计算机网络

DZMING LI

目录

1 计算机网络概述 3
1.1 计算机网络基本概念 3
1.2 计算机网络体系结构 4
2 物理层 6
2.1 通信基础 6
2.2 传输介质 11
2.3 物理层设备 12
3 数据链路层 13
3.1 数据链路层的功能 13
3.2 组帧 13
3.3 差错控制 13
3.4 流量控制与可靠传输机制 16
3.5 介质访问控制 17
3.6 局域网 20
3.7 广域网 23
3.8 数据链路层设备 23
4 网络层 25
4.1 网络层的功能 25
4.2 路由算法 26
4.3 IPv4 27
4.4 IPv6 32
4.5 路由协议 33
4.6 IP 组播 35
4.7 移动 IP 36
4.8 网络层设备 36
5 传输层 37
5.1 传输层提供的服务 37
5.2 UDP 协议 38
5.3 TCP 协议 38
6 应用层 43
6.1 网络应用模型 43
6.2 DNS 系统 43
6.3 FTP 44
6.4 电子邮件 45
6.5 WWW 46

计算机网络概述

Chapter 1

1.1. 计算机网络基本概念

1.1.1. 计算机网络的定义、组成与功能

应当区分 internet 和 Internet, internet 应当翻译为互连网，由多个计算机网络互联而成，不管你用什么方式。Internet 则是一个专有名字，翻译成互联网或者因特网。 也就是我们每天都上的上的那个最大的，使用 TCP/IP 协议的网。

互联网由网络边缘和网络核心构成。

网络边缘由端系统 end-system 和接入网络 access network 构成。我们日常的电子设备和服务器都属于 end-system 或者 host.

1.1.2. 计算机网络的分类

按照覆盖范围分类： 广域网、城域网、局域网、个人区域网、

城域网：宽带城域网，往往包含管理平台，业务平台，网络平台。

核心交换层，汇聚层，接入层。

个人区域网：简称 WPAN，包含蓝牙和 zigbee 两张典型技术

局域网通常采用交换技术，覆盖范围较小。

巨大企业和校园网就是典型的城域网。

还可以**按拓扑结构分类**。

总线性：用当个传输线连接所有计算机。建网和增删设备容易，通信效率低，故障影响大。星型：各个终端连接到一个中央设备。换言之，任意两个其他节点通信都需要经过中央节点。 环型：计算机连接成闭合回路，单向传播，可以是单环或者双环 树型：是星型的扩展，节点之间按层次连接，信息交换在上下层节点进行，同层节点不交换或极少交换

网状：节点之间的连接是任意的，无规则，又称无规则拓扑

这些基本分类可以混合起来连成混合网络。

1.1.3. 计算机网络的主要性能指标

- **速率** speed: 速率是对于某条具体的通信链路而言的。但我为 bit per second，也就是 bps
- **吞吐** Throughput：吞吐量和速率的区别是速率是瞬时速度，而吞吐量是平均速度。吞吐的单位往往是 Mbps/Gbps

1.1.3a. 时延

时延 Delay：由四部分组成

- 发送时延: 也称为**传输时延** $d_t = \frac{L}{R}$，其中 R 是链路带宽，单位是 bit/s，在物理层的通信基础章节会讲得更详细, L 是分组长度单位为 bit
- 传播时延: 链路长度除以速度
- 结点处理时延: 记为 d_{proc} 常常被忽略
- 排队时延: 记为d_q，不稳定

1.1.3b. 时延带宽积

$$G = d_p \times R$$

- **误码率**: $P_e = \frac{N_e}{N}$

带宽也是性能指标，我们会在下一章节中学到带宽和速率的关系。

1.2. 计算机网络体系结构

1.2.1. 计算机网络分层结构

物理层，数据链路层，网络层，传输层，应用层

1.2.2. 计算机网络协议、接口、服务等概念

协议由以下三个要素构成：

- 语法：实体之间交换的格式和结构
- 语义：通过解释数据信息的意义，告诉我们该发什么和响应什么
- 时序：也称为**同步**，定义实体之间交换信息的顺序

协议是用来描述两个对等实体之间的,而对于自上而下的,我们称之为服务。比如电子邮件是服务，而 http 是协议。

1.2.3. *ISO/OSI* 参考模型和 *TCP/IP* 参考模型

ISO/OSI 模型在应用层中间分离出了表示层和会话层。 从上到下：应用层，表示层，会话层

会话层负责建立会话，更具体一点：核实双方身份确认是否有权参与会话，达成一致，支付费用，管理和控制会话：保存数据，重新建立连接

表示层：处理应用实体间交换数据的语法，解决格式和数据表示的差别。加密解密，字符编码转换，解压缩都属于表示层

为什么表示层在会话层上面呢,因为在下面的不用理解上面是什么。会话层不需要在意自己传的数据到底是什么意思。

传输层在会话层实体之间提供端到端的数据传送。

TCP/IP 模型把数据链路层和物理层都归为网络接口传送层。

物理层

CHAPTER 2

2.1. 通信基础

2.1.1. 信道、信号、带宽、码元、波特、速率、信源与信宿等基本概念

2.1.1a. 信道、信号

- **信道**: 信号的传输介质
- **信号**可以分为模拟信号和数字信号，模拟信号是一个连续的波形，数字信号是离散的取值。数字信号当然同样必须依赖真实的物理现象来传播，而这些物理现象本身通常是连续变化的。

模拟通信和数字通信最大的区别在于我们如何利用这些物理变化来表示信息。模拟方式中电压的具体大小可以直接对应某个连续的信息值。例如 1.21 V 与 1.22 V 可以代表两个略有不同的声音幅度。数字方式则预先约定一组离散的符号或状态。即使电压在一定范围内变化，也可以表示同一个符号。i

信道可以分为**恒参信道**和**随参信道**。

2.1.1b. 带宽

- **带宽**: 在模拟通信中，带宽用来表示信号所能传输的频率范围，单位是 Hz，因为频率范围越大，就往往意味着可变识别的信号状态越大。

2.1.1c. 码元

码元: 在数字通信系统中，通常用时间间隔相同的信号表示一个离散值，这样的时间间隔内的信号就是码元，这个时间间隔称为**码元长度**，在这个码元的状态数决定了它能表达的信息是多还是少，如果状态数 M 大于 2，就称为 M 进制码元，状态数取决于数字通信系统能区别几种电平状态的脉冲

一个二进制码元携带 1bit 的信息，一个四进制的码元携带 2bit 的信息，一个 M 进制携带 $\log_2 M$ bit 的信息

2.1.1d. 波特、速率

码元速率是每秒传送的码元数目 单位是**波特**又称**波特率**，记为 R_B

信息速率或**数据传输速率**表示每秒传送的二进制比特速，单位是 bit/s，因此也称**比特率**，记为 R_b

因此我们有

$$R_b = R_B \log_2 M$$

2.1.1e. 信源与信宿

- **信源**是发送数据的源头，取决于发出的信号是模拟还是数字，信源可以分为**模拟信源**和**数字信源**，电话机就是模拟信源，计算机就是数字信源。
- **信宿**: 接受数据的终点

信道并不代表是一条电路。

在数据的传输过程中，我们接受的并非是整个波，而是通过采样才能得到某一时刻的电平高低，从而获得信息。

在同步传输中，双方拥有相同的时钟。靠这个来确保双方的时间间隔一样。

但时钟同步并没有那么简单，于是我们有异步传输，通过

单向通信：只有一个方向的通信，没有回来的方向，比如电视和广播 **半双工通信**: 双方都可以发送跟接受，但不能同时发送/接受。因此一般是一方发送一方接受，然后过段时间反过来 **全双工通信**: 双方可以同时发送和接受信息

值得注意的是，单工在现实中的用法有时不严谨，单工无线电台其实是双向交替通信。

信源发送出去的数据不一定是适合在信道上传播的，因此会通过变换器。从信道到信宿因此会需要反变换器。

2.1.2. 奈奎斯特定理与香农定理

2.1.2a. 奎奈斯特公式

想象一下，我们有一根电话线，希望用它将信息从城市 A 传到城市 B。你可以在电话线的一端施加不断变化的电压，而另一端可以检测电压的变化。 假设你想传递一串二进制数字：我们规定：0 用低电压表示。1 用高电压表示。每隔一段固定时间，发送一个数字。这时，我们会想：我能让这根电话线每秒传递多少个数字？如果只要缩短每个数字占用的时间，就可以无限提高传输速度，那么一根电话线岂不是能够传输无限多的信息？ 但事实并非如此

当你缓慢改变电压时，接收端可以清晰地分辨高低电压。但如果你以极快的速度改变电压，接收端观察到的波形可能发生严重失真，以至于无法区分相邻的数字。电话线对于信号频率的宽容度是有限的，我们用带宽描述信道允许通过的频率范围。于是就有了一个核心问题：假设一条信道只能传输某个有限频率范围内的信号，那么我们最多能以多快的速度，向接收者发送可以区分的符号？这中间会经过一串复杂的傅立叶分析，但考试并不会考察。

我们只需要记住奎奈斯特公式并学会应用即可。

$$C = 2B \log_2 M$$

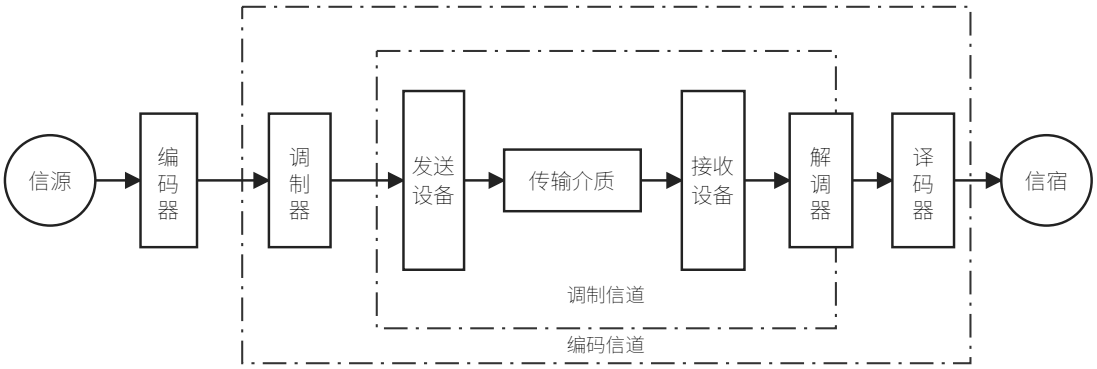
2.1.2b. 香农公式

$$C = B \times \log_2\left(1 + \frac{S}{N}\right)$$

其中 S/N 是性噪比

奈奎斯特定理源于 **香农定理**:

2.1.3. 编码与调制



模拟信源发出的信号称为**模拟基带信号**,数字信源发出的信号被称为**数字基带信号**. 在计算机网络中，发出的通常都是数字基带信号，称为**数字调制**。

基带信号的频谱通常集中在零频附近,并向较高频率延伸,而不是集中在某个远离零频的载波频率附近。

2.1.3a. 单级不归零码

归零: 在脉冲周期的中间时刻回到零电平。其中每个周期在归零时间的占比称为**占空比**，不归零码被称为 **NRZ**，归零码称为 **RZ**

极性:只有正电平就是**单极**，正负电平都有就是双极。

编码信道包含了调制信道。编码的过程是从一组数字信号到另一组数字信号，而不是从 0 和 1 的二进制信息到数字信号。编码是

为什么需要编码呢？因为数字基带信号有时不适合直接在信道中传输，要进行码型变换和波型变换。

对于数字信号，我们把直接在信道中传送基带信号称为**基带传输**.

