计算机网络 第二章 读书笔记

2.1 应用层协议原理

客户-服务器架构中有一个永远在线的主机，叫做 server，其他向它请求服务的计算机我们叫做 clients，这种架构中，各个 client 不是直接与彼此交流的，这种架构的另个特点是 server 有个公开的，固定的地址，叫做 IP 地址，由于 server 有个固定的地址而且一直在线，所以 client 可以随时发送数据包到 server。用这种结构的应用有 Web， FTP ,Telnet 和 email .

对于客户-服务器架构的应用来说，一个服务器可能不够用，所以便出现了 data center，这里集合很多主机，用来组成一个强大的虚拟主机 。

在构建应用之前，我们还得知道程序通信的基本原理，在计算机术语中叫做进程通信，如果进程运行于同一个终端系统，那进程间通信就可以解决问题，但是我们这本书更关注的是在不同计算机间通信。

两个不同终端系统中的进程通过网络交换 message 来通信，进程通信在应用层。

一个网络应用由彼此通信的进程组成，我们通常把发送方叫做 client，响应方叫做 server 。P2P 应用中也许你发现终端即是 client 也是 server，但是我们定义如下：

In the context of a communication session between a pair of processes, the process that initiates the communication (that is, initially contacts the other process at the beginning of the session) is labeled as the client. The process that waits to be contacted to begin the session is the server.

通过学习我们知道，应用包含很多对交流进程，两个进程彼此发送信息的时候，进程发送到网络和从网络中接收信息都要一个软件接口，叫做 socket，一个类比就是 process 是房子，socket 是门。

一个主机上的程序如果想发送数据包到另一个主机上的程序，那么接收方就得有个地址，为了识别目标，两个信息需要被指明：（1）主机的地址（2）能明确出主机上的接收程序的标识符 。

在互联网中，我们用 IP 地址来识别主机，它是个 32 位的数，可以认为它是唯一的，除此之外，我们还得有个标识说明哪个程序（或者更准确，哪个接口）来接受消息，因为通常一台计算机会运行很多网络应用，目标主机的 port number（端口号）能解决这个问题 。很流行的应用都有特定的端口号，举个栗子，Web server 的端口号是 80， mail server 的端口号是 25，一个列表。

数据包在传输中会丢失，对于一些应用，包的丢失可能造成极大的损失（比如商务软件），因此，我们需要一种保障信息传输正确且完整的服务，如果一个协议提供了这样一

种服务，就叫做提供 reliable data transfer ，有了这个服务，发送方可以确信信息一定能完整无误的到达。
当协议不提供这种服务的时候，也许就会出现信息丢失，这种协议可以被 loss-tolerant application 接受，大多数的多媒体软件是用这种协议的，因为一些小的信息丢失并不会造成很大影响 。

Throughput 吞吐量
前边我们已经说过，两个进程间通信，发送方可以发送信息到接收方的速率就是吞吐量，因为网络中会有其他 sessions 共享带宽，而且这些 sessions 会来来去去，就造成了吞吐量随时间波动，这些就引出了另一个传输层协议可以提供的服务，叫做保证指定值的吞吐量，这服务会确保吞吐量始终是大于等于你给定的值，举个栗子，一个网络通话应用需要 32 kpbs 速率，如果协议不能提供这个速率， 那这个通话可能得放弃，这些对吞吐量有特殊需求的应用叫做 bandwidth-sensitive application ，当今许多多媒体应用都是这种，当然也有一些应用用自适应编码技术来跟当前可用吞吐量匹配 。还有一类弹性应用是无所谓吞吐量的。

2.2 Web 和 Http
超文本传输协议 HTTP 是 Web 的心脏，HTTP 在两个程序中执行：client program 和 server program ，运行于不同的终端系统，通过交换 HTTP messages 来彼此交流，HTTP 定义了这些 messages 的结构和 client 与 server 交换信息的方式 。再详细说明 HTTP 之前，我们先来看一下 Web 术语，

Web page（也叫做 document）由 objects 组成，一个 object 就是一个简单的文件，例如一个 HTML 文件，一张 JPEG 图片，或者一个视频片段，这都是可以通过 URL 寻址的，大部分 Web page 都有一个 base HTML file 和几个其他的 objects。
base HTML file 通过其他对象的 URL 在页面中引用它们，URL 有两个部分：存储这个对象的服务器的主机名和这个对象的路径名 。

