实验室检查点 0:网络预热

截止日期:4 月 11 日星期二晚上 10 点

欢迎来到 CS144:计算机网络简介。在本次热身中,您将在计算机上安装 Linux,学习如何通过 Internet 手动执行一些任务,用 C++ 编写一个通过 Internet 获取网页的小程序,并(在内存中)实现一个网络的关键抽象:写入器和读取器之间的可靠字节流。我们预计此热身需要您 2 到 6小时才能完成(未来的实验将花费您更多的时间)。关于实验室作业的三个要点:

- 在深入研究之前阅读整个文档是个好主意!
- 在这个由 8 部分组成的实验作业过程中,您将构建自己的Internet 重要部分的实现 路由器、网络接口和TCP 协议(将不可靠的数据报转换为可靠的字节)溪流)。

  大多数星期将建立在您之前完成的工作的基础上,即您将在本季度中逐渐构建自己的实施,并且您将在未来几周继续使用您的工作。这使得“跳过”检查点变得困难。

- 实验室文件不是“规范” 这意味着它们不适合以单向方式使用。它们的编写方式更接近软件工程师从老板或客户那里得到的详细程度。如果您发现某些内容不明确并且您认为答案很重要,我们希望您能够从参加实验室会议并提出澄清问题中受益。我们将更新课程中的“实验室常见问题解答”文档网站回应迟来的需要澄清的问题。

0 合作政策

编程作业必须是您自己的作品:您必须编写您为编程作业提交的所有代码,除了我们作为作业的一部分提供给您的代码。请不要从 Stack Overflow、GitHub 或其他来源复制并粘贴代码。如果您自己的代码基于在 Web 或其他地方找到的示例,请在提交的源代码的注释中引用 URL。

与他人合作:您不得向其他人展示您的代码、查看其他人的代码或查看前几年的解决方案。您可以与其他学生讨论作业,但不要复制任何人的代码。如果您与其他学生讨论作业,请在您提交的源代码的评论中注明他们的名字。请参阅课程管理讲义了解更多详细信息,如果有任何不清楚的地方,请在 EdStem 上询问。 GitHub Copilot 或 ChatGPT 等服务应被视为相当于“前一年参加过 CS144 的学生”。

EdStem:请随意在 EdStem 上提问,但请不要发布任何源代码。

1 在计算机上安装 GNU/Linux

CS144 的作业需要 GNU/Linux 操作系统和支持 C++ 2020 标准的最新 C++ 编译器。请选择以下三个选项之一：

1.推荐:安装 CS144 VirtualBox 虚拟机映像（说明位于 https://stanford.edu/class/cs144/vm howto/vm-howto-image.html）。

2.使用我们班级的优惠券代码使用 Google Cloud 虚拟机（说明位于 https://stanford.edu/class/cs144/vm howto）。

3. 运行 Ubuntu 22.10 版本,然后安装所需的软件包：

```
sudo apt update && sudo apt install git cmake gdb build-essential clang \
    clang-tidy clang-format gcc-doc pkg-config glibc-doc tcpdump tshark
```

4.使用另一个 GNU/Linux 发行版,但请注意,您可能会遇到障碍,并且需要轻松地调试它们。您的代码将在带有g++ 12.2 的Ubuntu 22.10 LTS 上进行测试,并且必须在这些条件下正确编译和运行。

5.如果您有 2020-23 MacBook（配备 ARM64 M1 或 M2 芯片）,VirtualBox 将无法成功运行。相反,请从https://stanford.edu/class/cs144/vm howto/安装 UTM 虚拟机软件和我们的ARM64 虚拟机映像。

2 手动联网

让我们开始使用网络。您将手动执行两项任务:检索网页（就像 Web 浏览器一样)和发送电子邮件消息（就像电子邮件客户端一样）。这两项任务都依赖于称为可靠双向字节流的网络抽象:您将在终端中键入字节序列,相同的字节序列最终将以相同的顺序传递到在另一个终端上运行的程序计算机（服务器）。服务器用自己的字节序列进行响应,并将其传递回您的终端。