差分码：又称为**相对码**，与之相对的是**绝对码**，绝对码采用固定的某个电平对应 0 或 1，差分码根据和上一个脉冲是否有跳变来决定，有表示 1，无跳变表示 0 差分的不归零码是 **NRZI**，差分的归零码是 **RZI**

AMI 码:0 仍然是正常的 0 采用正负脉冲交替编码信息 1

2.1.3b. 曼彻斯特编码

双相/曼彻斯特码：曼彻斯特码用信号中间的跳变表示要求发送一个比特，信号都必须在这个比特的正中间发生一次电平跳变，如果是高跳到低表示 1，低跳到高表示 0

2.1.3c. 差分曼彻斯特编码

差分曼彻斯特码: 和差分码一样

差分码靠相比是否有跳变区分，曼彻斯特码靠跳变的方向区分。

曼彻斯特码因为要求跳变，所以其实在一个比特里面需要两个比特的信息，更消耗带宽。

而差分曼彻斯特码用比特开始的时候是否跳变来编码信息。

2.1.3d. 多元码

2.1.3e. 米勒码，

2.1.3f. CMI 码

2.1.3g. nBmB 码

2.1.3h. 调制与解调的概念

调制的根本原因是因为我们我们最终想要的信号本身没法通过既有的信道传播，所以我们先对信号调制后透过信道发送，再让接受方解调。

基带信号是模拟信号就是模拟调制，基带信号是数字信号就是数字调制，而计算机网络采用的数字调制。

在调制中，我们有一个基础的波称为载波，而三种不同的调制方式就是改变载波的参数。

ASK: Amplitude Shift Keying 改变振幅

FSK: Frequency Shift Keying 改变频率

PSK: Phase Shift Keying 改变相位

DPSK:

(todo 对几种 SK 的深度讲解)

QAM 同时改变振幅和相位: 也就是每个符号就包含 4 个相位

基带调制/编码：

带通调制就是把数字信号转换为模拟信号传输

2.1.3i. *QAM* 基本原理

正交幅度调制: 的核心思想是选择能够分解的正交信号

2.1.4. 电路交换，报文交换与分组交换

电路交换在进行数据交换之前，先在子网中建立连接.i 每次传输结束知乎都要拆除电路

报文交换与分组交换都属于存储转发交换。它们的特点是发送的数据一起组合成一个数据单元发送出去。也就是路由器实际上暂存了数据报，路由器可以动态的选择路径。这样效率通常比较高。

存储转发交换和电路交换的区别是它们的直接的核心差别是，是否为通信建立专用的物理存储资源，和是否采用存储转发

报文交换和分组交换的区别是是否切分。如果对传输的数据不做任何限制就是报文交换，否则就是分组交换。

2.1.5. 数据报与虚电路

分组交换又可以往下拆分，分为数据报和虚电路。路由器自由地选择线路来传递数据报，每一个报的路径都可能不同。

虚电路仍然使用存储转发，它虚电路与电路交换的区别是，并没有直接建立物理线路，而是用路由器的存储转发来模拟了线路的感觉。因此，它并不预分配带宽

虚电路在通过链路时通常使用全部带宽。一条链路可能同时被多条虚链路使用。

虚电路分为永久型和交换型

每条链路都有一个标识，称之为 VCID，分组也会标识 VCID，虚电路网络依据是 VCID 进行分组转发。

虚电路按序发送按序接受，但数据报交换按需交换，不一定按序接受。

转发表会记录邻接链路的接续关系

2.2. 传输介质

2.2.1. 双绞线、同轴电缆、光纤与无线传输介质

2.2.1a. 双绞线的分类与特性

双绞线: 由两根相互绝缘，按一定规则绞在一起的铜导线组成。为了抗干扰，可以添加屏蔽层，称为**屏蔽双绞线**。我们在日常生活中，通过网口给电脑接有线网的时候，用的就是双绞线。

当前局域网中常用双绞线。

双绞线还可以分为五类，超五类。

2.2.1b. 同轴电缆

同轴电缆: 顾名思义，它由同轴的两个导体构成，分为内导体和外导体.

电路为了形成回路通常都需要一来一回的形态，所以双绞线需要两根线，而同轴电缆也需要内外导体。

有线电视网络主要使用同轴电缆。双绞线和同轴电缆可以用来传播模拟信号也可以用来传播数字信号。

2.2.1c. 光纤的分类与特性

光纤：有以下优点

- 通信容量特别大
- 传输损耗少，远距离传输经济
- 体积小，重量轻
- 抗雷电点此干扰
- 无串音干扰，保密性好

光纤可以分为**阶跃光纤**和**梯度光纤**。阶跃的路径更近似折线，梯度的路径更近似曲线。

光纤可以分为多模和单模，多模有多种传输路径，单模有一种传输路径。

2.2.1d. 无线传输介质

无线传输

2.2.2. 物理层接口的特性

USB 就属于典型的物理层接口特性

英文的 feature 会更好理解。

- 机械特性：也称**物理特性**指名通信实体间硬件连接接口的机械特定，包括：接线器的形状，尺寸，引脚数目，排列，固定，锁定装置。我们对店员插头的尺寸规范啥的，都属于机械特性。
- 电气特性：在物理连接上，一切有关电路的特性：最大传输速率，电平，输入输出阻抗

- 功能特性：物理接口各条信息好线的通途，包括功能的规定和分类，比如是数据信号线还是控制信号线，定时信号线，接地线
- 规程特性：事件发生的动作序列和合法顺序

2.3. 物理层设备

2.3.1. 中继器

信号经过了很长一段电缆之后，可能会衰减，也可能会混入噪音和失真。中继器往往不是简单放大衰减的信号，而是接受了信号之后再重新发送。

2.3.2. 集线器

集线器的本质是一个多端口的中继器。每个端口都会把收到的信号同时往其他端口转发，但由于只有一条线路，因此任何端口一旦发就占据了整个网络，别的端口发就会冲突，这就是常说的**冲突域**。集线器是典型的半双工网络。

数据链路层

CHAPTER 3

3.1. 数据链路层的功能

3.2. 组帧

3.3. 差错控制

3.3.1. 检错编码

3.3.1a. 汉明距离的概念与意义

比特翻转有自己的代数规律，我们定义第 n 位的比特翻转位 T_i，那么我们有 $T_i^2 = \mathrm{id}$，同时我们还有 $T_i \circ T_j = T_j \circ T_i$

3.3.1b. 奇偶校验码

奇偶校验是最最简单的一种校验方式。而后续的海明校验中往往也包含奇偶校验的思想，不过我对现有教材不满意的点在于，并没有给出足够深层次的 insight，让它更多成为一种想到的技巧，而不是某种结构的必然性。

我们可以把校验的本质看成一种判别式，它是一个接受任意比特流的函数，对每一个比特流返回一个布尔值。分别代码合法和非法。

我们的要求是： 对任意比特串，只要翻转其中任意一个比特，判别结果就必须改变。

我们希望一个合法码翻转了一位之后就一定变成非法码。但是之所以要求非法码翻转后就变成合法码是因为这样利用率最高。

由于翻转之后判别结果必须改变，我们要从一个 n 位比特串集合中选出一个合法集合 ，并且要求任意两个合法码字都不能只相差一个比特。

把全部 2^n 个比特串按照尾比特是否相同，配成如下的 $2^{\{n-1\}}$ 对：

比如 0000 配对 0001

0010 配对 0011

那么我们势必不可能同时把每一对中的两个比特都作为合法的。一对当中只能取一个。所以最多有一半的合法码。

而如果我们让合法码集取最大，那么就一定保证每个非法码都统一对应着一个合法码配对了。

首先存在一种判别法，能够每接受一条比特流信息就分成两类。而对于每条信息，这一条信息所有的比特翻转情况都刚好能被归为另一类。那么异或判定就是其中最简单的函数，一旦进行比特翻转，它的异或判定的结果就一定会变。

借助了判别法之后，我们思考如何构造信息来满足这样的判别法。那就是我们所有发出的信息都要满足该判别式。但是这样会限制我们能发出的信息。

最终我们能得出在这一编码中，一半的信息是正确的，一半的信息是错误的。

所以对于每条任意的信息。我们期望把它编码成正确的信息。

那么这一条信息本身拿到 1 个比特，剩下的位数就能用来表示信息了。

下面我们来证明，这种满足翻转判别式值变化，让合法码数量最大，的判别式本身就是奇偶校验判别，换言之，异或判别。因为任意一个比特串一定可以通过纯 0 比特串翻转得到。

我们如何让已知的信息编码成满足表达式的情况呢，对于一个 2^n 长的的编码，它的合法码情况是 2^{n-1} 也就是任意的 n-1 位比特流它都可以得到双射对应来表达。

但这样的可能情况太多了，我们想要一种简单的构造方式而非复杂的映射，

这理想情况就是在已有的信息流上追加位数，这被称为 Systematic code

再通过追加一位编码，这样是有可能的么。

那就是用一位用作校验位。让校验位始终满足表达式约束。

接下来我们再介绍一种检错编码，它就是循环冗余 CRC 编码。

3.3.2. 纠错编码

3.3.2a. 汉明码

如果我们吧这些信息对应成位置，冗余的比特就能不只用来检错而是纠错。

这次判别式产生的结果不在是布尔值，而是有 n 个结果，其中的第一组结果是不出错的。然后所有出错的结果被拆到余下 n 组里。理论上对于一个 n 位的比特流，只要确保它的各位翻转出现的结果出现在不同的结果组里就行了。

不过这样的判别式的确很多。其中存在一些这样的情况：同一种结果编码，对于不同的比特流，代表的是翻转不同位的结果。比如说 111 是合法的，000 也是合法的，然后 110 和 010 的判定结果是是一样的，但是一个是翻转了第 3 位，一个是翻转了第 2 位，这非常不优雅。

我们期望最终的结果恰恰能刚好对应翻转位，无论原来的码字是什么，只要翻转的是同一个位置，判别结果就会按照同一种规律变化。其中一种结果用来对应无错误，剩下的 n-1 种结果都用来意一一对应表示错误的 n-1 个位置。

比如，我们定义一个 7 位码的判别式有 8 种输出，值域是{0,..,7}，其中 0 代表无错误，剩下的输出是几就代表第几位被翻转了。

转化成数学表达就是上

$$f(x) = f(y) \quad \Rightarrow \quad f(T_i(x)) = f(T_i(y))$$

为了方便起见，我们记 $g_i = f \circ T_i$

接下来我们来研究任意多次翻转。它最终一定能够写成多个翻转复合的形式，也就是 $T_i \circ \cdots \circ T_j$

我们考虑第二个任意翻转，也就是 $T_k \circ ... \circ T_l$

这两个翻转的复合的结果就是，它们之间如果有重复翻转的归 0，其他的翻转复合保留。那么这不就是模 2 加法，换言之，异或么。

我们定义全 0 比特是合法码，那么我们可以证明。

如果 x 是合法码，y 是合法码，那么 $x \oplus y$ 也是合法码。 换言之，最终合法码是一个对异或封闭的集合。

接下来我们发现比特翻转的闭包刚好构成阿贝尔群。我们记所有的翻转行为和 id 翻转都是这个群上的元素。封闭性和结合性交换性单位元都是容易证明的，每一个元素的逆元就是它自身。

