1 应用层协议原理

网络应用程序有两个例子，一个是 web 应用程序，另一个是 P2P 文件共享系统。

web 应用程序中包含着浏览器程序和 web 服务器程序。浏览器程序运行在用户主机上，web 服务器程序运行在 web 服务器主机上。

P2P 文件共享系统在参与文件共享的社区中的每台主机中都有一个程序。

1.1 网络应用程序体系结构

网络应用程序体系结构有两种主流的结构，一个是客户-服务器体系结构，还有一个是对等 (P2P) 体系结构。

1.1.1 客户-服务器体系结构

这个结构有两个特征：

1. 有一个总是打开的主机称为服务器，它服务于来自许多其他称为客户的主机的请求。
2. 该服务器具有固定的、周知的地址，该地址成为 IP 地址。

1.1.2 P2P 体系结构

在一个 P2P 体系结构中，对位于数据中心的专用服务器有最小的依赖。应用程序在间接连接的主机对之间使用直接通信，这些主机对被称为对等方。这种对等方通信不必通过专门的服务器。

P2P 体系结构具有自扩展性，尽管每个对等方都由于请求文件产生工作量，但每个对等方通过向其他对等方分发文件也为系统增加了服务能力。

P2P 体系结构面临着 3 个主要挑战：

1. ISP 友好，P2P 应用需要设计对 ISP 友好的模式。
2. 安全性，P2P 应用高度分布和开放，所以不安全。
3. 激励，只有用户资源自愿向应用提供带宽、存储和计算资源，P2P 应用才能成功。

1.2 进程通信

在两个不同端系统上的进程，通过跨越计算机网络交换报文而相互通信。发送进程生成并向网络中发送报文，接收进程接收这些报文并可能通过将报文发送回去进行相应。

1.2.1 客户和服务器进程

对每对通信进程，我们通常将这两个进程之一标识为客户，而另一个进程标识为服务器。发起通信的进程被标识为客户，在会话开始时等待联系的进程是服务器。

1.2.2 进程与计算机网络之间的接口

从一个进程向另一个进程发送的报文必须通过下面的网络。进程通过一个称为套接字的软件接口向网络发送报文和从网络接收报文。

套接字是同一台主机内应用层与运输层之间的接口，由于该套接字是建立网络应用程序的可编程接口，因此套接字也称为应用程序和网络之间的应用程序编程接口。

1.2.3 进程寻址

在一台主机上的进程为了向在另一台主机上运行的进程发送分组，接收进程需要有一个地址。为了标识该接收进程，需要定义两种信息：1. 主机的地址；2. 定义在目的主机中的接收进程的标识符。

在因特网中，主机地址由 IP 地址标识。

目的地端口号用于定义在目的主机中的接收进程，比如说 Web 服务器用端口号 80 来标识。

1.3 可供应用程序使用的运输服务

一个运输层协议必须为它的应用程序提供四种服务：可靠数据传输、吞吐量、定时和安全性。

1.3.1 可靠数据传输

运输层确保由应用程序的一端发送的数据正确、完全地交付给该应用程序的另一端。如果应用层协议提供了这样的确保数据交付服务，就认为提供了可靠数据传输。

1.3.2 吞吐量

可用吞吐量就是发送进程能够向接收进程交付比特的速率。运输层协议能够以某种特定的速率提供确保的可用吞吐量。

1.3.3 定时

运输层协议能够提供定时保证，比如发送方注入进套接字中的每个比特到达接收方的套接字不迟于 100ms。

1.3.4 安全性

运输层协议能够为应用程序提供一种或多种安全性服务。比如，在发送主机中，运输协议能够加密由发送进程传输的所有数据，在接收主机中，运输层协议能够在数据交付给接收进城之前解密这些数据。

1.4 因特网提供的运输服务

因特网为应用程序提供两个运输层协议，即 UDP 和 TCP。

1.4.1 TCP 服务

当应用程序调用 TCP 作为其运输协议时，TCP 将提供两种服务:

1. 面向连接的服务：在应用层数据报文开始流动之前，TCP 让客户和服务器经历一个握手过程，互相交换运输层控制信息。在握手过程之后，一个 TCP 连接就在两个进程的套接字之间建立了。
2. 可靠的数据传送服务：通信进程能够依靠 TCP，无差错、按适当顺序交付所有发送的数据。

