15-213,20XX 年秋季

代理实验室:编写缓存 Web 代理

分配时间:11 月 19 日,星期四

截止日期:12 月 8 日,星期二,晚上 11:59

最后提交时间:12 月 11 日星期五晚上 11:59

1 简介

Web 代理是充当 Web 浏览器和终端服务器之间中间人的程序。浏览器不是直接联系终端服务器来获取网页,而是联系代理,代理将请求转发到终端服务器。当终端服务器回复代理时,代理将回复发送到浏览器。

代理有多种用途。有时在防火墙中使用代理,以便防火墙后面的浏览器只能通过代理联系防火墙之外的服务器。代理还可以充当匿名器:通过剥离所有标识信息的请求,代理可以使浏览器对 Web 服务器匿名。代理甚至可以用于缓存 Web 对象,方法是存储来自服务器的对象的本地副本,然后通过从缓存中读取它们来响应未来的请求,而不是再次与远程服务器通信。

在本实验中,您将编写一个简单的 HTTP 代理来缓存 Web 对象。在本实验的第一部分中,您将设置代理来接受传入连接、读取和解析请求、将请求转发到 Web 服务器、读取服务器的响应,并将这些响应转发到相应的客户端。第一部分将涉及学习基本的 HTTP 操作以及如何使用套接字编写通过网络连接进行通信的程序。在第二部分中,您将升级代理以处理多个并发连接。这将向您介绍如何处理并发,这是一个重要的系统概念。在第三部分也是最后一部分中,您将使用最近访问的 Web 内容的简单主内存缓存向代理添加缓存。

2 物流

这是一个个人项目。

3 讲义说明

特定于站点:在此处插入一段内容,解释讲师如何将 proxylab-handout.tar 文件分发给学生。

将讲义文件复制到您计划执行工作的 Linux 计算机上的受保护目录,然后发出以下命令:

```
linux> tar xvf proxylab-handout.tar
```

这将生成一个名为 proxylab-handout 的讲义目录。自述文件描述了各种文件。

4 第一部分:实现顺序 Web 代理

第一步是实现一个处理 HTTP/1.0 GET 请求的基本顺序代理。其他请求类型（例如 POST)是严格可选的。

启动后,您的代理应该侦听端口号将在命令行上指定的端口上的传入连接。建立连接后,您的代理应该从客户端读取整个请求并解析该请求。应该判断客户端是否发送了有效的HTTP请求;如果是这样,它就可以建立自己到适当 Web 服务器的连接,然后请求客户端指定的对象。最后,您的代理应该读取服务器的响应并将其转发给客户端。

4.1 HTTP/1.0 GET 请求

当最终用户在 Web 浏览器的地址栏中输入诸如 http://www.cmu.edu/hub/index.html 之类的 URL 时,浏览器将向代理发送一个 HTTP 请求,该请求以以下行开头:类似于以下内容:

```
获取 http://www.cmu.edu/hub/index.html HTTP/1.1
```

在这种情况下,代理应该将请求解析为至少以下字段:主机名、www.cmu.edu;以及路径或查询及其后面的所有内容，/hub/index.html。这样,代理可以确定它应该打开到 www.cmu.edu 的连接并发送自己的 HTTP 请求,该请求以以下形式的行开头:

```
获取 /hub/index.html HTTP/1.0
```

请注意,HTTP 请求中的所有行均以回车符“\r”结尾,后跟换行符“\n”。同样重要的是,每个 HTTP 请求都以空行终止:“\r\n”。

您应该注意到,在上面的示例中,Web 浏览器的请求行以 HTTP/1.1 结尾,而代理的请求行以 HTTP/1.0 结尾。现代 Web 浏览器将生成 HTTP/1.1 请求,但您的代理应该处理它们并将它们作为 HTTP/1.0 请求转发。

重要的是要考虑到 HTTP 请求,即使只是 HTTP/1.0 GET 请求的子集,也可能非常复杂。教科书描述了 HTTP 事务的某些细节,但您应该参考 RFC 1945 以获取完整的 HTTP/1.0 规范。理想情况下,根据 RFC 1945 的相关部分,您的 HTTP 请求解析器将完全健壮,但有一个细节除外:虽然规范允许多行请求字段,但您的代理不需要正确处理它们。当然,您的代理永远不应该由于格式错误的请求而提前中止。

4.2 请求头

本实验中重要的请求标头是 Host、User-Agent、Connection 和 Proxy-Connection 标头:

- 始终发送主机标头。虽然这种行为在技术上并未受到 HTTP/1.0 规范的认可,但有必要从某些 Web 服务器(尤其是使用虚拟主机的 Web 服务器)中诱导出合理的响应。

  Host 标头描述了终端服务器的主机名。例如访问http://www. cmu.edu/hub/index.html,您的代理将发送以下标头:

  主办单位:www.cmu.edu

  Web 浏览器可能会将自己的主机标头附加到其 HTTP 请求中。如果是这种情况,您的代理应该使用与浏览器相同的主机标头。

- 您可以选择始终发送以下用户代理标头:

  用户代理:Mozilla/5.0(X11;Linux x86_64;rv:10.0.3)
  壁虎/20120305 火狐/10.0.3

  标头在两个单独的行中提供,因为它不适合作为书面文件中的单行,但您的代理应将标头作为单行发送。

  User-Agent 标头标识客户端(根据操作系统和浏览器等参数),Web 服务器通常使用标识信息来操作它们所服务的内容。发送这个特定的 User-Agent: 字符串可能会改善您在简单的 telnet 式测试期间返回的材料的内容和多样性。

- 始终发送以下连接标头:

  连接:关闭

- 始终发送以下代理连接标头:

代理连接:关闭

Connection 和 Proxy-Connection 标头用于指定在第一次请求/响应交换完成后连接是否保持活动状态。让您的代理为每个请求打开一个新连接是完全可以接受的（也是建议的）。指定 close 作为这些标头的值会提醒 Web 服务器您的代理打算在第一次请求/响应交换后关闭连接。

为了您的方便,所描述的 User-Agent 标头的值在 proxy.c 中作为字符串常量提供给您。

最后,如果浏览器发送任何额外的请求标头作为 HTTP 请求的一部分,您的代理应将它们原封不动地转发。

4.3 端口号

本实验有两类重要的端口号:HTTP 请求端口和代理的侦听端口。

HTTP 请求端口是 HTTP 请求 URL 中的可选字段。也就是说,URL 的形式可能是 http://www.cmu.edu:8080/hub/index.html,在这种情况下,您的代理应该连接到端口 8080 上的主机 www.cmu.edu,而不是默认 HTTP 端口,即端口 80。无论 URL 中是否包含端口号,您的代理都必须正常运行。

侦听端口是代理应侦听传入连接的端口。您的代理应该接受指定代理侦听端口号的命令行参数。例如,使用以下命令,您的代理应侦听端口 15213 上的连接:

Linux> ./代理 15213

您可以选择任何非特权侦听端口（大于 1,024 且小于 65,536）,只要它不被其他进程使用即可。由于每个代理必须使用唯一的侦听端口,并且许多人将同时在每台计算机上工作,因此提供了脚本 port-for-user.pl 来帮助您选择自己的个人端口号。使用它根据您的用户 ID 生成端口号:

linux> ./port-for-user.pl droh droh:45806

port-for-user.pl 返回的端口 p 始终为偶数。因此,如果您需要额外的端口号（例如 Tiny 服务器）,您可以安全地使用端口 p 和 p + 1。

请不要随意选择您自己的端口。如果这样做,您将面临干扰其他用户的风险。

5 第二部分:处理多个并发请求

一旦你有了一个工作的顺序代理,你应该改变它以同时处理多个请求。
实现并发服务器的最简单方法是生成一个新线程来处理每个新的连接请求。其他设计也是可能的,例如教科书第 12.5.5 节中描述的预线程服务器。

- 请注意,您的线程应在分离模式下运行以避免内存泄漏。
- CS:APP3e 教科书中描述的 open clientfd 和 open Listenfd 函数基于现代且与协议无关的 getaddrinfo 函数,因此是线程安全的。

6 第三部分:缓存 Web 对象

在本实验的最后部分,您将向代理添加一个缓存,用于将最近使用的 Web 对象存储在内存中。 HTTP 实际上定义了一个相当复杂的模型,Web 服务器可以通过该模型给出有关如何缓存它们所服务的对象的指令,并且客户端可以指定如何代表它们使用缓存。但是,您的代理将采用简化的方法。

当您的代理从服务器接收到 Web 对象时,它应该在将该对象传输到客户端时将其缓存在内存中。如果另一个客户端从同一服务器请求相同的对象,您的代理不需要重新连接到服务器;它可以简单地重新发送缓存的对象。