甚至它还是个向量空间，我们添加 {0,1} 作为标量域，它数乘的结果。0 倍翻转操作，表示不执行这个操作；1 倍翻转操作，表示执行一次这个操作。分配率显然满足了。

我们把翻转第 i 位记位 T_i 会发现一种情况，那就是

n 个纠错比特本质上提供了 2^n 的信息，

对于一个比特流，我们想额外

我们继续发展奇偶校验的思想，怎么保证任意一条信息刚好被编码成了

假设我们发送 7 个比特中有三个冗余比特，那么 $2^3 = 8 \geq 7 + 1$，我们就能确保存在一种编码使得纠错码的每个具体形态对应着一种纠错，同时剩下一种无错情况也能被表示。

你可能会问，在这一情况下纠错码如何发现自己的错误的。很简单，因为经过良好的设计，理论上翻转了某一个象征着某一位有错的编码形成的编码样态能够象征自己的编码。

3.4. 流量控制与可靠传输机制

3.4.1. 流量控制、可靠传输

流量控制是指接受方控制发送方的速率,别一口气传太多接受不过来,让接受方有足够的缓存空间接受每一帧。

可靠传输是为了避免有些丢失或出现错误，要求重传。可靠传输方案着眼于重传。对于链路层来说,可靠传输并不是必需的,因为总是可以把任务交给上层,不过在底层不够可靠的情况下提供可靠传输可以提高效率。，有线的局域网，也就是接下来介绍的以太网，并不要求可靠传输，而无线的局域网就会实现可靠传输。

3.4.2. 停止-等待协议

发送方每次只允许方式一帧，接受方每次成功接受一帧后，返回一个确认帧。

复制王道的图

3.4.2a. 停-等协议利用率计算

我们可以计算得到停止-等待协议的利用率 U_{sender} 为

$$U_{\text{Sender}} = \frac{t_{\text{Seg}}}{t_{\text{Seg}} + \text{RTT} + t_{\text{ACK}}}$$

3.4.3. 滑动窗口机制

3.4.3a. 滑动窗口协议工作原理

滑动窗口机制: 在任意时刻，发送方维持一组连续的帧序号，称之为发送窗口。接受方也维持一组帧序号，称接受窗口。因此每次确认都是确认一组，没收到确认的情况下还可以发几帧。

k 比特帧序号需要和发送窗口和接受窗口满足如下关系

3.4.3b. 滑动窗口协议信道利用率计算

3.4.3c. 滑动窗口协议大小与分组序号字段比特位数间的约束关系

3.4.4. 后退 N 帧协议*(GBN)*

停止等待协议的利用率很低,在大多数时间内,我们都在等待。而滑动窗口的好处就在于,允许我们等待的时候继续发。

接受方的窗口长度为 1，对于确认，接受方可以对每一个包逐一发送确认，也可以累积确认。但总而言之，只要确认了，就说明之前的全都被确认了。
同时我们可以讲一下公式:

$$U = \frac{NT_D}{T_A + \mathrm{RTT} + T_D}$$

当我们发送出一帧之后，在等待这一帧回来的时间中，要让信道利用率最高，

最理想的情况就是 N 足够大，我们一直在发；但是当 N 不够大的时候，我们的帧长越大，我们发送的帧就越少。

我们要尽可能的用慢我们的带宽，也就是发送速率。当数据帧很大的时候，我们的往返时间就一直在尽可能的多花时间在处理带宽上。

左边的发送窗口

GBN 协议的窗口固定为 1，严格按序接受。

假设我们的缓存窗口仍然是 1，那么我们就没办法

如果某一帧超时或者出错，发送方不但需要重传这一帧，还要重传之后所有发送了但还没被确认的帧。

那么在选择重传协议下，假设发送窗口发了 5 个包 0-4，接受窗口存了 3 个包，我们此时 ACK0

$$W_s \leq 2^k - 1$$

这是因为序号是循环使用的。假设发送窗口发了 8 个包，8 个包全部丢失了，我们不得不 ACK0 来表示需要接受第一个包，但因为序号循环使用。发送方会以为全部收到了。

3.4.4a. 选择重传协议*(SR)*

选择重传协议需要逐一确认每个包。

选择重传协议

3.5. 介质访问控制

3.5.1. 信道划分

多路复用问题研究的是如何把一个信道分配给多个人。这个问题适用于集线器或者无线网络情况。

多路访问可以划分为 3 种类

在电磁波中，不同的数据最终会叠加在一起。

3.5.1a. 频分多路复用

FDM，Frequency Division Multiplexing：多个人分别用不同的频率。往往是把本来相同的频率通过调频实现，在模拟通信中，频分多路复用是非常方便的。但是频分多路复用会引起各路信号间的串绕。非线性的传染可以用带通滤波器消除，但线性和就没那么好办的，因此对线性的要求高。

3.5.1b. 时分多路复用

TDM，T 代表 Time. 把时域划为多个等长的时隙。每路占用不同的实隙，在时域上不重叠。

时分多路复用可以分为同步 STDM 和异步 ATDM 两种。同步时分多路复用把第一个时隙给第一个信号，第二个时隙给第二个信号。以此类推。不过这样当每一路信号没有东西发送的时候它也依然会占用着时隙。

在异步时分多路复用的情况下，我们会把数据交给复用器。复用器内部有缓存区，然后按需挨个时隙发包。

不过同步多路复用因为知道每个时隙对应谁，所以不需要在传输中包含来源信息，但是异步多路复用就需要了。

3.5.1c. 波分多路复用

WDM，实际是一种频分多路复用。只是在光纤通讯中光的频率太高，用波长替代频率讨论了。

光纤的远距离传播中往往采用波分多路技术和光纤放大器来实现超大容量传输。

还有 DWDM 这里的 D 是 Dense，能够在一根光纤中复用 100 路以上的的光波信号。

3.5.1d. 码分多路复用

码分多路复用的核心是利用正交性，这对于学过傅里叶级数的并不陌生。

每发送一个比特，我们都把它转换成一个码片。

又称码分多址，Code Division Multiple Access. 当然，其实时间和

码分多用的优势涉及非常复杂的技术细节，不必了解。考纲只需要知道概念和基础原理会做题就行。

3.5.2. 随机访问

我们需要知道，无论是 ALOHA 还是 CSMA，都只是对一类协议的描述，而不是某种具体的协议，或者说，本质上只是一类介质访问控制策略的思想方法

3.5.2a. ALOHA 协议

ALOHA 是夏威夷岛上的人彼此打招呼的时候用的词。而这个协议在 1970 年由夏威夷大学首次发明。

ALOHA 协议最早是为了在不同的终端连接夏威夷岛上的打字机。终端有这样的特征：是突发的数据段，在绝大多数时间内它都是静默的。这和电话这种持久的数据流不太一样。如果使用分时系统那么就会效率特别低。

ALOHA 协议的帧是定长的，大家使用同一载波竞争接入信道，也就是说只要有两方的无线电波在时间段上有一点重叠，信息就会被破坏。

ALOHA 协议啥都不管，先直接发送，发送完了等一会儿来确认信息有没有被收到，等多久呢，等消息到远端然后返回的时间，如果没收到，就说明信息被破坏了。那么说明有人在跟你有无线电波的重叠。那么我们就随机退避任意的时间重发。

在随机接入协议中，一个节时隙 ALOH 点以全速发送，如果碰撞了就重发。为了避免所有设备都立即随机重发，所以会以随机速度重发。

3.5.2b. 时隙 *ALOHA* 协议

时隙 ALOHA 的思路是，减少碰撞区间。既然纯 ALOHA 中的碰撞来自数据帧在时间上的任意重叠，那我们能不能通过限制发送的起始时间，让数据帧只能完全重叠或者完全不重叠？也就是划分时隙。

不过时隙 ALOHA 并不能让人完全满意，我们认为 ALOHA 协议的根本问题是不管信道是否空闲都发送，这大大增加了冲突的可能。而 CSMA 让我们在发消息之前先监听有没有人.

3.5.2c. CSMA 协议

CSMA 协议可以分为以下三种：

- 非坚持 CSMA:：如果信道空闲，就直接发，如果信道时间忙，等待一段时间重新侦听
- 1 坚持 CSMA：与非坚持的区别是，如果信道忙，会一直侦听到能发出去为止
- P-坚持 CSMA： 先划分时隙，如果信道闲了，以概率 p 立刻在时隙开始发送，以概率 1-p 在下一个时隙重新进行此过程；如果忙就等待一个时隙
  值得注意的是，如果单单是 CSMA 并不知道自己是否碰撞了没有，它的碰撞可能要依赖 ACK 和超时重传。

3.5.2d. CSMA/CD 协议

CD 意为 collision detection 是说碰撞检测，它在发完了之后能够凭据物理手段，特别是有线信道中的电压等等明显知道自己碰撞没有。如果碰撞了就立刻停止别发了，都等待一段时间。

在以太网中有时我们还会发送强化干扰信号，这是为了让碰撞更明显，帮助别人更容易依赖物理手段检测到。

CSMA/CD 是以太网采用的协议

3.5.2e. *CSMA/CA* 协议

CSMA/CA 是无线局域网采用的协议。在无线情况下，检测碰撞是很困难的，因为信号的自干扰，还关于隐藏站点问题：假设 A 和 B 发送的信息在 AP 处碰撞，但是 A 可能观察不到 B 的无线电波，所以它始终没法获得关于全局的信息，等等，所以我们需要另一种方法。

然后我们这样想，虽然 A 没法观察到 B，但或许 AP 能够帮助我们把消息告诉 A 呢？这个概念就是 RTS 和 CTS。在尝试给别人发送消息之前，我们总是先发送一个 RTS。

RTS 和 CTS 都是控制帧，RTS 意味着 Request to send，CTS 则是 Clear to send，假设 A 和 B 都需要给 AP 发消息，那么 AP 允许了 B 发送，但是 CTS 也会传给 A，叫 A 不要跟着再争了。

值得注意的是，普通的数据帧通常是比较长的，更容易冲突，而 RTS 和 CTS 的帧是很短的，即便冲突了，只要随机避退就行，也不会特别浪费。

CTS 还包含了一个时间段，这个时间段称为 NAV，Network Access Vector，意思是说我们这段信息会独占多久，让即使出现很小的空闲别人也别强发，会造成冲突。

而抢发中其中一种格外常见的情况是信息需要回复。我们不希望打断这一消息，于是制定了一个优先级机制。这时候我们的核心思想是：如果某个站点正在回复上一帧，那么它应该比其他准备发起新通信的站点更早获得发送机会。因此有些帧的帧间间隔比较短是 SIFS，有些比较长是 DIFS。事实上我们会知道现在的 Wifi 系统会有更多更复杂的优先级，但我们只要知道 SIFS 和 DIFS 就行了。SIFS 和 S 代表 Short，DIFS 的 D 代表 Distributed， IFS 是 Inter-Frame Space，帧间间隔。这个帧间间隔的长度。我们收到 CTS 的时候，只需要等待 SIFS，但是我们发 RTS，却要等待 DIFS。

3.5.3. 轮询访问:令牌传递协议

3.6. 局域网

3.6.1. 局域网的基本概念与体系结构

3.6.1a. 冲突域与广播域的概念

一个最简单的局域网就是把每个电脑都通过网线连接起来。一条电脑发出的信息别人都能接受到。

广播是局域网的核心。甚至可以认为，局域网的本质就应该是广播的网络单元。为什么需要广播呢？**因为我们知道发现网络里有谁**。要么有一个中心化的索引记录着网络中的索引（这不就是路由器么），而去中心化的方式就只有广播。