TCP 还有拥塞控制机制，当发送方和接收方之间的网络出现拥塞时，TCP 的拥塞控制机制会抑制发送进程。

1.4.2 UDP 服务

UDP 服务没有握手过程，也没有 UDP 连接，没有拥塞控制机制，仅提供最小服务。

当进程将一个报文发送进 UDP 套接字时，UDP 协议并不保证该报文将到达接收进程，而且到达接收进程的报文也可能是乱序到达的。

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。应用层协议定义了如下内容:

1. 交换的报文类型，例如请求报文和响应报文。
2. 各种报文类型的语法，如报文中的各个字段及这些字段是如何描述的。
3. 字段的语义，即这些字段中包含的信息的含义。
4. 一个进程何时以及如何发送报文，对报文进行响应的规则。

2 Web 和 HTTP

2.1 HTTP 概况

Web 应用的应用层协议是 HTTP，也就是超文本传输协议。

web 客户程序和 web 服务器程序运行在不同的端系统中，通过交换 HTTP 报文进行会话。HTTP 定义了这些报文的结构以及客户和服务器进行报文交换的方式。

2.1.1 一些 web 术语

一个 Web 页面是由对象组成的，一个对象就是一个文件，通过 URL 地址寻址得到。

每个 URL 地址由两部分组成：存放对象的服务器主机名和对象的路径名。比如，URL 地址为 http://www.someSchool.edu/someDepartment/picture.gif，其中 www.someSchool.edu 是主机名，/someDepartment/picture.gif 是路径名。

web 浏览器实现了 HTTP 的客户端，web 服务器实现了 HTTP 的服务器端，它用于存储 web 对象，每个对象由 URL 寻址。

2.1.2 HTTP 的作用

HTTP 定义了 web 客户向 web 服务器请求 web 页面的方式，以及服务器向客户传送 web 页面的方式。

HTTP 使用 TCP 作为它的支撑运输协议，HTTP 客户首先发起一个与服务器的 TCP 连接，一旦连接建立，该浏览器和服务器进程就可以通过套接字接口访问 TCP。客户向它的套接字接口发送 HTTP 请求报文并从它的套接字接口接收 HTTP 响应报文，服务器从它的套接字接口接收 HTTP 请求报文并从它的套接字接口发送 HTTP 响应报文。

2.1.3 无状态协议

HTTP 服务器向客户发送被请求的文件，而不存储任何关于该客户的状态信息，所以我们说 HTTP 是一个无状态协议。

假如某个特定的客户在短短的几秒钟内两次请求同一个对象，服务器并不会因为刚刚为该客户提供了该对象就不再做出反应，而是重新发送该对象，就像服务器已经完全忘记不久之前所做过的事一样。

2.2 请求 html 文件的时间

在这里将估算一下从客户请求 HTML 基本文件起到该客户收到整个文件止所花费的时间。

首先定义往返时间 RTT：一个短分组从客户机到服务器然后再返回客户所花费的时间。RTT 包括分组传输时延、分组在中间路由器和交换机上的排队时延以及分组处理时延。

HTTP 客户向 HTTP 请求 HTML 文件时，需要发起 TCP 连接，然后请求文件，最后服务器发送 HTML 文件。发起 TCP 连接和请求文件各占一个 RTT 时间，所以总的时间为两个 RTT 加上服务器传输 HTML 文件的时间。

2.3 非持续连接和持续连接

在客户和服务器相互通信的过程中，客户发出一系列请求并且服务器对每个请求进行响应。对于非持续连接的 HTTP，每个请求-响应对经过一个单独的 TCP 连接发送。而对于持续连接的 HTTP，所有请求-响应对都经过相同的 TCP 连接发送。

2.3.1 非持续连接的 HTTP

HTTP 客户向 HTTP 服务器请求一个 web 页面，该 web 页面包含一个 html 文件和 10 个 jpeg 图形，该 html 文件的 url 为：http://www.someSchool.edu/someDepartment/home.index。步骤如下：

1. HTTP 客户进程在端口号 80 发起一个到服务器 www.someSchool.edu 的 TCP 连接，随后在客户和服务器上分别有一个套接字与该连接相关联。
2. HTTP 客户经它的套接字向服务器发送一个 HTTP 请求报文，其中包含 html 文件的路径/someDepartment/home.index。
3. HTTP 服务器进程经它的套接字接收该请求报文，根据路径名得到 html 文件。在一个 http 响应报文中封装对象，并通过套接字向客户发送响应报文。
4. HTTP 服务器进程通知 TCP 断开该 TCP 连接。
5. HTTP 客户接收响应报文，TCP 连接关闭。客户从响应报文中提取出该文件，检查该 html 文件，得到对 10 个 jpeg 图形的引用。
6. 对每个引用的 jpeg 图形对象重复前 4 个步骤。