由于 Web 浏览器（比如火狐和 IE）执行 client 方的 HTTP 协议，所以在 Web 的语境中，我们可以把 browser 和 client 替换使用 ，Web 服务器执行 server 方的 HTTP 协议，存放 Web objects（每个都可以是可以通过 URL 寻址的）。
HTTP 定义了 Web client 如何从 Web server 请求 Web pages 和服务器如何把 Web pages 发送给 client。

HTTP 是用 TCP 作为传输层协议，HTTP client 先与 sever 建立一个 TCP connection ，当 connection 完成，浏览器和服务器进程就通过 socket interface 用 TCP 通信 。HTTP 把消息送入到 socket interface 中，消息就脱离了 client 的控制而由 TCP 控制，由于 TCP 提供可靠

地传输服务，所以我们就不用考虑和担心它是怎么处理信息的丢失和顺序问题，这也就体现了分层架构的好处，那些工作是较低层协议的工作，在这里，准确的说，是 TCP 的工作。

因为 HTTP 服务器不会保留 clients 的任何信息，所以如果我们连续请求两次同一个对象，它就会回复两次，因此，HTTP 被称为 stateless protocol，

在网络应用中，client 和 server 可能不间断的通信，也可能是周期性的，还可能是断断续续的，由于这些交流是在 TCP 协议之上发生的，那么就得考虑是不是得为每对客户-服务器都建立独立的 TCP 连接呢，或者是都用同一个 TCP 连接？前一个方法是 non-persistent connection，后一种方法是用了 persistent connection，虽然 HTTP 默认是用 persistent connection，但 HTTP clients 和 server 可以被设置成用 non-persistent connection 。

cookies 技术有四个部件：1）.HTTP response message 中的 cookie header line 2）.HTTP request message 中的 cookie header line 3）.在用户终端系统中的一个 cookie 文件并且这个文件由用户的浏览器管理 4）.Web site 上的一个后端数据库 。
假设，Susan 经常在 PC 上用 IE 浏览网页，第一次进入 Amazon.com，假设以前曾经进入过 eBay 网站，当 request 到达 Amazon Web server，这个服务器生成一个独一无二的识别数字，然后再后端数据库生成一个被这个数字索引的入口，服务器然后回复 Susan 的浏览器，在 HTTP response 里有个 Set-cookie：行，这个行里包含着识别码，

Web cache，也叫做 proxy server，是个代替 origin Web server 满足 HTTP request 的网络实体，Web cache 有自己的磁盘存储器并在里边保存最近所请求 object 的 copy。

虽然 Web cache 可以减少用户感知的 response time，却产生一个新问题——cache 中的拷贝也许会过时，也就是说，当 cache 还保存着上次的 copy 的时候，也许服务器里的数据已经被修改了，幸好 HTTP 有个机制可以让 cache 证实它内存中的 objects are up to date，这个机制叫做 conditional GET，如果符合以下两项，一个请求就可以叫做 conditional GET：1）.请求方法是 GET 2).请求信息包含一个行——If-Modified-Since ：

2.3 文件传输协议： FTP
典型的 FTP 会话中，用户想要从远端的计算机接收文件或发送文件给远端，为了用户能连接远端用户，用户必须提供用户识别和密码，提供这些授权信息之后，用户就可以从本地文件系统发送文件到远端文件系统，反之亦然，
用户通过 FTP user agent 与 FTP 进行交互，用户先提供远端计算机的 hostname，引发本地计算机的 FTP client process 与远端计算机的 FTP server process 建立 TCP 连接，然后用户提

供用户身份证明和密码，这些会被当做 FTP commands 的一部分通过 TCP 连接发送出去，一旦服务器核准了这个用户，用户就可以把本地文件系统的文件复制到远端文件系统，反之亦然 。