3.6.1b. *MAC* 地址

链路层地址有不同的叫法，比如物理地址 比如 LAN 地址和 MAC 地址。

如果整个网络里只有点对点的网络，那么显然是不需要地址的。但这会让我们需要 $\frac{n(n-1)}{2}$ 条线路，更别说无线通讯从根本上就不是点对点的。只要有中间转发，那么帧中就必须包含目的地址。

3.6.2. 以太网与 *IEEE 802.3*

3.6.2a. 以太网

以太网诞生于七十年代施乐的实验室。严格意义上讲,以太网应该指的是 DEC,Intel 和施乐推出的以太网 DIX Ethernet v2。

1983 年的 IEEE 802.3 建立了一种接近局域网规格。严格来讲，它不应该称为以太网。但实际中我们常常混为一谈。它们的帧格式有一些微小变化，但是可以互操作。

今天，市面上实际采用的标准都是 DIX Ethernet v2

3.6.2b. 以太网 *CSMA/CD*

经典的以太网是采用总线型的同轴电缆连接的，称为 10 Base-5，一个冲突域里面最多有 5 个网段。网段之间用中继器连接。采用 CSMA/CD 协议搭配指数退避算法。

3.6.2c. 以太网指数退避算法

在冲突域内两个相距最远的主机的端到端传播时延的两倍被称为争用期。

比如这是我们第一次冲突，那么我们退避{0,1}倍时间中随机。

这是第二次冲突，退避{0,1,2,3}倍时间中随机。

第三次冲突，退避{0,1,2,3,4,5,6,7}倍时间中随机，

冲突 10 倍之后，但是最多也只能退避 1023 倍时间中随机取。

如果 16 次之后仍然冲突，放弃传输报告上层。

3.6.2d. MAC 地址

MAC 地址有 48 位，仅仅取决于网卡，是全球唯一的，靠 IEEE 来分配管理，网卡生产商需要经过授权。它表示为 12 个十六进制数字，用连字符或者冒号分割。高 24 位是厂商代码，低 24 位厂商自行分配。

FF:FF:FF:FF:FF:FF 是广播地址。

MAC 地址的整套设计都只是为了一件事：防止链路层的身份地址重复。

网络适配器有多种不同的速率，比如 10Gbps 的和 1Gbps 的 100Mbps 的等等，它们会在网线连上的那一刻交换。

3.6.2e. 以太网帧结构

以太网帧采用**曼彻斯特编码**：

- 前同步码：8 字节。前七个字节都是 10101010，用于同步时钟。原因是不同适配器的时钟保持出厂时就一致是不现实的，它们的时钟速率总有些偏差。
- 目的地址：6 字节的 MAC 地址
- 源地址：同上
- 类型字段: 2 字节，决定你是 ARP/IPv4/IPv6
- 数据
- CRC: 4 字节
- 前同步码

一个好问题是为什么要带有类型字段。我们会发现，在互联网的更底层的包结构里。因为上一层有多种协议，如果你直接扔给上层。上层的数据就会不知道怎么处理。当然，你可以在上层设置一个区别不同协议的头部。请记住，这个头部区别于每个协议自己的头部。这个头部其实可以归到下一层的协议的头部里。

以太网帧的前同步码是硬件生成的，也不受 CRC 的保护，在未经特殊说明的情况下，**当提到以太网帧，我们总是不包含前同步码**.

以太网的适配器包含一种特殊的模式，称之为混杂模式，这种情况下它会接收所有可以接受到的帧，又称为绣探

3.6.2f. 以太网标准

10Base-5 半双工 **10Base-T**: 代替同轴电缆的经典以太网，采用两根双绞线，从而实现了全双工，速度为 10Mb/s。采用集线器来连接多个主机.

快速以太网: **100Base-T**

以太网提供的是**不可靠服务**，不可靠就是不是可靠传输，没有后退 N 帧或者选择重传协议。

3.6.3. IEEE 802.11 无线局域网

3.6.4. VLAN 基本概念与基本原理

局域网的本质是广播的最小单元。但局域网之内的设备一旦大到一个量级，广播就会带来巨大的负担。所以我们希望能够把局域网切的小一点。

同一个局域网之内是没有流量隔离的。但或许你会想要隔离流量，就不得不需要多个局域网。但多个局域网就会给每一个局域网分配一个交换机。但其实很多交换机的接口都没有用满。

VLAN 技术能够让你用更少的交换机去隔离出更多的局域网。

支持 VLAN 的交换机在里面增加了流量隔离功能，交换机会记住哪个属于哪个 VLAN。

那么对于不同 VLAN 之间的互通呢。比如说一个 VLAN 在两个交换机上各有一部分怎么办。一个思路是靠路由器。但这个就不在链路层了。

另一个思路就是直接用线吧应该属于同一个 VLAN 的给连接起来。当然这样也有不方便的地方。假设你有很多个 VLAN 都分布在不同的交换机上就要拉很多线。一个更好的解决方案是**干线连接**. 每台交换机上有一个特地配置的端口叫干线配置端口。发送到交换机之后会在帧加上一个 VLAN tag，然后到达别的交换机之后 tag 会删掉。

3.7. 广域网

3.7.1. 广域网的基本概念

广域网通常跨城市，甚至国家和洲。广域网并不是因特/互联网，相反，因特/互联网可以由广域网组成。

- 广域网是一种公共数据网络，通常由大型运营商经营
- 广域网研究的重点是宽度核心交换技术

3.7.2. *PPP* 协议

当我们拨号上网的时候,就需要点对点协议,因为此时我们要和路由器建立起一些关于链路层协议的配置

- 封装成帧的方法
- 链路控制协议
- 网络控制协议 NCP，用于对接不同的网络层协议
  PPP 帧的结构如下：
  标志字段：首尾都是 0x7E 作为定界符 地址和控制字段：无实际作用，甚至可以通过协商

3.8. 数据链路层设备

只要数据帧传输在端到端的链路连接，那么就是在一个局域网内。我们接下来思考的问题是，怎么让局域网越来越大。

在经典以太网中，我们可以通过集线器吧大家疯狂连接起来，但是这时我们遇到了物理层的限制，就是冲突域会变得越来越大，就越来越容易冲突。

我们很容易想到，对于两个小的以太网网，我们能不能把小的以太网网连接成大的局域网呢，这样在小的冲突域内的消息不会和另一个冲突域上的消息冲突，冲突的概率就低了很多。这一概念就是网桥。

我们考虑 ABCD 四台机器的情况下。 平时 AB 发消息，CD 就不能发。对于一个局域网内部，所有的信息还是照常广播，但凡是局域网外的消息，网桥看这个 MAC 地址，把它转发过去。但 A 在往 B 发的时候 D 能往 C 发而不至于冲突了。

3.8.1. 以太网交换机

交换机的本质就是多端口的网桥。

假设所有的主机都只通过交换机而不通过集线器连接。那么传统的冲突就不会发生，这被称为**交换局域网**.。

当然，除了消除冲突之外，交换机还有一个好处，那就是支持**异质链路**. 不同速率的以太网可以被这样连成一个以太网。请注意异质链路和异构链路的区别，快速以太网和吉比特以太网是异质链路，以太网和无限局域网是异构链路。

冲突域和广播域的区分是真题考察过的考点，我们在此强调一下。冲突域是基于物理层定义的，因此物理层上的设备没法隔离冲突域。广播域是基于链路层定义的，因此链路层设备也没法隔离广播域。

广播风暴指的是局域网大量帧持续在回路里传播，使得局域网的带宽被耗尽的情况。成因可以是大量产生异常帧，也可以是环路。只要还在同一个广播域，我们就没有真正的抑制广播风暴。

为了避免形成环形回路，交换机会增加使用 BTP 功能. 抑制广播风暴的办法是避免环路拓扑，但交换机和网桥本身都无法抑制广播风暴。想要抑制广播风暴，需要把广播域划分的更小，也就是引入更高层的设备，也就是网络层的路由器。

指的区分的是冲突域和广播域的概念。冲突域是物理层层面的域，也就是链路层设备可以隔开它们。但物理层设备不能。广播域是链路层层面的域。一个局域网的本质就是广播的单元。

3.8.2. 交换机的工作原理

假设我们有两台主机连接一个交换机，初始情况下交换表是空的。

A 向 B 发生一帧，经过交换机，这一帧包含了来源的 MAC 地址，那么我们可以知道，这个 MAC 地址一定对应着 A 端口，以后有人发给 A，就直接往 A 端口传。

交换表里没有主机 B 对应的 MAC 地址，所以要靠泛洪广播确定 B 能收到。

B 能收到了之后会往 A 回传，那么此时交换机就知道了哪个端口对应 B 主机的地址了。

交换机还有老化机制，可以避免交换机的表存入过多历史设备。当出现集线器背后的拓扑变更，适配器变换等等，搭配老化机制和重新学习，交换机就能自然而然的更新表。

而对于我们拔下线插到另一个端口的情况，交换机往往还有链路断开检测，对于断开的端口自动清空表项。`

网络层

Chapter 4

4.1. 网络层的功能

网络层在 TCP/IP 标准中被称为 Internet Layer

在 OSI 基本模型中，对网络层的定义是：网络层提供必要的功能和规程，使传输层实体之间能够进行无连接方式或面向连接方式的传输。层

根据 TCP/IP 标准，明确指出网络层应该只有一种协议 IP 协议，将源主机传送至目的主机。

「网络层使传输层实体不必考虑底层的路由选择和中继转发问题」

4.1.1. 异构网络互联

我们在今天的网络体系中把链路层作为广播域。不过早期

异构指的是通讯技术和运行协议的不同，在数据链路层，我们有以太网，有无线网，它们之间就是异构的。

要想让异构网络互联，基本策略是协议转换和**构建虚拟互联网络**.

假设三个网络 A 和 B 和 C 互联，协议转换的思路是这样的。

1. A 网络发出符合协议 A 的分组。
2. 中间设备将协议 A 的分组转换为协议 B 的分组
3. 下一个中间设备再将协议 B 转换为协议 C。
4. 最终由 C 网络接收。

这时，各个网络仍然遵循各自的协议，中间设备负责解决它们之间的不兼容问题。

而虚拟互联网络的概念就是建立一个更高一层的抽象协议，在逻辑上把它们构成一个互联的网络。

1974 年，RFC 675 在阿帕网中首次发明了**网关**这个概念

一个数据包需要从网络 A 到达网络 B。会经历以下过程。

- 网络 A 将数据包交给连接两个网络的网关。
- 网关移除网络 A 的封装，检查互联网数据包。
- 网关决定下一步经过哪个网络。
- 网关重新封装数据包，使其适应网络 B，再进行转发

到了 90 年代之后，网关通常被称为路由器。

而这个更高层面的协议也就是 IP 协议。

当然，实际上它们并不是真的互联的，中间有路由器，但是高层的协议通过屏蔽掉了这些。

网络层分割了广播域，把多个链路层网络连接到一起。

4.1.2. 路由与转发

显然网关需要承担以下职责：

转发: 通过一条链路收到分组后，向哪条链路发送出去

路由: 当分组从源主机流向目的主机的时候，需要通过某种方法决定分组经过的路径，称之为路由算法。路由器通过路由表来实现路由选择

从 80 年代开始，逐渐兴起路由器这一用法。

4.1.3. *SDN* 基本概念

Software Defined Network

SDN 把网络层分为两个平面，控制层面和数据层面

4.2. 路由算法

4.2.1. 静态路由与动态路由

静态路由算法通常由人工进行网络配置，配置好了网络就不动了

动态路由算法能够在网络状态改变的时候自动计算最佳路由

4.2.2. 距离-向量路由算法

这是一种异步的，迭代式的分布式路由算法。分布式意味着没有节点掌握全局信息。

4.2.3. 链路状态路由算法

这是一种全局式路由算法。全局意味着至少有一个节点知道全局的信息。每个路由器在构建时需要构建出整个网络的拓扑图，为了构建拓扑图，周期性的检测和收集费用。然后变成在网络拓扑图上求最短路径问题。使用 Dijkstra 算法。