非持续连接的缺点为：

1. 必须为每一个请求的对象建立和维护一个全新的连接。
2. 每一个对象经受两倍 RTT 的交付时延，即一个 RTT 用于创建 TCP，另一个 RTT 用于请求和接收一个对象。

2.3.2 持续连接的 HTTP

在采用持续连接的情况下，服务器在发送响应后保持该 TCP 连接打开。在相同的客户与服务器之间的后续请求和响应报文能够通过相同的 TCP 连接进行传送。

2.4 HTTP 报文格式

HTTP 报文有两种：请求报文和响应报文。

2.4.1 HTTP 请求报文

一个典型的 HTTP 请求报文：

```
GET /somedir/page.html HTTP/1.1\r\n
Host: www.someschool.edu\r\n
Connection: close\r\n
User-agent: Mozilla/5.0\r\n
Accept-language: fr\r\n
\r\n
```

HTTP 请求报文的第一行叫做请求行，后继的行叫做首部行：

- 请求行有 3 个字段：方法字段、URL 字段和 HTTP 版本字段。方法字段有：GET、POST、HEAD、PUT 和 DELETE。
- 首部行中，Host 用于指定对象所在的主机，Connection 用于指定这条连接在请求响应结束后是否关闭，User-agent 用于指定浏览器的类型，Accept-language 用于指定对象的哪个语言版本。

各个方法：

- POST，将使用一个“entity body”。使用 POST 报文时，用户可以向服务器请求一个 web 页面，这个 web 页面的特定内容依赖于用户在表单字段中输入的内容。entity body 中包含的就是用户在表单字段中的输入值。
- GET，此时 entity body 为空。不过 HTML 表单也经常使用 GET 方法，它会在所请求的 URL 中包含输入的数据。
- HEAD，类似于 GET 方法。当服务器收到使用 HEAD 方法的请求时，将会用一个 HTTP 报文进行响应，但是并不返回请求对象。
- PUT，允许用户上传对象到指定的 web 服务器上指定的路径。
- DELETE，允许用户或者应用程序删除 web 服务器上的对象。

2.4.2 HTTP 响应报文

一个典型的 HTTP 响应报文：

```
HTTP/1.1 200 OK\r\n
Connection: close\r\n
Data: Tue, 09 Aug 2011 15:44:04 GMT\r\n
Server: Apache/2.2.3(CentOS)\r\n
Last-Modified: Tue, 09 Aug 2011 15:11:03 GMT\r\n
Content-length: 6821\r\n
Content-Type: text/html\r\n
\r\n
(data data data data data ...)
```

响应报文的第一行为状态行，还有 6 个首部行，最后是 entity body：

- 状态行，包含协议版本字段、状态码和相应状态信息。
- 首部行中，Connection：close 告诉客户发送完报文后关闭该 TCP 连接，Data 用于指示服务器产生并发送该响应报文的日期和时间。Server 用于指定服务器类型，Last-Modified 用于指定对象创建或最后修改的日期和时间，Content-Length 用于指定发送对象的字节数，Content-Type：用于指定 entity body 中的对象类型。

一些常见的状态码和相应状态信息：

- 200 OK：请求成功。
- 301 Moved Permanently：请求的对象已经被永久转移了。新的 URL 定义在响应报文的 Location 中。
- 400 Bad Request：一个通用差错代码。
- 404 Not Found：被请求的文档不在服务器上。
- 505 HTTP Version Not Supported：服务器不支持请求报文使用的 HTTP 协议版本。

2.5 用户与服务器的交互

cookie 技术有 4 个组件：

- HTTP 响应报文中的一个 Set-cookie 首部行。
- HTTP 请求报文中的一个 Cookie 首部行。
- 在用户端系统中保留有一个 cookie 文件，并由用户的浏览器进行管理。
- 在服务器端有一个后端数据库。