HTTP 和 FTP 都属于 file transfer protocol，有很多共同点，例如，它们都运行与 TCP 之上，然而，这两个应用层协议之间也有些重要的不同点，最大的不同就是 FTP 用两个平行的 TCP 连接传输文件：a control connection 和 a data connection ，前者用来在两个 host 之间发送控制信息——比如用户身份证明，密码，commands to change remote directory，commands to “put” or “get “files，
data connection 就是用来传输文件的，因为 FTP 用单独的控制连接，FTP is said to send its control information out-of-hand，而 HTTP，正如你想到的，所有的信息都是通过一个连接来传输，所以，HTTP is said to send its control information in-hand，我们接下来的章节还会看到，email 的主要协议 SMTP 也是 sends control information in-hand .
当用户与远端计算机开始 FTP 对话，client 方先会与 server 方在服务器的 21 号端口建立 control TCP connection，client 方会通过这个连接发送用户身份证明和密码，也会通过它发送命令修改 remote directory，当服务器收到一个请求文件的命令时，server 方就会与 client 建立 TCP data connection ，但是每个 data connection 发送一个 file 后就会关闭，下一个文件要发送的时候再建立新的 data connection，因此，用 FTP 协议，这整个过程 control connection 一直是开着的，但是 data connection 每个文件都需要生成一个，
整个会话过程中，FTP 服务器必须保存用户状态，特别是，服务器必须要用一个特定的用户账户来关联 control connection，服务器还必须在用户浏览远端 directory tree 时随时追踪用户当前位置，为每个正在进行的会话追踪状态信息极大的限制了 FTP 可以同时支持的会话数目，我们回想 HTTP，是 stateless，不用追踪任何用户状态。

2.4 因特网中的电子邮件

电子邮件是互联网发端时最受欢迎的应用，这些年来变得越来越制作精良而且功能强大，成为互联网上最重要，最有用的应用
in this section we examine the application-layer protocols that are at the heart of Internet e-mail，在这之前，我们先来大体看一下互联网邮件系统和它重要的组成部分。

SMTP 把消息从发送方 server 传输到接收方 server，SMTP 要比 HTTP 岁数大很多，虽然 SMTP 有很多好品质，在网络中无处不在，但它是个传统技术，有很多古老的特点，例如，它限制所有的信息体为 7-bit ASCII，这个限制在 1980 年代显得合情合理，因为那时候没有人会用邮件发送很大的附件，图片，视频或者音频文件，但是今天，多媒体时代，7-bit 的限制真是个痛点——这需要二进制的多媒体数据在通过 SMTP 传输前被编码成 ASCII，还需要经过 SMTP 传输的 ASCII 数据再次解码成为二进制，从上边学习我们知道，HTTP 传输多媒体数据前就不需要 ASCII 编码。

现在我们简单的对比下 SMTP 和 HTTP，两个协议都是用来把文件从一个计算机传输到另一个计算机，HTTP 在 Web server 和 Web client 之间传输文件，SMTP 在两个 mail server 间传输文件，传输文件时，persistent HTTP 和 SMTP 都是用 persistent connection，因此两个协议有共同点，然而，它们还有明显的不同，首先，HTTP is mainly a pull protocol——人们从服务器加载信息并且从利用 HTTP 从服务器拉信息，特别是，TCP 是想要收到文件的机器启动的，另一个方面，SMTP is primarily a push protocol——发送方 mail server 把文件 push 给接收方的 mail server，特别是，TCP 连接是由想要发送文件的一方启动的
第二个不同，我们是说早期，SMTP 需要每个消息，包括消息的内容，都必须是 7-bit ASCII 格式，如果消息中含有内容不是 7-bit ASCII，那这个消息需要被编码成 7-bit ASCII，HTTP 协议不实行这种限制
第三个不同关于一个包含有文本和图片（或者其他 media types）的文档是怎么被处理的，HTTP 会对每个 object 进行封装，而 Internet mail 把多有 objects 放在一个 message 中。

2.5 DNS

就像人们可以有很多种办法被识别一样，Internet host 也是一样，一种方式就是 hostname， hostname 提供很少，甚至根本没有信息来显示这台主机在网络中的位置，再者，hostname 包含可变长度的字母数字，这些将是路由器处理它们变得困难，因为这些原因，我们还用 IP 地址来辨识主机
IP 地址由四个字节组成并且有刚性的层次结构，每个字节都是十进制 0-255 间的数，IP 地址是分层次的，因为我们从左到右读的时候，我们会获得越来越精确的信息，关于主机在哪里，就像是我们读信件上的地址一样，越往后读越是清楚地知道精确的地址。