4.2.4. 层次路由

4.3. IPv4

IP 协议提供的是一种尽力而为的服务，不提供可靠传输。这是因为让路由器提供差错控制会大大提高路由器的复杂度。况且即使 IP 协议自身是可靠的，并不能替代端到端的检查。因为可能在传到上层的情况出错。

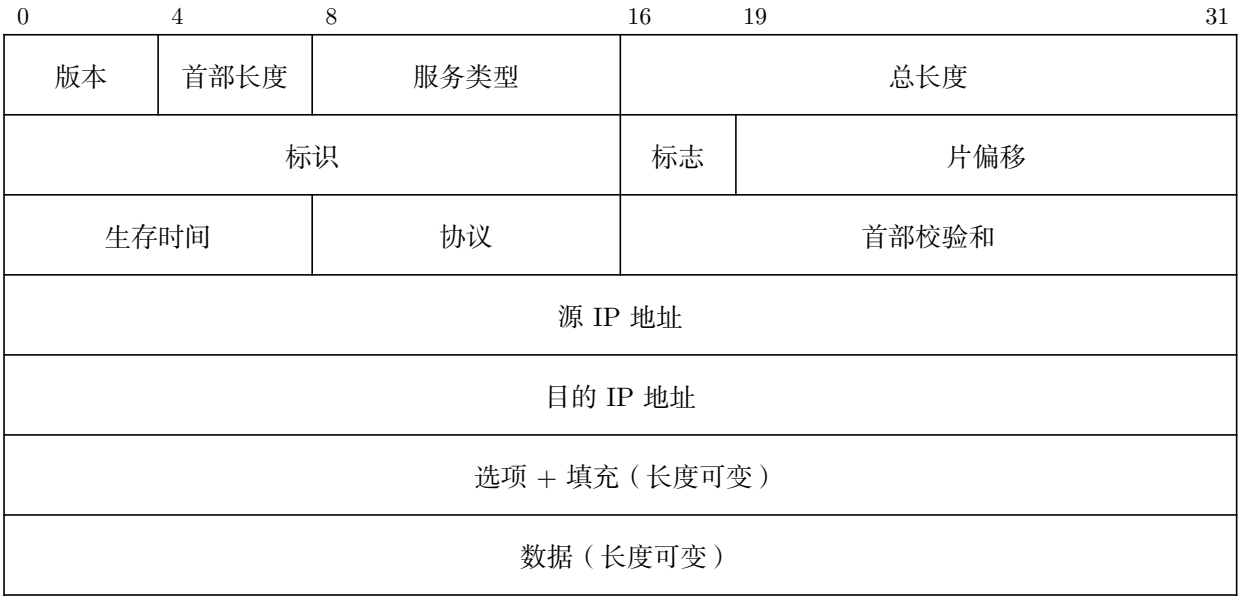
IP 协议是一种采用数据报的网络协议，而它最终成为了网络层唯一主流协议的原因也恰恰是这种简单性。

IP 地址诞生于 1980 年的 RFC760，它当时主要是为了把几个已有的链路层交换网络连接起来。

4.3.1. IPv4 分组

有时我们也说 IP 数据报，分组和数据报是一样的。分组是网络传输的数据单位，而数据报通常用于无连接的，独立寻址和转发的。而 IPv4 就是无连接的，我们在之后的传输层会提到。

IPv4 分组的布局和 MAC 地址的布局一样，同样不需要硬性记忆，只需要知道每一块的实际意义即可



选项与填充、数据部分均为长度示意，不固定占一行；无选项时首部为 20 字节。

首部，其中前 20 位是固定的，剩下的是可选的：

- 版本 4 位: 决定 IP 版本号，IPv4 值就为 4，IPv6 值就为 6, 首部长度 4 位: 前面已经提到 IP 地址的首部长度是可变的，此处的单位是 4 字节，那么 4 位最大能表示的是 15，也就对应着 60 字节。

- 区分服务 1 字节: 通常不用
- 总长度 2 字节，也就是 16 位，单位是字节，也就是说整个数据报最长可以是 65535 字节，减去我们的 20 位首部长度，最多可以封装 65515 字节的数据
- 标识 2 字节: 通过计数器机制产生，每发一个标识+1，作为每个数据报的标识
- 标志 3 位：最高位保留，剩下两位分别是 DF 和 MF，DF 是禁止分片标志，Don't Fragment，MF 是 More Fragments 标志。DF=0 标识路由器允许把数据报分片，DF=1 代表不允许，MF=0 代表未被分片或者已分片数据的最后一篇，MF=1 表示后面还有分片

4.3.1a. *IP* 数据报的分片

你或许会好奇，什么是分片？我们都知道 IP 协议运行在链路层协议上方，但链路层能承担的数据最大长度是有限的，链路层协议所能承载的最大数据量是 MTU，Maximum Transimission Unit 有时没法完整的承担整个 IP 数据报，我们就必须把 IP 数据报切割再拼凑起来。

如果路由器不允许分片，也就是 DF=1，那么就只能把它丢弃了。

- 片偏移 13 位
- 生存时间 1 字节
- 协议：值为 6 表示 ICP，值为 17 表示 UDP，值为 1 表示 ICMP

4.3.2. *IPv4* 地址与 *NAT*

4.3.2a. 特殊 *IP* 地址

IP 地址有以下特殊地址：

0.0.0.0: 仅仅用于本机尚未获得 IP 地址时的源地址。255.255.255.255 也就是全 1，**有限广播**地址，路由器接受到这个地址会把它发送给整个能直接连接到的局域网。换言之，有限广播就是在链路层广播，即使这个链路层包含多个 IP 子网。

一个 IP 地址由 32 位二进制构成，

一个主机不是只有一个 IP 地址，**IP 地址是与网络接口关联的，而不是和主机关联的**。

IP 地址除了表示一个物理连接的单播地址外，还有：

- 组播地址：于第 6 节介绍
- 网络地址：

4.3.2b. *NAT* 的原理

在目前 IPv4 紧张的情况下。我们没法为连接到互联网的主机分配到一个 IP，于是我们的想法是能否让多个同一出口的设备共享一个公网 IP。我们规定让家里多个设备用可能重复的 IP，这些 IP 是针对内网设备和路由器而言的，外网不认识。路由器出口最终会把修改包变成共同的地址。这个概念就是 **NAT**

外网把包回传的时候其实是往你的路由器的公网 IP 传的。那么问题来了，路由器怎么知道哪个包给发给内网的哪台机器呢，毕竟这些包的目的 IP 都是同一个公网 IP。

其实仅就网络层而言，这个问题实质上是无解的。但是我们在更高层的传输层。在更高的传输层我们通过不同应用的端口来区分，这被称为 NAPT 或者 PAT，其中的 P 就是 Port。

假设两台主机分别通过 50000 端口通信。路由器通过修改传输层的包，把端口号分别改成 40000 和 40001，IP 地址还是改成路由器的地址。

但显然端口映射是有问题的，一句话，其根本总结就是把内网的多个主机在外部看来逻辑上变成了一个主机。但是真正的主机来源信息留存在内网里，那么路由器永远只能通过内网主动发出去的消息建立映射。假如外网想要访问内网主机就没办法了。

反过来的问题，如果不是我们的电脑主动访问服务器，而是外面的电脑要访问内网的主机。

4.3.3. 子网划分、路由聚集、子网掩码与 *CIDR*

给网络中主机的分配地址时，你当然可以随机分配，但是这样任何一个路由器路由器就要记住每一个主机的分配。这很没必要。

最早的网络层协议不就是为了把多个协议不同的异构网络连接起来么，那么每一个链路层网络只需要配备一个路由器。

在 1981 年的 RFC791 中，我们提出了 ABC 三类地址，想到了利用前缀划分网络。

子网划分的本质是通过 IP 前缀来划定逻辑分组。每个地址都被分为网络号和主机号这两部分。网络号指代一个 IP 地址的前缀部分，满足这个前缀就属于这个网络。剩下的部分就是主机号。

好处在于划定路由边界，从而使得一个路由器不用记住每一个 IP 地址所属的端口，而只要发往对应的网络，让对应的路由器来处理。

有的网络需要的主机数量多，有的需要的主机数量少。分配一样长度的前缀就会占用一样的地址空间。显然没必要。所以最早互联网的想法是分为 ABC 三类，前缀分别是 8 位 16 位 24 位。

为了避免一个 IP 地址有多种前缀方式，我们采用哈夫曼编码。A 类前缀是 0，B 类是 10，C 类是 110.

IP 协议诞生于 1981 年，当时全球只有两百台登记 IP 的主机。

你看这个设计可能会觉得很奇怪，

4.3.3a. 子网划分

子网划分机制诞生于 RFC 950，它发现我们一个大的组织里面可能有多个小的组织，

对于像大学一样的场景，你当然可以用一个局域网，但这样会存在一些问题。

- 它们之间可能是异构链路：有的使用以太网，有的使用无线网。
- 广播域太大：我们在数据链路层设备那一节层级提到过广播风暴的问题

所以我们希望能在网络层简历分层的子网概念。一个组织网络内可以有多个子网。

你当然可以给每一个子网一个独立的网络号，这看上去似乎没啥问题。不过会增大路由表规模。

这时，你或许回想到，为啥不干脆分层呢。

不过在我们今天来看

因此一个网络可以是另一个网络的子网。

把大的网络划分成小的网络被称为**子网化**.

但是这样 A 类 B 类 C 类很僵死，我们于是相到了一个办法，那就是子网化。

而逆向的，把多个统一前缀小的网络作为整体利用起来，被称为**超网化**

子网掩码当然可以用一个长度来替代。

你可能会觉得这样是不是太死了，为啥偏偏是 8 16 和 24 呢，你说的对，所以我们有了 CIDR

127.xxx.xxx.xxx 是一个特殊的子网，它被保留用作网络软件测试和本地机器的进程间通信，实际的网络上绝不允许出现这个网络中的地址。被称为**回送地址**。

10.xxx.xxx.xxx 和 192.168.xxx.xxx 则是专门用于内网的**本地地址**或者**私有地址**，也就是不和 Internet 相连的网络，一旦相连就要靠 NAT 或者重新获取 IP.

IP 协议中除了发送到单个主机的**单播**，还有**广播**和**组播**，在第 6 节中我们会介绍组播。

广播是好理解的，我们已经知道了有限广播就是在链路层广播。除了有限广播，我们还想介绍**直接广播**.