2.1 获取网页

1. 在 Web 浏览器中,访问http://cs144.keithw.org/hello并观察结果。

2. 现在,您将手动执行与浏览器相同的操作。

   (a)在您的虚拟机上,运行telnet cs144.keithw.org http 。这告诉telnet程序在您的计算机和另一台计算机（名为cs144.keithw.org)之间打开可靠的字节流,并且在该计算机上运行特定的服务

该计算机： “http”服务,用于超文本传输协议,由
万维网.1

如果您的计算机已正确设置并且已连接到 Internet,您将看到：

```
user@computer:~$ telnet cs144.keithw.org http
正在尝试 104.196.238.229...
连接到 cs144.keithw.org。
转义字符是“^]”。
```

如果需要退出,请按住ctrl键并按] ，然后输入关闭 。

(b)输入GET /hello HTTP/1.1 （以第三个 。这告诉服务器 URL 的路径部分。
斜杠开头的部分。）

(c)输入主机：cs144.keithw.org URL。 （ http:// 。这告诉服务器主机部分
和第三个斜杠之间的部分。）

(d)连接类型：close 。这告诉服务器你已经完成了
发出请求,它应该在完成回复后立即关闭连接。

(e)再按一次 Enter 键： 。这会发送一个空行并告诉
服务器通知您已完成 HTTP 请求。

(f)如果一切顺利,您将看到与浏览器之前看到的相同的响应
通过告诉浏览器如何解释响应的 HTTP 标头。

3.作业：既然您已经知道如何手动获取网页,请向我们展示一下
能!使用上述技术获取 URL http://cs144.keithw.org/lab0/sunetid，
将 sunetid 替换为您自己的主要 SUNet ID。您将收到一个密码
X -Your-Code-Is:标头。保存您的 SUNet ID 和代码以包含在您的
写上去。

2.2 给自己发送一封电子邮件

现在您已经知道如何获取网页了,现在是时候发送电子邮件了,再次使用
到另一台计算机上运行的服务的可靠字节流。

1. SSH 至sunetid@cardinal.stanford.edu （以确保您位于斯坦福大学的网络上），
然后运行telnet 148.163.153.234 smtp 。[2] “smtp”服务是指简单的
邮件传输协议,用于发送电子邮件消息。如果一切顺利,您将看到：

```
用户@计算机:~$ telnet 148.163.153.234 smtp
```

1计算机名称具有等效的数字（104.196.238.229， Internet 协议 v4 地址），
服务的名称也是如此（80,TCP 端口号）。我们稍后会详细讨论这些。
2这些说明也可能在斯坦福大学网络外部起作用,但我们不能保证这一点。

正在尝试 148.163.153.234...
连接到 148.163.153.234。
转义字符是“^]”。 220
mx0b-00000d03.pphosted.com ESMTP mfa-m0214089

2. 第一步:向电子邮件服务器识别您的计算机。类型
HELO mycomputer.stanford.edu ⏎ 。等待看到类似“250 ... 你好”的内容
cardinal3.stanford.edu [171.67.24.75],很高兴见到你”。

3.下一步:谁发送电子邮件?输入邮件发件人:sunetid @stanford.edu ⏎ 。
将 sunetid 替换为您的 SUNet ID。3如果一切顺利,您将看到“250 2.1.0 Sender ok”。

4.下一步:收件人是谁?首先,尝试向自己发送电子邮件。类型.将 sunetid 替换为您自己的 SUNet ID。
RCPT 至:sunetid @stanford.edu ⏎
如果一切顺利,您将看到“250 2.1.5 收件人正常”。

5.是时候上传电子邮件了。数据类型 ⏎ 告诉服务器您已准备好开始。
如果一切顺利,您将看到“354 End data with <CR><LF>.<CR><LF>”。

6.现在您正在输入一封给自己的电子邮件。首先,首先输入您将在电子邮件客户端中看到的标题。在标题末尾保留一个空行。

354 以 <CR><LF>.<CR><LF> 结束数据来自:
sunetid@stanford.edu 至: ⏎
sunetid@stanford.edu ⏎
主题:CS144 实验室 0 的您好! ⏎
⏎