课本的第 72 ～ 73 页有一个生动的例子。

2.6 Web 缓存器

Web 缓存器也叫代理服务器，它是能够代表初始化 Web 服务器来满足 HTTP 请求的网络实体。

Web 缓存器有自己的磁盘存储空间，并在存储空间中保存最近请求过的对象的副本。可以配置浏览器，使得 HTTP 请求首先指向 web 缓存器。如果 web 缓存器没有该对象，web 缓存器会向初始服务器请求对象，然后存储一份副本，并向客户发送该副本。

Web 缓存器的优点：

- web 缓存器可以大大减少对客户请求的响应时间。
- web 缓存器能够大大减少一个机构的接入链路到因特网的通信量。通过减少通信量，该机构就不必急于增加带宽，因此降低了费用。

课本的第 75 页通过一个例子介绍了 web 缓存器的优点。

2.7 条件 GET 方法

条件 GET 方法可以让缓存器证实它的对象是最新的，条件 GET 方法要求：

- 请求报文使用 GET 方法。
- 请求报文中包含一个“If-Modified-Since”的首部行。

课本的第 76 ～ 77 页有一个例子介绍了这个方法的使用。

3 文件传输协议：FTP

FTP 的步骤如下：

1. 用户首先提供远程主机的主机名，使本地主机的 FTP 客户进程建立一个到远程主机的 FTP 服务器进程的 TCP 连接。
2. 该用户接着提供用户标识和口令，作为 FTP 命令的一部分在该 TCP 连接上传送。
3. 一旦该服务器向该用户授权，用户可以将存放在本地文件系统中的一个或者多个文件复制到远程文件系统。

FTP 使用了两个并行的 TCP 连接：

- 控制连接，用于在两主机之间传输控制信息。
- 数据连接，用于实际发送一个文件。

因为 FTP 使用一个独立的控制连接，所以 FTP 的控制信息是带外的。而 HTTP 协议在同一个 TCP 连接中发送请求和响应首部行，所以是带内的。

FTP 服务器必须在整个会话期间保留用户的状态，也就是对每个进行中的用户会话的状态信息进行追踪。

3.1 FTP 命令和回答

一些较为常见的命令如下：

- USER username，用于向服务器传送用户标识。
- PASS password，用于向服务器发送用户口令。
- LIST，用于请求服务器回送当前远程目录中的所有文件列表。
- RETR filename，用于从远程主机当前目录获取文件。
- STOR filename，用于在远程主机的当前目录存放文件。

一些典型的回答如下：

- 331 Username OK, Password required
- 125 Data connection already open; transfer starting
- 425 Can’t open data connection
- 452 Error writing file

4 因特网中的电子邮件

因特网电子邮件系统的 3 个主要组成部分：用户代理、邮件服务器和简单邮件传输协议。

一个典型的邮件发送过程是：从发送方的用户代理开始，传输到发送方的邮件服务器，在传输到接收方的邮件服务器，然后在这里被分发到接收方的邮箱中。

4.1 简单邮件传输协议 SMTP

SMTP 用于从发送方的邮件服务器发送报文到接收方的邮件服务器。

邮件服务器的端口号是 25。

SMTP 的工作过程：SMTP 客户端发现发送方邮件服务器中报文队列中的这个报文，然后创建一个到接收方邮件服务器上的 SMTP 服务器的 TCP 连接。经过一些初始 SMTP

握手后，SMTP 客户通过该 TCP 连接发送报文。在接收方邮件服务器上，SMTP 的服务器端接收该报文。在握手阶段，SMTP 客户端会指示发送方的邮件地址和接收方的邮件地址。

SMTP 不使用中间邮件服务器发送邮件。

SMTP 有一些常用的命令：HELO、MAIL FROM、RCPT TO、DATA 以及 QUIT。在课本的第 82 页有一个 SMTP 工作的例子。

对于每个报文，客户通过 MAIL FROM 开始，用一个独立的句点“.”指示该邮件的结束，并且仅当所有邮件发送完后才发送 QUIT。

4.2 邮件报文格式

一个邮件报文由首部、空行和报文体组成。

每个首部必须含有一个 From，一个 To，有时候还可能包含一个 Subject。一个典型的报文首部如下：

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Searching for the meaning of life
```

4.3 邮件访问协议

我们通过邮局访问协议将邮件服务器上的报文传送回本地 PC。流行的邮件访问协议有：第三版的邮局协议 POP3、因特网邮件访问协议 IMAP 以及 HTTP。

4.3.1 POP3

POP3 按照三个阶段进行工作：

- 特许阶段，用户代理发送用户名和口令来鉴别用户。
- 事务处理阶段，用户代理取回报文，同时用户代理还能对报文做删除标记、取消报文删除标记，以及获取邮件的统计信息。
- 更新阶段，在客户发出 quit 命令后开始，目的是结束该 POP3 会话。此时，邮件服务器删除那些被标记为删除的报文。

服务器对用户代理的命令有两种：

- +OK，指示命令正常。
- -ERR，指示命令出现了某些差错。

特许阶段有两个命令：user <username> 和 pass <password>，例子如下：