显然,如果您的代理要缓存曾经请求的每个对象,则它将需要无限量的内存。此外,由于某些 Web 对象比其他对象大,因此可能出现这样的情况:一个巨型对象将消耗整个缓存,从而根本无法缓存其他对象。为了避免这些问题,您的代理应该同时具有最大缓存大小和最大缓存对象大小。

6.1 最大缓存大小

整个代理的缓存应具有以下最大大小:

MAX_CACHE_SIZE = 1 MiB

在计算其缓存大小时,您的代理必须仅计算用于存储实际 Web 对象的字节数;任何无关的字节,包括元数据,都应该被忽略。

6.2 最大物体尺寸

您的代理应仅缓存不超过以下最大大小的 Web 对象:

最大对象大小 = 100 KiB

为了您的方便,这两个大小限制都在 proxy.c 中以宏的形式提供。

实现正确缓存的最简单方法是为每个活动连接分配一个缓冲区,并在从服务器接收数据时累积数据。如果缓冲区的大小超过了最大对象大小,则可以丢弃该缓冲区。如果在超过最大对象大小之前读取了整个 Web 服务器的响应,则可以缓存该对象。使用此方案,您的代理将用于 Web 对象的最大数据量如下,其中 T 是最大活动连接数:

MAX_CACHE_SIZE + T * MAX_OBJECT_SIZE

6.3 驱逐政策

您的代理的缓存应采用近似于最近最少使用 (LRU) 逐出策略的逐出策略。它不必是严格的 LRU,但应该相当接近。请注意,读取对象和写入对象都算作使用该对象。

6.4 同步

对缓存的访问必须是线程安全的,并且确保缓存访问不受竞争条件的影响可能是本部分实验中更有趣的方面。事实上,有一个特殊的要求,即多个线程必须能够同时从缓存中读取。当然,一次只能允许一个线程写入缓存,但对于读取者来说,不得存在这种限制。

因此,使用一个大的排它锁来保护对缓存的访问并不是一种可接受的解决方案。您可能想要探索一些选项,例如对缓存进行分区、使用 Pthreads 读写锁或使用信号量来实现您自己的读写解决方案。无论哪种情况,您不必实施严格的 LRU 驱逐策略,这一事实将为您在支持多个读取器方面提供一定的灵活性。

7 评价

该作业的评分为 70 分:

- 基本正确性:基本代理操作 40 分(自动评分)
- 并发性:处理并发请求 15 分(自动评分)
- 缓存:工作缓存 15 分(自动评分)

7.1 自动评分

您的讲义材料包括一个名为 driver.sh 的自动评分器,您的讲师将使用它来为基本正确性、并发性和缓存分配分数。从 proxylab-handout 目录:

Linux> ./driver.sh

您必须在 Linux 计算机上运行该驱动程序。

7.2 鲁棒性

与往常一样,您必须提供一个能够抵御错误甚至格式错误或恶意输入的程序。
服务器通常是长时间运行的进程,Web 代理也不例外。仔细考虑长时间运行的进程应如何应对不同类型的错误。对于多种错误,您的代理立即退出当然是不合适的。

稳健性还意味着其他要求,包括不受分段错误等错误情况的影响以及缺乏内存泄漏和文件描述符泄漏。

8 测试与调试

除了简单的自动评分器之外,您将没有任何示例输入或测试程序来测试您的实现。您将必须提出自己的测试,甚至可能是自己的测试工具来帮助您调试代码并决定何时拥有正确的实现。在现实世界中,这是一项宝贵的技能,因为人们很少知道确切的操作条件,并且通常无法获得参考解决方案。

幸运的是,您可以使用许多工具来调试和测试代理。请务必执行所有代码路径并测试一组有代表性的输入,包括基本案例、典型案例和边缘案例。

8.1 小型网络服务器

您的讲义目录中包含 CS:APP Tiny Web 服务器的源代码。虽然不如 thttpd 强大,但 CS:APP Tiny Web 服务器将很容易让您根据需要进行修改。这也是代理代码的合理起点。它是驱动程序代码用来获取页面的服务器。