7.输入电子邮件的正文 - 任何您喜欢的内容。完成后,以单行上的一个点结束: 。
⏎ 。期望看到类似以下内容: “250 2.0.0 33h24dpdsr-1 消息已接受交付”。

8.输入退出垃圾邮 ⏎ 结束与电子邮件服务器的对话。检查您的收件箱并
件文件夹以确保您收到电子邮件。

9.作业:既然您已经知道如何手动向自己发送电子邮件,请尝试向朋友或实验室合作伙伴发送一封电子邮件,并确保他们收到。最后,告诉我们您可以向我们发送一份。使用上述技术将您自己的电子邮件发送至 cs144grader@gmail.com。

3是的,可以提供虚假的“发件人”地址。电子邮件有点像邮政服务的真实邮件,因为回信地址的准确性(大部分)取决于荣誉系统。您可以在明信片上写下任何您喜欢的内容作为寄信人地址,电子邮件也是如此。请不要滥用这一点 说真的。有了工程知识,就有了责任!垃圾邮件发送者和犯罪分子通常使用虚假的“发件人”地址发送电子邮件,这样他们就可以冒充其他人。

2.3 聆听和连接

您已经了解了 telnet 的功能:一个客户端程序,可以与其他计算机上运行的程序建立传出连接。现在是时候尝试成为一个简单的服务器了:等待客户端连接到它的程序。

1.在一个终端窗口中,在 VM 上运行`netcat -v -l -p 9090`。你应该看到:

```
user@computer:~$ netcat -v -l -p 9090 监听
[0.0.0.0]（系列 0,端口 9090）
```

2.让netcat保持运行。在另一个终端窗口中,运行`telnet localhost 9090`（也在您的虚拟机上）。

3.如果一切顺利， netcat将打印类似“Connection from 本地主机 53500 已收到！”。

4.现在尝试在任一终端窗口 - netcat （服务器)或telnet （客户端)中键入内容。请注意,您在一个窗口中键入的任何内容都会出现在另一个窗口中,反之亦然。您必须点击才能传输字节。

5.在netcat窗口中,输入ctrl -C退出程序。请注意， telnet 程序也立即退出。

3使用操作系统流编写网络程序插座

在本热身实验的下一部分中,您将编写一个简短的程序来通过 Internet 获取网页。您将利用 Linux 内核和大多数其他操作系统提供的一项功能:能够在两个程序之间创建可靠的双向字节流,一个程序在您的计算机上运行,另一个程序在 Internet 上的另一台计算机上运行（例如,Web 服务器（例如 Apache 或 nginx)或 netcat 程序）。

此功能称为流套接字。对于您的程序和 Web 服务器来说,套接字看起来就像一个普通的文件描述符（类似于磁盘上的文件,或者stdin或stdout I/O 流）。当两个流套接字连接时,写入一个套接字的任何字节最终都会以相同的顺序从另一台计算机上的另一个套接字输出。

然而实际上,互联网并不提供可靠的字节流服务。相反,互联网真正做的唯一事情就是尽其“最大努力”将称为互联网数据报的短数据传送到目的地。每个数据报都包含一些元数据（标头）,指定诸如源地址和目标地址之类的内容（它来自哪台计算机以及它前往哪台计算机)以及一些要传递到目的地的有效负载数据（最多约 1,500 字节)电脑。

尽管网络尝试传送每个数据报,但实际上数据报可能会 (1) 丢失， (2) 无序传送,(3) 传送的内容已更改,甚至 (4) 重复传送并多次传送。通常,连接两端的操作系统的工作是将“尽力而为的数据报”（互联网提供的抽象)转换为“可靠的字节流”（应用程序通常需要的抽象）。

两台计算机必须合作,以确保流中的每个字节最终都在其正确的位置传递到另一侧的流套接字。他们还必须告诉对方他们准备从对方计算机接受多少数据,并确保发送的数据量不会超过对方愿意接受的数据量。所有这一切都是通过 1981 年制定的、被称为传输控制协议 (TCP) 的商定方案来完成的。

