段时间没有接受到该地址作为源地址的帧，就删除源地址

- 交换机和路由器的对比：

- 交换机：使用链路层的MAC地址转发分组（属于第二层的分组交换机）

- 路由器：是使用网络层IP地址转发分组的交换机（属于第三层的分组交换机）

6.4.4 虚拟局域网

虚拟局域网（Virtula Local Network，VLAN）

支持VLAN的交换机允许经一个单一的物理局域网基础设施定义多个虚拟局域网。

第七章：无线网络和移动网络

7.1 概述

无线主机：运行应用程序的端系统设备。

无线链路：主机通过无线通信链路连接到一个基站或者另一台无线主机。

基站：负责向与之关联的无线主机发送数据和从主机接收数据。

切换：一台移动的主机的移动超出了一个基站的覆盖范围而到达另一个基站的覆盖范围，它将改变其接入到更大网络的连接点。

7.2 无线链路和网络特征

* 递减的信号强度（路径损耗）
* 其他源的干扰
* 多径传播

信噪比（Signal -to-Noise Ratio，SNR）：是所受到的信号和噪声强度的相对测量。

比特查错率（BER）：接收方收到的有错传输比特的概率。

7.3 WiFi：802.11无线LAN

IEEE 802.11无线LAN（也称为WiFi）均使用相同的媒体访问协议（CSMA/CA）

7.3.1 802.11体系结构

基本服务集（Basic Service Set，BSS）

- 多个无限站点

每个无线站点都有一个6字节的MAC地址

- 一个称为接入点（Access Point，AP）的中央基站

802.11 标准要求

- 每个AP周期性发送信标帧，每个信标帧包括该AP的SSID和MAC地址。

7.3.2 802.11 MAC协议

使用CSMA/CA协议(这儿不全l

假设一个站点有一个帧要发送：

1. 若某站点最初监听到信道空闲，它将在一个 分布式帧间间隔（DIFS）后发送帧

2. 否则，选取一个随机回退值，并开始计数

3. 计数值减为0，该站点发送整个数据帧并等待确认

4. 如果收到确认，发送站点知道帧被正确接受。

7.3.5 802.11中的高级特色

如果一个节点连续发送两个帧而没有收到确认（信道上一个比特查错的隐式指示），该传输速率降低到前一个较低的速率。如果10个帧连续得到确认，或如果用来追踪自上次降速以来时间的定时器超时，该传输速率提高到上一个较高的速率。

7.3.6 个人域网络：蓝牙和ZigBee

第八章：计算机网络中的安全

了解基本概念和协议即可

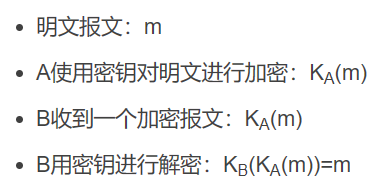
8.2 密码学的原则

假设A向B发送一个报文

报文的最初形式称为明文，A使用加密算法对明文进行加密得到密文。而可以阻止 入侵者解密 被传输的数据 的秘密信息叫 密钥。

A给B提供了一个密钥KA，密钥是一串数字或字符，作为加密算法的输入之一。加密算法以密钥和明文报文作为输入，生成的密文为输出。B则会为解密算法提供密钥KB，密文和密钥是解密算法的输入，输出为明文

我们用符号对上述事件进行描述

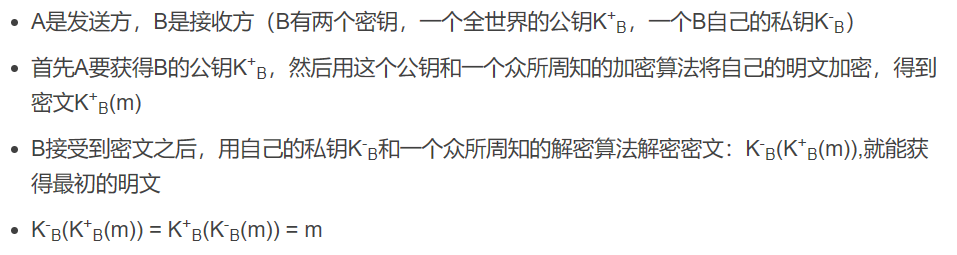


对称密钥系统中，A和B的密钥相同且对外保密。

8.2.2 公开密钥加密

公开密钥系统中，使用一对密钥，一个密钥是所有人知道的K+B，另一个密钥只有A或B中一个人知道K-B。

假设AB之间要通信



8.3.1 密码散列函数

密码散列函数的性质：不能找到：x不等于y，满足H(x)=H(y)

所以这就保证了发送方生成的（m，H(m)）不能被伪造。

8.3.2 报文鉴别码

虽然密码散列可以让报文独一无二，但是如果入侵者对B宣称自己才是真正的A，B是无法判断的。所以AB之间约定一个字符串s——鉴别密钥,将其附加到明文再通过密码散列函数得到H(m+s)称为报文鉴别码