如果目的地址的主机号是全 1 的，那么这个包将被直接广播。路由器会向网络上的所有主机转播它。

划分好子网知乎。子网划分之后，我们要介绍一些特殊的 IP 地址，这些地址不能被直接用来代表单个的物理连接。

4.3.3b. 路由聚合

4.3.4. ARP 协议、*DHCP* 协议与 *ICMP* 协议

4.3.4a. ARP 协议

ARP 协议用来勾连以太网和 IP 网：当我们已经知道下一跳在本地以太网上，知道了对方的 IP 地址之后，我们需要知道对方的 MAC 地址。严格来讲，ARP 更像一个链路层协议。

每台主机上会有一个专用的内存区域称为 ARP 缓存，或者 ARP 表。它会存储所在的局域网内其他主机和路由器的 IP 地址和 MAC 的映射关系。这个映射表会经常更新。

如果这个表中有，直接就能查到。如果这个表中没有，会发布一个链路层广播，其中包含了我们想要找的 IP。然后持有这个 IP 的主机会回应。

4.3.4b. *DHCP* 协议

当一个组织获得一块地址块后，就需要为每台主机决定一个 IP 地址，当然，我们总是可以让管理员给每个主机和路由器手动指定 IP 地址，但 DHCP 协议提供了一种自动的方式：我们会在子网内运行 DHCP 服务器，用户在电脑上通过运行 DHCP 客户端获得 IP 地址。

后续 RFC3074 通过 MAC 地址划分避免多重响应啥的，我在这里只是提一下，考研显然不涉及 RFC3074 啥的。

通常来说我们买的路由器里面就包含了 DHCP 服务器的功能。

DHCP 是应用层协议，使用 UDP，服务器的端口是 67，客户端的端口是 68

DHCP 的过程分为四步，可以记忆为 DORA: Dicover，Offer ，Requset，ACK

Discover：一台计算机刚连接到网络时，首先需要知道 DHCP 服务器在哪里。虽然这个时候主机还没获得 IP 地址，所以我们发送一个特殊的广播数据报，源地址是 0.0.0.0。

Offer: DHCP 服务器会报告自己有哪些地址池可分配。

Request：主机通过 Request 决定接受某个 DHCP 服务器的 offer

ACK：服务器确认

DHCP 分配的每个地址都是有租期的，同时还会把分配子网掩码和默认网关等信息，本地 DNS 的 IP 等信息。

在 DHCP 的标准中，设计者决定要求应当允许子网内有多台 DHCP 服务器，并且还允许这些服务器使用重叠的地址尺。这一决定或许一定程度上是由于 DHCP 衍生于古老的 BOOTP 协议。Anyway，IPv6 通过自带地址分配，终极地解决了这个问题。

4.3.4c. *ICMP* 协议

IP 协议被设计的特别简单，尽力而为，可是这也意味着如果 IP 协议无法送达，IP 协议本身是不能帮助我们知道出现了什么问题了，因此我们需要 ICMP 协议。

不过 ICMP 与其说是独立协议，ICMP 是一种特殊的 IP 数据报。

举一个例子，主机 A 向主机 B 发送一个 IP 数据报。路由器 R 收到这个数据报，查找路由表，发现自己根本不知道如何到达 B 所在的网络。按照 IP 的尽力而为原则，R 完全可以直接丢弃这个数据报。但问题来了：A 根本不知道这个数据报为什么没有到达 B。

所有符合 IPv4 标准的路由器的必须支持 ICMP。它希望能报告一些特定的信息：其中包括五种：

- 终点不可达

- 源点抑制
- 时间超时
- 参数问题
- 路由重定向

4.4. IPv6

4.4.1. IPv6 的主要特点

IPv6 解决了解决的首个问题是 IPv4 地址空间的耗尽问题

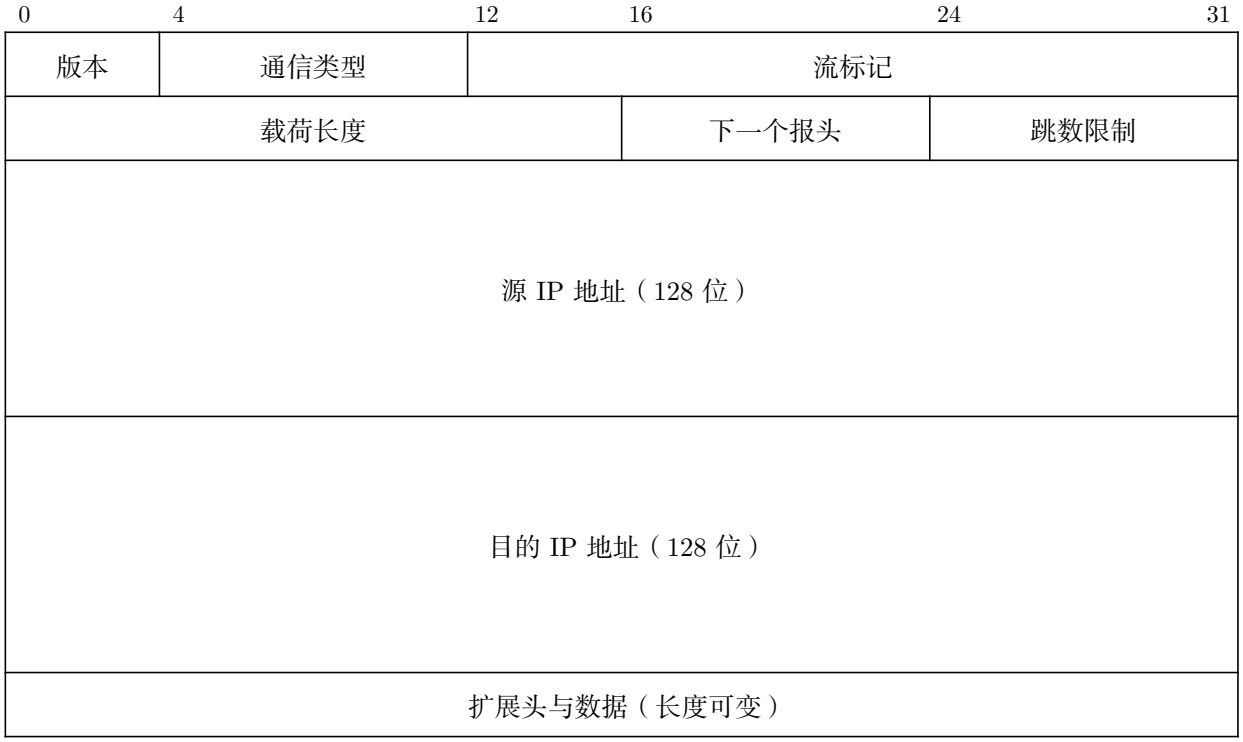
IPv6 还解决了 IPv4 中的性能问题:

IPv6 取消了分片机制，如果发送的数据报无法通过一个较小 MTU 的链路，路由器会丢弃他然后返回一个 ICMP 差错报文。发送主机收到之后会发现更小的数据报。取消了分片机制能够大大加快转发速度。

校验和字段也被取消了。因为每次都重新计算校验和会加大端到端延迟，并且也没有太多效果。

IPv6 还解决了地址自动配置的问题，不再需要 DHCP。

4.4.2. IPv6 数据报格式



源地址和目的地址各占 128 位，即各跨四行；最后一行的扩展头与数据仅为长度示意，不固定为 32 位，也不属于基本首部。

通信类型：

跳数限制：虽然在IPv4中，按理说如果路由器处理花了1.x秒，TTL应该减2而不是减1。但由于很多路由器实际上只会无论如何-1，所以IPv6不叫TTL了，直接就叫跳数限制，规定经过一个路由器-1。

4.4.3. *IPv6* 地址

由于 IPv6 地址空间巨大，因此分配给主机的地址空间可能带有多组 0000，比如 8000:0000:0000:0000:4321:0501:AB96:56CD，为了方便起见我们可以压缩表示，比如 00D0 写成 D0，0000 写成 0。对于多组:0:可以直接简写成::。

由于 IPv6 的地址较长，我们不采用子网掩码来表示，而是直接给出地址/前缀长度的格式。

IPv6 还有一种嵌入 IPv4 的表示，这是为了方便把已有的 IPv4 地址映射到 IPv6，因此最后 32 位用点分十进制表示。

4.5. 路由协议

4.5.1. 自治系统

各种各样的算法都需要在路由器之间交换网络信息，当网络的规模大到一定程度，会极大的消耗带宽和时间，难以收敛。而且个个组织也通常有自治性的需要，规定所有网络用一种算法并不现实。

我们会按照组织边界，管理边界或者技术边界等把网络分为多个**自治系统**（AS），每个 AS 内路由器使用相同的协议，我们让网关路由器负责的负责与其他自治系统交换路由可达信息。

一个 AS 并不一定是一个网络。相反，它可以有宣告多个 IP 前缀。

4.5.2. 域内路由与域间路由

自治系统之间的路由被称为域间路由或内部网关协议，也就是**IGP**，自治系统内部的路由被称为域内路由，也就是 **EGP**。

4.5.3. *RIP* 路由协议

RIP 是最早的 IGP

RIP 使用 DV 算法，度量跳数，也就是每条链路权值均为 1

RIP 封装到 UDP 中传输。

在 RIP 中，相邻路由器通过 **RIP 响应报文**来交换距离向量，也称为 **RIP** 通告，交换频率约为 30s 一次

RIP 被限制在两点间最短距离不超过 15 的 AS 间使用，因为距离 16 在 RIP 中被视作无穷大。

超过 RIP 如果超过 180s 还没收到邻居的信息，则视为不可达。

4.5.4. *OSPF* 路由协议

与 RIP 不同，OSPF 的权值不只是跳数，也可以是链路带宽

OSPF 报文直接封装到 IP 数据报中进行传输。

OSPF 有以下相较于 RIP 的优点。

- 支持多播路由
  也有一些缺点：
- 对路由器处理器要求高：
- 占用一定带宽： 因为需要频繁的发送和应答广播信息

4.5.5. *BGP* 路由协议

BGP 是 Internet 事实上的 EGP。BGP 通过 179 端口建立的 TCP 实现。

BGP 协议非常非常复杂，暂且不要去思考一个子网中为什么需要多台 BGP 路由器，以及非网关的 BGP 路由器等等问题，教科书不要求我们掌握细节。所以我们只要记住下面的基本概念即可。

BGP 要求在每个系统内指定对应的路由器作为 BGP 路由器，而 BGP 路由器之间并不是发现的，而是网络管理员会选择让哪些合适的路由器。这些 BGP 路由器通过 IGP 协议把路由表信息传播到普通的路由器。

一个 AS 内部的多个 BGP 使用内部 BGP 会话（记为 iBGP）来彼此交换子网可达性信息。

通常来说 eBGP 会寻找相邻的路由器交换信息。

BGP 有四类报文

- OPEN：开启 BGP 连接，也就是紧随着 TCP 握手之后进行
- UPDATE
- KEEPALIVE：即使没有新的更新，也要发保活消息，不然对方不知道自己是否下线了。
- NOTIFICATION：用来报错，发了 NOTIFICATION 之后 BGP 连接就终止了。

你可能会好奇都有 TCP 连接了干嘛还要维护自己的 open 和 keepalive 机制呢。因为负责 TCP 的是操作系统中的 TCP 协议栈，就算它没问题，BGP 应用程序本身可能是有问题的。所以高层协议必须总是建立自己的连接机制。

在使用 BGP 的网络中，每个 AS 都有一个全局唯一的自治系统号 ASN 作为 AS 的表示。

BGP UPDATE 消息时包含两类内容

- 前缀
- 属性

前缀加上属性通常称为一条路由。 其中属性又包括

- AS-PATH：到达某子网需要遍历的 AS 集合
- NEXT-HOP：一个开始某 AS-PATH 的路由器接口。
  我们举个例子，一个最简单的三个 AS1，AS2，AS3，A 首先会发送自己的网络前缀，然后把自己的 ASN 加上入到这个前缀的 AS-PATH 里，然后发给 B 然后 B 收到之后会把自己的 ASN 加入 AS-PATH 之前，然后 AS2 又发给 AS3，现在 AS3 知道了有了(AS2,AS1)这条路径可以到 AS1 的子网前缀，对于 AS2 来说，它收到的 next-hop 是就是 AS1 面向 AS2 的网关的出口地址，而 AS2 转发给 AS3 的时候，无论是 AS1 还是 AS2 的子网的 next-hop 都会是 AS2 面向 AS3 的网关的出口地址