在本实验中,您将只需使用操作系统预先存在的对传输控制协议的支持。您将编写一个名为“webget”的程序,该程序创建 TCP 流套接字、连接到 Web 服务器并获取页面,就像您在本实验室前面所做的那样。在未来的实验中,您将实现此抽象的另一面,通过自己实现传输控制协议,从不太可靠的数据报中创建可靠的字节流。

3.1 让我们开始吧 获取并构建起始代码

1.实验作业将使用名为“Minnow”的入门代码库。在您的虚拟机上,运行git clone https://github.com/cs144/minnow以获取实验室的源代码。

2.可选:随意将您的存储库备份到私有 GitHub/GitLab/Bitbucket 存储库（例如,使用https://stackoverflow.com/questions/10065526/github-how-to-make-a-fork-中的说明） of-公共-存储库-私有),但请绝对确保您的工作保密。

3.进入Lab 0目录:cd minnow

4. 创建编译实验软件的目录： cmake -S 。 -B构建

5.编译源码:cmake --build build

6.在构建目录之外,打开并开始编辑writeups/check0.md文件。这是您的实验室检查点写作的模板,并将包含在您的提交中。

3.2 现代 C++:大部分是安全的,但仍然快速且低级

实验室作业将以现代 C++ 风格完成,该风格使用最新 (2011) 的功能来尽可能安全地进行编程。这可能与过去要求您编写 C++ 的方式不同。有关此样式的参考,请参阅 C++ 核心指南(http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines)。

基本思想是确保每个对象都被设计为具有尽可能小的公共接口,具有大量的内部安全检查并且很难被不当使用,并且知道如何在自身之后进行清理。我们希望避免“成对”操作（例如 malloc/free 或 new/delete）,在这种情况下,配对的后半部分可能不会发生（例如,如果函数提前返回或引发异常）。相反,操作发生在对象的构造函数中,而相反的操作发生在析构函数中。这种风格称为“资源获取即初始化”,或 RAII。

我们特别希望您:

- 使用https://en.cppreference.com上的语言文档作为资源。 （我们建议您避免使用 cplusplus.com,因为它很可能已经过时。）
- 切勿使用malloc() 或free()。
- 切勿使用new 或delete。
- 基本上永远不要使用原始指针(*),仅在必要时才使用“智能”指针（唯一指针或共享指针）。 （您不需要在 CS144 中使用这些。）
- 避免模板、线程、锁和虚拟函数。 （您不需要使用这些在 CS144 中。）
- 避免C 风格字符串(char *str)或字符串函数（strlen()、 strcpy()）。这些非常容易出错。使用 std::string 代替。
- 切勿使用C 风格的强制转换（例如，(FILE *)x）。如果必须的话,请使用 C++静态转换（在 CS144 中通常不需要它）。
- 更喜欢通过const引用传递函数参数（例如： const 地址和地址）。
- 使每个变量为常量,除非需要改变。
- 使每个方法成为const,除非它需要改变对象。
- 避免全局变量,并为每个变量提供尽可能小的作用域。
- 在提交作业之前,运行cmake --build build --target tidy以获取有关如何改进与 C++ 编程实践相关的代码的建议,并运行cmake --build build --target format 以一致地格式化代码。

关于使用 Git:实验室作为 Git（版本控制)存储库进行分发,这是一种记录更改、检查版本以帮助调试以及跟踪源代码来源的方式。请在工作时经常进行小型提交,并使用提交消息来识别更改的内容和原因。柏拉图式的理想是每次提交都应该编译,并且应该稳步走向越来越多的测试通过。进行小的“语义”提交有助于调试（如果每次提交都编译并且消息描述了该提交所做的一件明确的事情,则调试会容易得多）,并通过记录您随着时间的推移的稳定进展来保护您免受作弊指控 - 这是一个有用的方法对包括软件开发在内的任何职业都有帮助的技能。评分者将阅读您的提交消息,以了解您如何为实验室开发解决方案。如果您还没有学会如何使用 Git,请在 CS144 办公时间寻求帮助