```
telnet mailServer 110
user bob
pass hungry
```

事务处理阶段有四个命令：list、retr、dele 和 quit。

4.3.2 IMAP

IMAP 的两个重要特性：

1. IMAP 服务器把每个报文与一个文件夹联系起来，并且维护了 IMAP 会话的用户状态信息。
2. IMAP 允许用户代理获取报文组件的命令，也就是说，一个用户代理可以只读区一个报文的报文首部，或只是一个多部分 MIME 报文的一部分。

5 DNS: 因特网的目录服务

5.1 DNS 提供的服务

识别主机有两种方式：通过主机名或者 IP 地址。
域名系统 DNS 的作用：进行主机名到 IP 地址转换的目录服务。
DNS 的组成：

1. 一个由分层的 DNS 服务器实现的分布式数据库。
2. 一个使得主机能够查询分布式数据库的应用层协议。

DNS 协议运行在 UDP 上，使用 53 号端口。
课本第 88 页上有一个 DNS 服务的例子。
DNS 提供的服务：

- 主机别名，有着复杂主机名的主机能拥有一个或者多个别名。
- 邮件服务器别名。电子邮件应用程序可以调用 DNS，对提供的邮件服务器别名进行解析，以获得该主机的规范主机名及其 IP 地址。
- 负载分配，一个 IP 地址集合与同一个规范主机名相联系。

5.2 DNS 工作机理概述

5.2.1 集中式 DNS 服务器

集中式 DNS 服务器的缺点:

- 单点故障
- 通信容量
- 远距离的集中式数据库
- 维护

5.2.2 分布式、层次 DNS 服务器

域名 DNS 采用分布式、层次数据库，分为三层:

1. 第一层是根 DNS 服务器，用于返回顶级域名 DNS 服务器的 IP 地址。
2. 第二层是顶级域 DNS 服务器，负责 com、org、net 等顶级域名，用于返回相应权威 DNS 服务器的 IP 地址。
3. 第三层是权威 DNS 服务器，用于返回一个组织机构的 IP 地址。

书本第 91 页有分布式、层次数据库工作的例子。

5.2.3 本地 DNS 服务器

当主机发出 DNS 请求时，这个请求是发往本地 DNS 服务器，本地 DNS 服务器起着代理的作用，并将请求转发到 DNS 服务器层次结构中，获得相应 IP 地址后，再返回给主机。

书本第 92 页有使用本地 DNS 服务器的例子。

5.2.4 DNS 缓存

在一个请求链中，当某 DNS 服务器接收一个 DNS 回答时，它能将该回答中的信息缓存在本地存储器中，这就是 DNS 缓存。

DNS 服务器在一段时间后将丢弃缓存的信息。

5.3 DNS 记录和报文

5.3.1 DNS 资源记录

DNS 服务器中存储了资源记录，用于提供主机名到 IP 地址的映射。
资源记录的数据结构是一个 4 元组：(Name, Value, Type, TTL)：

- TTL 是这条记录的生存时间，决定了资源记录应当从缓存中删除的时间。
- Name 和 Value 取决于 Type 的值：
  - 如果 Type=A，则 Name 是主机名，Value 是该主机名对应的 IP 地址。
  - 如果 Type=NS，则 Name 是一个域名，Value 是一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。
  - 如果 Type=CNAME，则 Name 是一个主机别名，Value 是一个规范主机名。
  - 如果 Type=MX，则 Name 是一个邮件服务器别名，Value 是一个邮件服务器的规范主机名。

5.3.2 DNS 报文

DNS 有两种报文：DNS 查询报文和 DNS 回答报文。
DNS 报文格式如下：

- 前 12 个字节是首部区域，有 6 个字段：
  - 标识符，用于标识该查询。
  - 标志字段，含有若干标志：“查询／回答”标志位、“希望递归”标志位等。
  - 其他 4 个有关数量的字段：问题数、回答 RR 数、权威 RR 数和附加 RR 数。这些字段指出了在首部后的 4 类数据区域出现的数量。