存的路由器采用以下优先级过滤路由：

- AS 网络的管理员设置的本地偏好值
- 本地偏好值相同，AS-PATH 最短
- AS-PATH 相同，具备最近 NEXT-HOP 的被选择

4.6. *IP* 组播

4.6.1. 组播的概念

让单个分组同时达到多个主机

4.6.2. *IP* 组播地址

224.0.0.0/4，也就是 224.0.0.0-239.255.255.255，前缀为 1110 的，被归为 D 类地址，每个 D 类地址代表一个多播组，把目的地址设成这个就是向目标组多播。

多播只能用于 UDP，不能用于 TCP，因为 TCP 是一对一的连接，不产生 ICMP 差错报文（因为会有多条路径，反馈太多）

组播分为永久组播组和临时组播组，永久组播组是事先规定好的，比如 224.0.0.1 就是本地所有 IPv4 多播主机，224.0.0.2 是所有本地链路所有多播路由器，224.0.0.5 用于 OSPF. 当然了我只是举个例子，你并不要具体记忆这些地址。

临时组播组是大家自由创建和加入的,不过我们具体是怎么确定哪个计算机在哪个组里呢,这就得依赖组成员关系管理，由 IGMP 协议控制。 IGMP 协议用于实现多播，因此和 ICMP 一样属于 IP 协议的一部分。IGMP 的具体细节不做要求。

4.7. 移动 IP

4.7.1. 移动 IP 的概念

假设我们在车上使用电脑，DHCP 和具体的局域网有关。但是当在网络间移动的时候，我们需要不停地 DHCP，原来的 TCP 连接就会丢失需要重新连接。

4.7.2. 移动 IP 通信过程

家里的 IP 地址，首先我们需要在我们家中所在的局域网配置 Home Agent.

4.8. 网络层设备

4.8.1. 路由器的组成和功能

从功能角度，路由器可以分为输入端口，交换结构，输出端口和路由处理器

交换结构包含三种类型：

- 基于内存交换：早期的路由器像是传统的计算机，分组到达端口时通过中断方式输入内存，根据计算后再从内存输出到输出端口。
- 基于总线交换：路由器的输入和输出端口同时连接到一条总线上，有点像计算机 I/O 中的 DMA 方式。因为确实感觉有点没必要让数据经过内存走一遍，计算机直接控制就行了。同时控制也从计算机改为由专用硬件控制，可以直接把转发表写入端口
- 基于网络交换：总线交换受制于总线速率，同一时刻只能有一个分组经过总线。复杂的互联网络可以并行交换传输，交换速率大大提高

4.8.2. 路由表的分组转发

传输层

CHAPTER 5

5.1. 传输层提供的服务

5.1.1. 传输层的功能

但请注意，IP 提供的是一种尽力而为（best-effort）的交付服务。它不保证：数据报一定到达。也不保证数据报按照发送顺序到达。数据报只到达一次。更不保证数据报传输有多快。即使 IP 数据报成功到达主机 B，操作系统仍然面临一个问题：应该把里面的数据交给哪个应用程序？

传输层的核心是为应用进程提供端到端的逻辑通信服务，注意，是应用进程，这是它区别于网络层的地方。一台主机可能运行不同的应用。

因此，端口寻址就是传输层要实现的第一个功能，而接下来的问题，就是主机在网络层收到不同的 IP 数据报后，怎么正确的数据把它们传给不同的端口，这个过程就是多路复用。

而传输层作为端上的最后一层协议，会被应用层复用，它还想额外实现一些功能。

传输层可以理解为应用层和网络层之间插入的抽象层，理论上我们可以让应用直接调用IP协议来完成这一切，但我们发现许多不同的应用本身都包含传输数据和端口寻址这一层，于是我们把这一层抽象出来，传输层不关心数据具体的意义，传输层只要负责传递数据就可以了。同时重传和差错控制等等也可以交给这一层解决。

5.1.2. 传输层寻址与端口

假设都使用相同的操作系统，进程 ID 是一个不错的标识，不同的主机有着不同的操作系统，因此，我们必须再抽象一层，提出端口这个概念。端口号为 16 位二进制整数，最大是 65535。两个不同的网络服务互相通信，往往会使用不同的端口号。

5.1.3. 无连接服务与面向连接服务

无连接服务是指数据传输之前无需进行任何信息交换，也就是通常所说的握手，直接构造报文段。反过来，面向连接服务就要先建立逻辑连接再传输数据，结束之后还要再删除连接。

我们回顾一下之前的协议，显然 IP 和以太网都是无连接服务。

5.2. UDP 协议

5.2.0a. UDP 特点

UDP 是无连接不可靠的，消息可能乱序到达，也没有拥塞控制。它就可以看成是一个只提供端口寻址和多路复用的最最最基本的传输层协议，还加入了校验和的差错控制。

5.2.1. UDP 数据报

UDP 的数据报格式非常简单，源端口号和目的端口号各两个字节，还有各两个字节的长度和校验和。

你可能会为为什么还需要长度标识，IPv4 首部里不是已经包含了，你说的对，UDP-Lite 就没有首部。

5.2.2. UDP 校验

5.3. TCP 协议

TCP 提供的是面向连接的可靠服务，基本上提供的功能应有尽有。

TCP 提供的是全双工通信，发送的同时也能接受。TCP 连接的两端都有发送缓存和接受缓存，应用程序发送消息进入缓存之后就可以干自己的事情了。

TCP 要搞接受缓存的根本原因是应用程序不一定来得及处理完那么多数据。

TCP 是面向字节流的，意思就是它不需要自己操心报文的到达啥的。

同时缓存还可以用来处理乱序到达的字节。

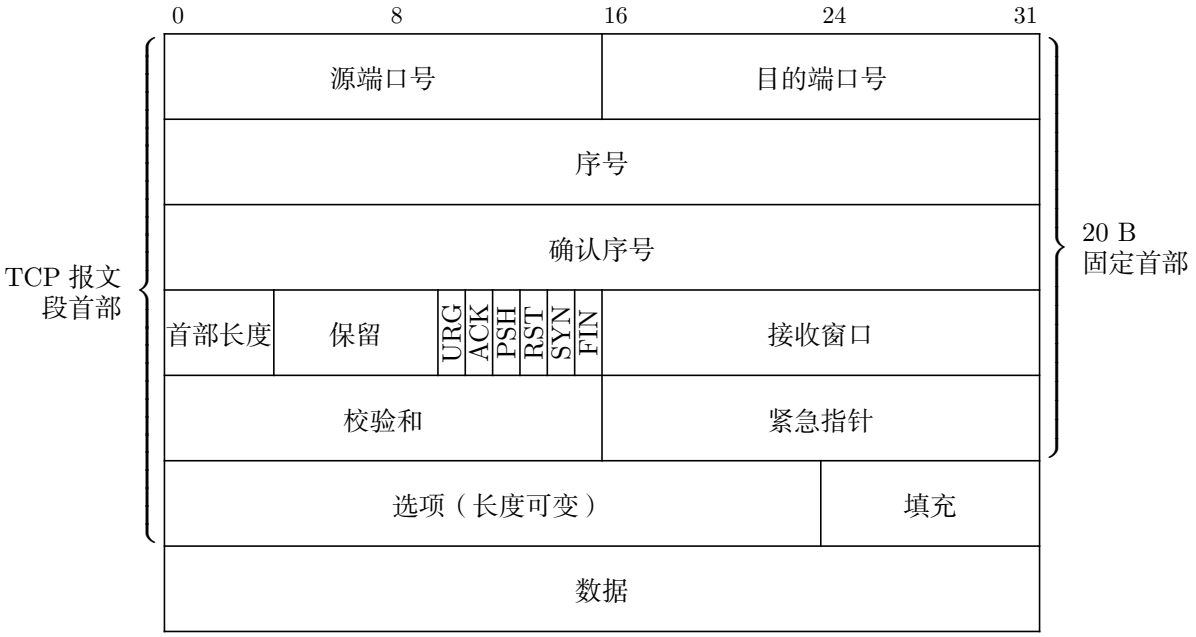
TCP 能从缓存中放入报文段的数据数量取决于 MSS，Maximum Segment Size，这个大小是根据本地主机的链路层来设置的。省得本地主机再切。

值得注意的是，MSS 是 TCP 报文段中的数据段的最大长度，而不是整个 TCP 报文段的最大长度，也就是不包含首部。

5.3.1. TCP 段

5.3.1a. TCP 报文段序号与确认序号

5.3.1b. TCP 段结构



选项和填充的宽度仅为示意，并非固定为 24 位和 8 位；数据部分的长度也可变。

序号：每一个报文段的实质是一连串的字节，我们给这些字节编号之后，首部中的序号段表示该报文段所携带数据的第一个字节的序号，而不是报文段自身的编号。

TCP 采用累积确认，因此确认序号不能是第一个字节的序号；我们把它设置为了期待收到的下一个字节的编号，比如我已经收到了字节 0-999，那么发送的 ACK 确认序号就是 1000。

你可能会问我们为什么需要 ACK 标志，这是因我们第一次建立连接向对方发送消息的时候没有确认任何信息？你可能又会说为什么不把 ACK 序号设为 0，但是序号是循环使用的，$2^{32}-1$ 用完了之后就回到了 0 了，所以 0 本身也是个有效的序号。

SYN=1 表示这个报文段刚好是用来建立请求连接，或者回应建立请求连接的。

紧急指针和 URG 标志位：RFC9293 表面新应用不该使用紧急机制，作为历史包袱即可。

FIN=1 用于断开连接过程

RST=1 代表出现严重差错，必须释放并且重新连接

接受窗口字段表示接受方的缓存剩余空间，用于实现流量控制。

保留字段的值目前为 0。

5.3.2. *TCP* 连接管理

5.3.2a. *TCP* 连接建立过程

TCP 是全双工通信，这意味着对方在像我们发送的时候我们也在不断向对方发送消息，所以 ACK 本身是可以在帮别人发送消息的时候顺带完成的，通过确认序号。

主机 1 首先会向主机发送第一次握手，此时 ACK=0，SYN=1，这个特殊的报文段被称为 **SYN 报文段**.

然后主机 B 会回复二次握手，此时 ACK=1，SYN=0，这个报文段被称为 **SYNACK 报文段**.

对于普通的空数据报文段，是不会消耗序号的，但是 SYN 报文段和 SYNACK 报文段却需要消耗序号，这是为什么呢?

这是因为序号的本质是用来确认我们收到了某条消息。在握手阶段中，我们需要确认每一条消息都被收到。但是在传输过程中，我们不需要对 ACK 再发送 ACK，不然你还可以发送 ACK 的 ACK 的 ACK，有点没完没了。累积确认的一个好处就是 ACK 丢失了也不要紧。

5.3.2b. *TCP* 连接拆出过程

TCP 的断开连接，通常称为四次挥手，过程如下：

用户在发送完最后一个数据段之后，发送一个 FIN 段，其中 FIN=1，序号要+1，这个段本身不再发送数据，并且后续也不能发送任何数据.