或查阅教程（例如， https://guides.github.com/introduction/git-handbook）。最后,欢迎您将代码存储在GitHub、GitLab、Bitbucket等上的私有存储库中,但请确保您的代码不可公开访问。

3.3 阅读Minnow支持代码

为了支持这种编程风格,Minnow 的类将操作系统函数（可以从 C 调用)封装在“现代”C++ 中。我们为您提供了 C++ 包装器,希望您熟悉 CS 110/111 中的概念,特别是套接字和文件描述符。

请仔细阅读公共接口（ util/socket.hh和util/filedescriptor.hh文件中“public:”后面的部分。 （请注意，Socket是 FileDescriptor 的一种类型,TCPSocket 是一种类型)插座。）

3.4 编写webget

现在是实现webget 的时候了,它是一个使用操作系统的 TCP 支持和流套接字抽象通过 Internet 获取网页的程序,就像您在本实验室之前手动执行的操作一样。

1. 在构建目录中,在文本编辑器或 IDE 中打开文件 ../apps/webget.cc。
2. 在获取 URL 函数中,找到以“// Your code here”开头的注释。
3. 使用您之前使用过的HTTP (Web) 请求格式,实现此文件中所述的简单 Web 客户端。使用 TCPSocket 和 Address 类。

4、提示：

- 请注意,在HTTP 中,每行必须以“\r\n”结束（仅使用“\n”或endl是不够的）。
- 不要忘记在客户的请求中包含“连接:关闭”行。这告诉服务器它不应该等待您的客户端在此之后发送更多请求。相反,服务器将发送一个回复,然后立即结束其传出字节流（从服务器套接字到您的套接字的字节流）。您会发现传入的字节流已结束,因为当您读取来自服务器的整个字节流时,套接字将到达“EOF”（文件结尾） 。这就是您的客户端知道服务器已完成回复的方式。
- 确保读取并打印服务器的所有输出,直到套接字到达“EOF”（文件结尾） 单次调用 read 是不够的。
- 我们预计您需要编写大约十行代码。

5.通过运行make来编译程序。如果您看到错误消息,您需要在继续之前修复它。

6.通过运行./apps/webget cs144.keithw.org /hello测试您的程序。这与您访问http://cs144.keithw.org/hello时看到的相比如何在网络浏览器中?与 2.1 节的结果相比如何?请随意尝试 - 使用您喜欢的任何 http URL 进行测试!

7.当看起来工作正常时,运行cmake --build build --target check webget来运行自动化测试。在实现 get URL 函数之前,您应该会看到以下内容:

```
$ cmake --build 构建 --target check_webget
测试项目 /home/cs144/minnow/build
      开始 1:使用错误检查器进行编译
1/2 测试 #1:使用错误检查器进行编译 ........ 通过 1.02秒
      开始2:t_webget
2/2 测试 #2:t_webget .....***调用失败的函数:get_URL(cs144.keithw.org, /nph 0.01秒
-hasher/xyzzy)
警告: get_URL() 尚未实现。
错误:webget 返回的输出与测试的预期不符
```

完成作业后,您将看到:

```
$ cmake --build 构建 --target check_webget
测试项目 /home/cs144/minnow/build
      开始 1:使用错误检查器进行编译
1/2 测试 #1:使用错误检查器进行编译 ........ 通过 1.09秒
      开始2:t_webget
2/2 测试 #2:t_webget .....通过 0.72秒

100% 测试通过,2 次测试中有 0 次失败
```

8.评分者将使用与make check webget运行不同的主机名和路径来运行您的webget程序,因此请确保它不仅适用于单元测试使用的主机名和路径。

4 内存中可靠的字节流

到目前为止,您已经了解了可靠字节流的抽象如何在Internet 上的通信中发挥作用,即使 Internet 本身仅提供“尽力而为”(不可靠)数据报的服务。

为了完成本周的实验,您将在一台计算机的内存中实现一个提供此抽象的对象。 (您可能在 CS 110/111 中做了类似的事情。) 字节

写在“输入”侧,并且可以以相同的顺序从“输出”侧读取。字节流是有限的:写入者可以结束输入,然后就不能写入更多字节。当读取器读取到流的末尾时,它将到达“EOF”(文件末尾)并且无法读取更多字节。