AB的通信变成：

* A生成报文m，用s级联m生成m+s，用密码散列函数得到H(m+s)，将（m,H(m+s)）发给B
* B收到（m,h），因为知道s，所以能够自己根据散列函数计算出H(m+s)，如果h=H(m+s),则一切正常

8.4 端点鉴别

8.4.1 鉴别协议ap1.0

A直接给B发报文，说自己是A

入侵者可以冒充A

8.4.2 鉴别协议ap2.0

A有一个IP地址，B通过验证 报文携带的的IP数据报的源地址 是否 与A的IP地址相匹配来鉴别A

入侵者可以获得A的IP地址进而冒充A

8.4.3 鉴别协议ap3.0

AB约定一个密令口令进行通信

入侵者可以窃听AB以此获得密令，然后冒充A

8.4.4 鉴别协议ap3.1

AB使用一个加密的口令

入侵者可以窃听A的通信，并记录A对应的加密版本，然后给B回放加密版本的录音，以此假装A

8.4.5 鉴别协议ap4.0

3.0失败的原因是B无法判断A是否处于活跃状态，所以4.0选择利用不重数——一个协议的生存期只能使用一次的数。

* A向B发送报文
* B选择一个不重数R，发给A
* A使用和B约定好的对称密钥加密这个不重数：KA-B(R），发给B
* B收到加密密钥之后解密，如果结果等于自己设置的R，B就知道A活跃，之前收到的报文确实是A发来的，便开始解密报文

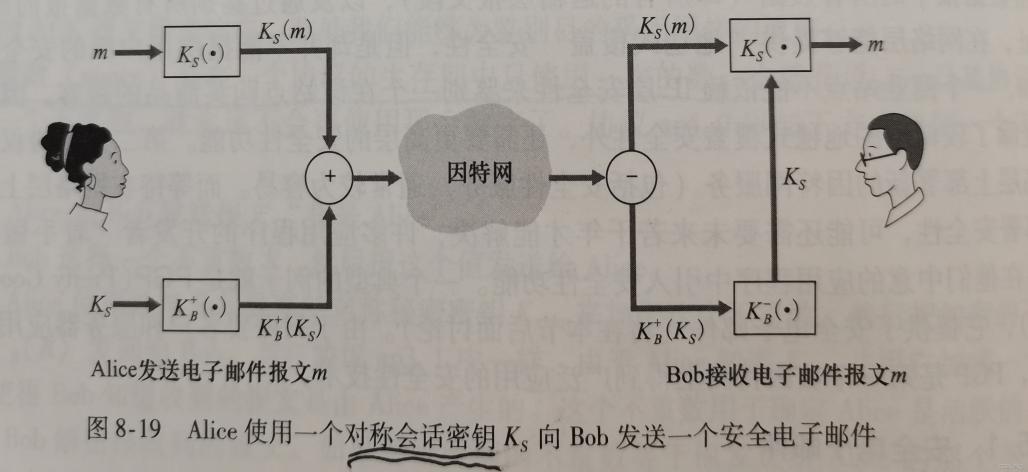
8.5.1 安全电子邮件

我们希望有的安全特性：

* 机密性（重中之重）
* 具备发送方鉴别
* 报文完整性
* 接收方鉴别

只保证机密性并且效率不错的情况：

* A选择一个随机对称会话密钥Ks
* A用这个对称密钥加密明文m： Ks(m)，再使用B的公钥 K+B加密这个密钥: K+B(Ks)
* A把上一步得到的两个部分进行级联形成一个包（加密的报文+加密的对称密钥）
* A把包发给B,B把包进行分解
* B收到之后用自己的私钥 K-B先解开加密的对称密钥得到Ks：K-B(K+B(Ks)）= Ks
* 用Ks解开加密的报文得到明文m



不关心机密性，只关心发送方鉴别和报文完整性（需要使用数字签名和报文摘要）：

* A对明文m使用散列函数H得到报文摘要H(m)
* A使用自己私钥K-A对报文摘要进行签名，得到数字签名K-A(H(m))

报文摘要H 数字签名K-（私钥）

* 把明文m和数字签名进行级联打包，发送给B
* B接受包，分解
* B用A的公钥K+A解密数字签名：K+A(K-A(H(m)))得到一个结果h
* B再用分解包得到的明文m，送进散列函数得到H（m）
* B比较h和自己得到的H(m)是否一致，若一致，B可以完全相信这个报文来自A并且未被篡改

能够提供机密性、发送方鉴别和报文完整性：

（报文机密性是指仅有发送方和预定的接收方能够明白得传输的报文内容。

报文完整性指信息在传输、互换、存储和处置进程中维持非修改,非破坏和非丢失的特性。）