服务器会返回一个 FINACK，请注意 FINACK 是能返回数据的。然后服务器可能会继续发包发送数据，一直到服务器把所有需要的数据发完为止。

服务器发完之后会反向向用户发送 FIN，然后用户回 FINACK，具体细节相同。

5.3.3. *TCP* 可靠传输

5.3.3a. *TCP* 的可靠数据传输机制

TCP 靠序号和累积确认避免了乱序。

TCP 会超时重传，采用了选择重传协议。

TCP 有校验和，因此实现了差错检测，有错就扔掉不发送 ACK

5.3.3b. *TCP* 计时器超时时间设置

计时器的合适的超时时间很重要，我们采用 RTT 采样方式，RTT 采样是间隔采用，而不是每发一个报文都采用，而且重传的报文永远不纳入采样，只有单次发送收到的被纳入采样。

显然采样会波动，

$$\text{Estimated RTT} = (1-\alpha) \times \text{EstimatedRTT} + \alpha \times \text{SampleRTT}$$

这是一个迭代公式。

超时时间的设置也有讲究，比较传输时间的波动情况也要考虑，我们定义

$$\text{DevRTT} = (1-\beta) \times \text{DevRTT} + \beta \times (\text{SampleRTT} - \text{EstimatedRTT})$$

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4 \times \text{DevRTT}$$

TCP 如果仅仅依赖超时重传存在一些问题，会导致等待较长时间才会重传。我们采用重复 ACK 加快重传。

下面我们讲 ACK 策略：

假设所有所期望的报文段按序到达，TCP 接收方会延迟发送 ACK，否则才会确认。

5.3.4. *TCP* 流量控制

TCP 采用滑动窗口协议，但是流量动态变化。发送端的发送窗口总是确保不大于接受端发回数据的接受窗口（还收到拥塞控制）

5.3.5. *TCP* 拥塞控制

拥塞是指太多主机太快地发送数据，超过了网络的处理能力。拥塞会导致：

- 数据分组通过网络的时延显著增加
- 由于队列满而导致大量分组被丢弃

  TCP 的拥塞控制是从端到端的角度推断网络是否拥塞。基本策略是网络未拥塞式，加性增加窗口大小，拥塞时乘性减小大小从而维持平衡。

  TCP 发送端维护一个拥塞窗口的长度变量 Congwin

  拥塞算法包含四部分：

- 慢启动
- 拥塞避免
- 快速重传
- 快速恢复

  慢启动：开始发送数据时，拥塞窗口用非常快的形式增长，先设置为 MSS 的值，然后每收到一个确认，Congwin 增加一个 MSS. 所以我们连续发的包会越来越多，收到数据会越来越快。直到达到了预设的 Threshold，这个时候我们要慢慢增加。

接下来进入拥塞避免阶段，拥塞避免阶段仍然在不断加高自己的容量。

拥塞避免：每经过一个 RTT，Congwin 的值 +1 MSS。如果计时器超时了，我们令 Threshold=Congwin/2，然后让回到 1

因为超时情况比较特殊，如果后续的包到达了，可能丢包的只是个别，但超时说明情况相对研究一下，因此回到 1 重新探测。

快速重传：收到 3 次重复 ACK 的时候。我们会重传一次 ACK 索期待的报文段

快速恢复：快速恢复配合快速重传使用。它和拥塞避免的差别是窗口不回到 1，而是回到 Threshold 然后进入拥塞避免状态

应用层

Chapter 6

6.1. 网络应用模型

6.1.1. 客户/服务器（*C/S*）模型

6.1.2. 对等（*P2P*）模型

6.2. *DNS* 系统

6.2.0a. *DNS* 的主要功能

DNS 负责把域名映射到 IP 地址，此过程称为域名解析。

为了域名解析，我们需要建立数据库。而让一个域名服务器存储全部

6.2.1. 层次域名空间

格式：..., 三级域名.二级域名.顶级域名

顶级域名又称 TLD

域 domain，是层次命名空间中的一颗子树。

6.2.2. 域名服务器

域名服务器要解决的核心问题是管理问题，比如我们更换了服务器，想把自己的域名重新映射到新的 IP。就要上服务器。

如果全球只依赖一个中心的服务器，那么它的流量和读写会非常大极其考验数据库的性能，你当然可以让几台同时处理，它们之间的同步会变得麻烦。

于是我们让域名和域名服务器采用层次化管理，以及委托机制。

DNS 服务器的管辖是**区(zone)**，任何一个 DNS 服务器都可以把自己的区中的东西委托给别的服务器。而顶级域名服务器中，上面的许多域名通常都会被委托给权威服务器。

因此 zone 是一个记录组，因为可以委托，所以 cs.example.com 如果委托出去，它就不在 example.com 所在的域名服务器的 zone 里了。

中间域名服务

域名服务器可以分为 4 类

- 根域名服务器: 有 13 台，从 a.root-servers.net 到 m.root-servers.net，它们是预配置好的 IP，每个根域名服务器都记录了顶级域名服务器的域名
- 顶级域名服务器
- 权威域名服务器：权威域名服务器是相较于 zone，或者说具体的域名而言，就是说在里面你能直接能找到它所对应的域名就是权威域名。
- 中间域名服务器
  互联网上共有 13 个不同的根域名服务器，它

6.2.3. 域名解析过程

域名分为递归解析和递归解析和反复解析

递归解析器可以利用共享缓存来提高效率

首先上层的根服务器和递归服务器往往是迭代查询而不是递归。

当然，如果我们是依赖缓存提供解析，需要告诉用户这是非权威的

- 先查询本地主机的缓冲区，如果以前解析过就可以立即响应
- 没有向本地域名服务器发出请求

6.3. *FTP*

6.3.1. *FTP* 协议的工作原理

FTP 等待客户端通过这个连接。每一个客户连接上之后，会 fork 出一个进程来接受用户的指令和下载。剩下一个进程继续等待连接。显然 FTP 是有状态的。

FTP 是一种非常古老的协议，同时承担了当时远程文件管理和文件内容传输的责任。

在今天其实大家都不推荐用FTP了，而且主流的浏览器也已经放弃了FTP的支持，这是因为FTP不加密，容易遭受攻击让用户下载到病毒软件。ftp 协议没法断点续传。对于这种情况 async 更合适。此外 ftp 对于想要改动服务器上的某一行文件也不适合。这种时刻应该用 NFS.

6.3.2. 控制连接与数据连接

FTP 的交互和文件传输全都使用 TCP

FTP 提供一些比如 ls 和 cd 之类的简单指令，这些都依赖控制连接传输。

数据连接的端口是 20. 控制连接的端口是 21， 数据连接是通过控制连接打开的。

每一次数据

FTP 这种控制连接与数据连接分开的方式被称为**带外控制**。而数据和命令放在一起的，比如 HTTP 被称为**带内控制**。

6.4. 电子邮件

6.4.1. 电子邮件系统的组成结构

电子邮件的核心是电子邮件服务器。我们每个人电脑上都有电子邮件客户端,它们也被称作为用户代理，User Agent，但是我们的电脑不是一直都上线的。因此我们需要一台邮件服务器时刻保证它能够接受互联网上发过来的邮件，邮件服务器为很多个用户提供服务，每个用户都有自己的邮箱。

电子邮件由

- User agent 用户代理
- 邮件服务器
- 简单邮件传输协议，Simple Mail Transfer Protocol, SMTP:
- 邮件读取协议

  典型的邮件读取协议有 POP，IMAP，JMAP 等

6.4.2. 电子邮件格式与 *MIME*

值得注意的是，一封电子邮件同时发给多个人，那么每个收件人都会在头里面看到其他的手机人。

Subject: 标题

Reply-to 有时发信用的地址和回信用的地址不同。就需要

Cc：在工作中我们需要这种情况。老板布置给你一个任务。你需要和同事沟通。然后希望让老板能看到你们之间的沟通的结果。这样老板可以方便的查阅并且插进来。但是你需要知道这条消息具体是发给同事的，老板不是主要要采取行动的收件人。

Bcc：假设你要向别人群发邮件，但是又不希望让它们知道彼此的地址，就应该使用这一功能。

SMTP 只能传输 7 位 ACSII 码内容。因此加入想要传图片啥的，就只能先封装成 7 位 ASCII 码再解码，这就是常说的 MIME 标准。通常来说我们喜欢用 base64 来编码图片啊啥的。我们所谓的附件，本质就是从邮件中读取出的 MIME 头。

6.4.3. *SMTP* 协议与 *POP3* 协议

6.4.3a. *SMTP* 及其特点

6.4.3b. *SMTP* 邮件发送过程

1. 连接建立阶段

我们的客户端会每过一段时间到 SMTP 服务器上去拉拉看如果有什么新邮件，有的话就说明需要拉取。我们会通过端口 25 和服务器建立 TCP 链接。

接着我们会向服务器发送一系列 RCPT 指令。

SMTP 协议的缺点就是它仅能 发送 ASCII 格式的邮件。

6.4.3c. *POP3* 邮件接受过程

6.5. *WWW*

6.5.1. *WWW* 的概念与组成结构

WWW 包括 Web 服务器，浏览器，HTTP 等部分。 一个 Web 网页通常会有一个 HTML，在基本页中再引用若干其他对象。i

每一个 URL 有两部分组成，协议名：主机域名，端口号，对象的路径名，query

6.5.2. *HTTP* 协议

6.5.2a. *HTTP* 及其特点

HTTP 协议的规范定义的并不会，流行的 web 客户端往其中添加新功能，成为了事实上的标准。

6.5.2b. *HTTP* 交互过程

6.5.2c. 持久和非持久连接的 *HTTP*

HTTP 根据使用策略的不同，可以分为非持久连接和持久连接。

HTTP 1.0 默认使用非持久连接。HTTP/1/1 默认使用持久连接

每个请求

HTTP/1.1 可以在请求头中附加 Connection:close 认为自己是持久的。

6.5.2d. *HTTP* 报文

HTTP 报文由四部分组成：

- 起始行
- 首部行
- 空白行
- 报文主体

6.5.2da. 请求报文结构



6.5.2db. 响应报文结构



SP 表示空格，CRLF 表示回车换行。首部行可以有多行，空白行分隔首部与报文主体；各栏宽度仅为示意。

6.5.2dc. 请求方法

HTTP 有一下集中典型请求方法

- GET: 请求读取 URL 标识的信息
- HEAD: 请求读取 URL 标识的信息首部，也就是响应报文中不包含对象
- POST: 向服务器提交信息，我们登录帐号，填表单都是在 post
- PUT: 在指名的 URL 下存储一个文档
- OPTION: 请求一些选项的信息，不必深入理解

HTTP 状态码由三位十进制数组成 1xx: 信息提示 2xx: 成功 3xx: 重定向 4xx: 客户端错误 5xx: 服务器端粗偶

常见的: 100: Continue 200: OK 301: Move Permanently 404: Not Found 505: HTTP Version Not supported

6.5.2e. *HTTP* 交互过程响应时间分析

6.5.2f. *Cookie* 工作原理

HTTP 是无状态协议，不保持客户的任何信息，但是无论是登录论坛还是购物网站。网站需要记住用户的身份，进行追踪。

浏览器会通过在响应报文中的 Set-cookie 这一栏给用户分配 Cookie

网站要自己在后台存储和维护 Cookie 信息，浏览器也要自己存储和管理自己的 Cookie，浏览器如果发现对应网站有 Cookie，就会在发送请求的时候携带 Cookie