8.2远程登录

正如教科书 (11.5.3) 中所述,您可以使用 telnet 打开与代理的连接并向其发送 HTTP 请求。

8.3卷曲

您可以使用curl 生成对任何服务器的HTTP 请求,包括您自己的代理。它是一个非常有用的调试工具。例如,如果您的代理和 Tiny 都在本地计算机上运行,Tiny 正在侦听端口 15213,代理正在侦听端口 15214,那么您可以使用以下curl 命令通过代理向 Tiny 请求页面:

```
linux> curl -v --proxy http://localhost:15214 http://localhost:15213/home.html * 将连接()到代理本地主机端口 15214 (#0)

* 尝试 127.0.0.1...已连接
```

```
* 连接到本地主机 (127.0.0.1) 端口 15214 (#0)
> 获取 http://localhost:15213/home.html HTTP/1.1 > 用户代理:curl/7.19.7 (x86_64-redhat-
linux-gnu)...
> 主机:本地主机:15213
> 接受:*/*
> 代理连接:保持活动状态
>
* HTTP 1.0,假设正文后关闭 < HTTP/1.0 200 OK < 服务器:Tiny Web
Server < 内容长度:120 < 内容类型:
text/html

<
<html>
<head><title>测试</title></head> <body> <imgalign=“middle”

src=“godzilla.gif”>
戴夫·奥哈拉伦
</正文> </
html>
* 关闭连接 #0
```

8.4网络猫

netcat,也称为 nc,是一种多功能网络实用程序。您可以像 telnet 一样使用 netcat 来打开与服务器的连接。因此,假设您的代理使用端口 12345 在 catshark 上运行,您可以执行如下操作来手动测试您的代理:

```
sh> nc catshark.ics.cs.cmu.edu 12345
获取 http://www.cmu.edu/hub/index.html HTTP/1.0

HTTP/1.1 200 好
 ...
```

除了能够连接到 Web 服务器之外,netcat 本身还可以作为服务器运行。使用以下命令,您可以运行 netcat 作为侦听端口 12345 的服务器:

```
sh> nc -l 12345
```

设置完 netcat 服务器后,您可以通过代理生成对其中的虚假对象的请求,并且您将能够检查代理发送到 netcat 的确切请求。

8.5 网络浏览器

最终您应该使用最新版本的 Mozilla Firefox 测试您的代理。访问关于 Firefox 会自动将您的浏览器更新到最新版本。

要将 Firefox 配置为使用代理,请访问

首选项>高级>网络>设置

看到您的代理通过真正的 Web 浏览器工作将是非常令人兴奋的。尽管代理的功能会受到限制,但您会发现您可以通过代理浏览绝大多数网站。

一个重要的警告是,使用 Web 浏览器测试缓存时必须非常小心。所有现代 Web 浏览器都有自己的缓存,您应该在尝试测试代理的缓存之前将其禁用。

9 上交须知

提供的 Makefile 包含构建最终交接文件的功能。从您的工作目录发出以下命令:

```
linux> 进行交接
```

输出是文件 ../proxylab-handin.tar,然后您可以提交该文件。

特定于站点:在此处插入一个段落,告诉每个学生如何提交他们的 proxylab-handin.tar 解决方案文件。

- 教科书第10-12章包含有关系统级I/O、网络程序的有用信息。
  ming、HTTP 协议和并发编程。
- RFC 1945 (http://www.ietf.org/rfc/rfc1945.txt) 是完整的规范
  HTTP/1.0 协议。

10 条提示

- 正如教科书第10.11 节中所讨论的,使用标准I/O 函数进行套接字输入和输出是一个问题。相反,我们建议您使用 Robust I/O (RIO) 包,该包在讲义目录中的 csapp.c 文件中提供。

- csapp.c 中提供的错误处理函数不适合您的代理,因为一旦服务器开始接受连接,就不应终止。您需要修改它们或编写自己的。

- 您可以随意修改讲义目录中的文件。例如,为了实现良好的模块化,您可以将缓存函数实现为名为cache.c 和cache.h 的文件中的库。当然,添加新文件将需要您更新提供的 Makefile。

- 正如CS:APP3e 文本第964 页旁注中所讨论的,您的代理必须忽略SIGPIPE 信号,并且应该妥善处理返回EPIPE 错误的写入操作。

- 有时,调用read 从已过早关闭的套接字接收字节将导致read 返回-1,并将errno 设置为ECONNRESET。您的代理也不应该由于此错误而终止。

- 请记住,并非网络上的所有内容都是 ASCII 文本。网络上的大部分内容都是二进制数据,例如图像和视频。确保在选择和使用网络 I/O 函数时考虑二进制数据。

- 即使原始请求是HTTP/1.1,也将所有请求转发为HTTP/1.0。

祝你好运!