您的字节流还将受到流量控制,以限制其在任何给定时间的内存消耗。该对象使用特定的“容量”进行初始化:在任何给定点它愿意在其自己的内存中存储的最大字节数。字节流将限制写入器在任何给定时刻可以写入的数据量,以确保流不会超出其存储容量。当读取器读取字节并将其从流中耗尽时,写入器可以写入更多内容。您的字节流用于单个线程 - 您不必担心并发写入器/读取器、锁定或竞争条件。

需要明确的是:字节流是有限的,但它几乎可以任意长地结束输入并完成流。[4] 在作家面前
您的实现必须能够处理比容量长得多的流。容量限制在给定点内存中保存的字节数(已写入但尚未读取),但不限制流的长度。只要写入器一次持续写入一个字节,并且读取器在写入下一个字节之前读取每个字节,容量只有一个字节的对象仍然可以承载数 TB 和 TB 长的流。

对于作者来说,界面如下所示:

```
无效推送(std::字符串数据); // 将数据推送到流中,但仅限于可用容量允许的数量。

无效关闭();无效 // 指示流已到达结尾。不会再写更多的了。
set_error(); // 发出流发生错误的信号。

bool is_close() const; uint64_t // 流是否已关闭?
available_capacity() const; // 现在可以将多少字节推送到流中? uint64_t bytes_pushed() const;
// 累计推送到流的总字节数
```

这是读者的界面:

```
std::string_view peek() const; // 查看缓冲区中的下一个字节void pop( uint64_t len );
// 从缓冲区中删除 `len` 字节

bool is_finished() const; // 流是否已完成(关闭并完全弹出)? bool has_error() const; // 流有错误吗?

uint64_t bytes_buffered() const; // 当前缓冲的字节数(压入且未弹出) uint64_t bytes_popped() const; // 从流中累积弹出的字节总数
```

请打开src/bytestream.hh和src/bytestream.cc文件,并实现一个提供此接口的对象。在开发字节流实现时,您可以使用 cmake --build build --target check0 运行自动化测试。

如果所有测试都通过,则check0测试将运行您的实现的速度基准。
任何高于0.1 Gbit/s (换句话说,每秒 1 亿位)的速度都是可以接受的

4至少最多264字节,在此类中我们将其视为本质上任意长

出于本课程的目的。 （实施的执行速度可能超过 10 Gbit/s,但这取决于计算机的速度,并不是必需的。）

对于任何最新问题,请查看课程网站上的实验室常见问题解答或者在实验室（或 EdStem)上询问你的同学或教学人员。

下一步是什么?在接下来的四个星期内,您将实现一个系统来提供相同的接口,不再在内存中,而是通过不可靠的网络。这就是传输控制协议,它的实现可以说是世界上最流行的计算机程序。

5 提交

1.在您的提交中,请仅对webget.cc和src顶层的源代码（bytestream.hh和bytestream.cc)进行更改。请不要修改util 中的任何测试或帮助程序。

2. 在提交任何作业之前,请按顺序运行这些作业：

(a)确保您已将所有更改提交到 Git 存储库。您可以运行git status以确保没有未完成的更改。请记住:在编码时进行少量提交。

(b) cmake --build build --target format（规范编码风格）

(c) cmake --build build --target check0 （以确保自动化测试经过）

(d)可选： cmake --build build --target tidy （建议改进遵循良好的 C++ 编程实践）

3.完成编辑writeups/check0.md,填写本次作业的小时数接受了你和任何其他评论。

4. “如何上交”的机制将在截止日期前公布。

5.如果实验过程中出现任何问题,请尽快告知课程工作人员,或通过发帖方式告知关于 EdStem 的问题。祝你好运,欢迎来到 CS144!