- 问题区域，包含：名字字段，指出正在被查询的主机名字；类型字段，也就是资源记录中的 Type 值。
- 回答区域，包含了对最初请求的名字的资源记录。
- 权威区域，包含了其他权威服务器的记录。
- 附加区域，包含了其他有帮助的记录。

5.4 在 DNS 数据库中插入记录

注册域名机构用于将域名输入 DNS 数据库，也就是在 DNS 数据库中插入记录。

当某个机构向注册域名机构注册域名时，需要提供基本和辅助权威 DNS 服务器的主机名和 IP 地址。随后注册登记机构将一个类型 NS 和一个类型 A 的记录输入相应的所有的 TLD 顶级域名 DNS 服务器中。

6 P2P 应用

P2P 体系结构对总是打开的基础设施服务器有着最小的依赖。

有两种特别适合于 P2P 设计的应用：

- 文件分发，其中应用程序从单个源向大量的对等方分发一个文件。
- 分布在大型对等方社区中的数据库。

6.1 P2P 文件分发

6.1.1 BitTorrent

BitTorrent 是一种用于文件分发的流行 P2P 协议。

洪流 torrent：参与一个特定文件分发的所有对等方的集合。

追踪器：当一个对等方加入某洪流时，它向追踪器注册自己，并周期性地通知追踪器它仍在该洪流中，追踪器跟踪正参与在洪流中的对等方。

稀缺优先技术：针对一个主机没有的块，在它的邻居中决定最稀缺的块，并首先请求那些最稀缺的块。最稀缺的块就是那些在它的邻居中副本数量最少的块。

BitTorrent 中的对换算法 (一报还一报)：一台主机有接收比特速率前 4 个对等方和一个试探的对等方。前 4 个对等方称为疏通，该主机每 10 秒重新计算一次接收比特速率并修改这 4 个对等方的集合。该主机每过 30 秒随机选择一名新的对换伴侣并开始与对方进行对换，如果双方接收速率都排前 4，它们将对方放入前 4 位列表中并继续与对方进行兑换，直到双方中的一个发现了一个更好的伴侣为止。需要知道的是，除了上述的 5 个对等方，其他的邻居都不能从这台主机接收到任何块。

课本第 101 页有 BitTorrent 的工作流程的例子。

6.2 分布式散列表

分布式散列表：一个分布式数据库，该数据库只包含键值对，它在数以百万计的对等方上存储键值对，每个对等方将保持键值对仅占总体的一个小子集。当查询分布式数据库时，它将定位拥有相应的键值对的对等方，然后向查询的对等方返回该键值对。

课本第 103 页上有 DHT 数据库应用的一个例子。

构建 DHT 的一种幼稚的方法：

1. 首先让每个对等方维护一个所有对等方的列表。
2. 然后向所有的对等方随机地散布键值对。
3. 实际查询时，查询的对等方向所有其他对等方发送它的查询，并且包含相应键值对的对等方能对其作出反应。

设计 DHT 的一种精确有效的方法：首先为对等方分配键值对：给定每个对等方一个整数标识符，每个键也是一个位于相同范围的整数。分配键值对的原则：为最邻近一个键的对等方分配一个键值对。

最邻近的定义：如果键小于标识符，就采取最邻近后继；如果键等于多个标识符，就在匹配的对等方中存储键值对；如果键大于所有标识符，对所有标识符进行模 $\mathbf{2^n}$，在具有最小标识符的对等方中存储键值对。

确定一个键的最近邻对等方的方法：

1. 环形 DHT
2. 对等方扰动

6.2.1 环形 DHT

环形 DHT 中每个对等方仅与它的直接后继和直接前任联系。这种情况下，为了找到相应键值对的对等方，平均需要发送 N/2 条报文。

而如果每个对等方都关联了其他所有对等方，那么每次查询仅需要发送一个报文。

研究表明 DHT 能被设计成每个对等方的邻居数量以及每个请求的报文数量均为 $O(\log N)$。

6.2.2 对等方扰动

对等方扰动：对等方不加警示地到来和离去。

为处理对等方扰动，我们要求每个对等方联系其第一个和第二个候机，并周期性地证实它的两个后继是存活的。

在课本的第 105 页有处理对等方突然离去的例子，课本的第 106 页有新的对等方加入 DHT 的例子。