我们介绍了两种识别主机的方法，人们更偏爱易于记忆的一种，而路由器更喜欢长度固定，层次严格的 IP 地址，为了调和这些偏好，我们需要一种目录服务来把 hostname 转化成 IP 地址，这就是 DNS（domain name system）的主要任务，
The DNS is (1) a distributed database implemented in a hierarchy of DNS servers, and (2) an application-layer protocol that allows hosts to query the distributed database.
DNS 服务器一般是运行 BIND（Berkeley Internet Name Domain）软件的 UNIX 机器，DNS 协议运行在 UDP 协议之上并且使用 53 端口
DNS 通常是被别的应用层协议雇佣——包括 HTTP，SMTP , FTP ,来将用户提供的 hostname 转换成 IP 地址，举个栗子，想想看浏览器访问 www.someschool.edu/index.html,为了能使用户的计算机向 Web server 发送 HTTP request，用户的计算机必须获得这个 URL 的 IP 地址。

下面就具体看下 hostname-to-IP-adress 的转换服务
假设用户主机上的一些应用需要将 hostname 转换成 IP 地址，那么这个应用得调用 client side of DNS，得到需要转换的 hostname，然后 DNS 就会向网络发送一个 query，所有的 DNS query 都是用 UDP datagram 在 53 端口发送的，经过一个毫秒到秒级的延迟，DNS 会收到回复消息，提供所请求的 mapping，然后它会被送到那个调用 DNS 的应用，因此，在那个应用的角度，DNS 就是个提供简单直接翻译服务的黑箱，但是事实上，这个黑箱很复杂，有很多分布在全球的 DNS 服务器组成，就像一个具体阐明 DNS 服务器是怎么与发送 query 的计算机交流的应用层协议
设计一个简单的 DNS，要有个 DNS 服务器，里面有所有的 mapping，这种集中制的设计，clients 直接把所有的 query 都发送到这个单独的 DNS 服务器，这个服务器也是直接将回复发送给 clients，虽然这种简单的设计很有吸引力，但是对今天的互联网是不适合的，由于今天的 hosts 数量非常庞大，所以这种设计的主要问题如下：
1）A single point of failure：如果 DNS 服务器崩溃了，那整个互联网怎么办，
2）Traffic volume，一个服务器得掌控所有的 DNS query（所有的 HTTP request 和从成千上万的计算机上聚集来的 email messages）
3）Distant centralized database：只有一个服务器意味着不可能离每个 client 都很近，如果把这个服务器放在纽约，那所有的澳洲的 query 必须穿过大半个地球，或许这时候线路还很拥堵，那么就会造成很大的 delay
4）Maintenance ：这个单独的服务器得保存互联网中所有主机的记录，不管是这会造成数据库非常大，而且为了让所有的计算机都包含进来得频繁更新。

2.6 P2P 应用
client-server file distribution，这种架构中，server 必须向每一个 peer 发送文件——给服务器造成极大的压力并消耗掉非常大的服务器带宽，在 P2P 架构中，任何一个 peer 都可以把它已经接受到的文件中的任何部分再次发送给其他的 peer，因此可以在分发环节帮助服务器缓解压力。
BitTorrent is a popular P2P protocol for file distribution，在它的语言中，很多 peers 都参与一个具体文件的传输叫做 torrent，洪流中的计算机从彼此那里下载相同大小的 chunks，一般 256 KB，当一个计算机先加入到 torrent 时，她没有 chunk，然后随时间它不断有越来越多的 chunks，而且在它下载 chunk 的时候它也会把自己有的 chunk 上传给其他计算机，当一个 peer 收到了这整个文件的时候，它可能会离开也可能会继续在洪流中把 chunks 分发给其他计算机，而且，任何时间都可能有任何计算机带着一部分或者全部整个文件离开，也可能会有再次加入
由于这是个很复杂的协议和系统，所以我们只讲它最重要的机能，多有的 torrent 都有一个 infrastructure node 叫做 tracker，当一个计算机加入到洪流中，它会向 tracker 注册并且周期性的告知自己还在洪流中，这种条件下，tracker 就可以追踪洪流中的 peers。