

实验室检查点 1: 将子字符串拼接成字节流

截止日期: 4 月 17 日星期一上午 11 点

协作策略: 与检查点 0 相同。

0 概述

对于检查点 0, 您使用 Internet 流套接字从网站获取信息并发送电子邮件, 并使用 Linux 的传输控制协议 (TCP) 内置实现。即使底层网络仅提供“尽力而为”的数据报, 该 TCP 实现仍设法生成一对可靠的有序字节流 (一个从您到服务器, 另一个在相反方向)。我们的意思是: 可能丢失、重新排序、更改或复制的短数据包。您还自己在一台计算机的内存中实现了字节流抽象。在接下来的几周内, 您将自己实现 TCP, 以在由不可靠的数据报网络分隔的一对计算机之间提供字节流抽象。

我为什么要这么做? 在不同的不太可靠的服务之上提供服务或抽象可以解释网络中的许多有趣问题。在过去的 40 年里, 研究人员和从业者已经弄清楚了如何通过互联网传达各种信息 消息和电子邮件、超链接文档、搜索引擎、声音和视频、虚拟世界、协作文件共享、数字货币。

TCP 本身的作用, 即使用不可靠的数据报提供一对可靠的字节流, 就是典型的例子之一。一种合理的观点是, TCP 实现被认为是地球上使用最广泛的重要计算机程序。

实验作业将要求您以模块化方式构建 TCP 实现。

还记得您刚刚在 Checkpoint 0 中实现的 `ByteStream` 吗? 在接下来的实验中, 您最终将通过网络传输其中的两个字节流: 一个“出站”字节流, 用于本地应用程序写入套接字的数据, 并且您的 TCP 将发送给对等方; 另一个是“进站”字节流。

来自对等方的数据将由本地应用程序读取的字节流。

1 入门

您的 TCP 实现将使用您在 Checkpoint 0 中使用的相同 Minnow 库, 以及其他类和测试。开始:

1. 确保您已将所有解决方案提交到 Checkpoint 0。请不要修改 `src` 目录或 `webget.cc` 之外的任何文件。否则, 您可能会在合并 Checkpoint 1 起始代码时遇到问题。

2. 在实验室作业的存储库中, 运行 `git fetch` 以检索最新版本的实验室作业。

3.通过运行`git merge origin/check1-startercode`下载检查点 1 的起始代码。

4. 确保您的构建系统已正确设置: `cmake -S . -B构建`

5.编译源码:`cmake --build build`

6.打开并开始编辑`wrieteups/check1.md`文件。这是您实验室的模板
撰写并将包含在您提交的内容中。

2 将子串按顺序排列

在本实验和下一个实验中,您将实现一个 TCP 接收器:该模块接收数据报并将其转换为可靠的字节流,以便应用程序从套接字读取字节流,就像您的 `webget` 程序从 Web 服务器读取字节流一样检查点 0。

TCP 发送方将其字节流分成短段(每个子串不超过1,460 字节),以便它们各自适合一个数据报。但网络可能会重新排序这些数据报,或者丢弃它们,或者多次传送它们。接收方必须将这些段重新组装成它们开始时的连续字节流。

在本实验中,您将编写负责此重组的数据结构:重组器。它将接收由字节字符串以及该字符串的第一个字节在较大流中的索引组成的子字符串。流的每个字节都有自己唯一的索引,从零开始向上计数。

界面如下所示:

```
// 插入一个要重新组装到 ByteStream 中的新子字符串。 void insert( uint64_t first_index,
std::string data, bool is_last_substring, Writer& output );
```

```
// 重组器本身存储了多少字节? uint64_t bytes_pending() const;
```

我为什么要这么做? TCP 针对重新排序和重复的稳健性来自于其将字节流的任意摘录拼接回原始流的能力。在离散的可测试模块中实现这一点将使处理传入的段变得更容易。

重组器的完整(公共)接口由`reassembler.hh`标头中的`Reassembler`类描述。你的任务是实现这个类。您可以向`Reassembler`类添加所需的任何私有成员和成员函数,但不能更改其公共接口。

2.1 重组器内部应该存储什么？

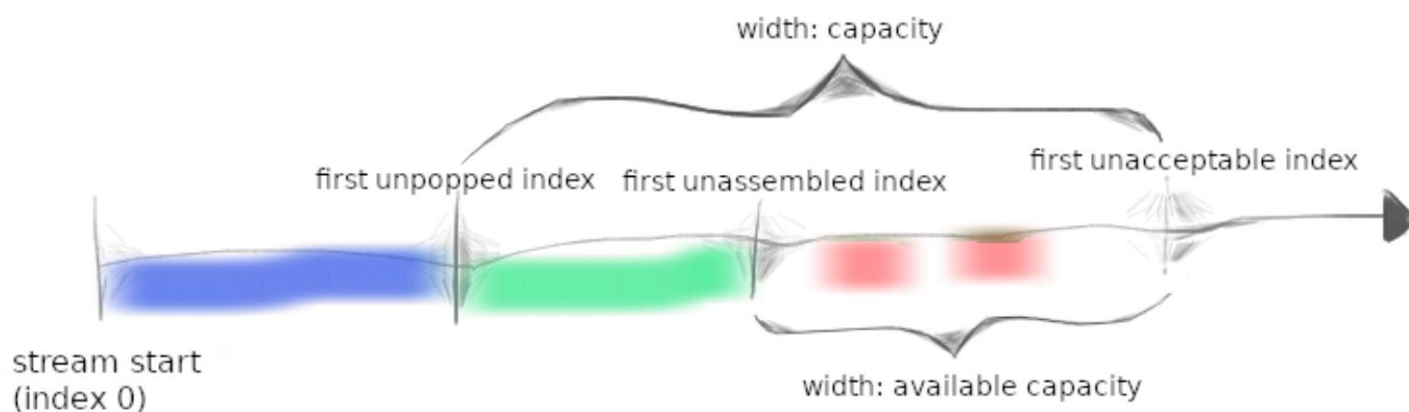
insert方法通知重组器有关 ByteStream 的新摘录,以及它在整个流中的位置（子字符串开头的索引）。


原则上,重组器必须处理三类知识:

- 1.流中下一个字节的字节。一旦知道它们,重组器就应该将它们推送给写入器。
- 2.适合流的可用容量但尚未写入的字节,因为较早的字节仍然未知。这些应该存储在重组器内部。
- 3.超出流可用容量的字节。这些应该被丢弃。重组器不会存储任何无法立即推送到字节流的字节,或者一旦知道较早的字节即可推送到字节流的字节。

此行为的目标是限制重组器和字节流使用的内存量,无论传入的子字符串如何到达。我们在下图中对此进行了说明。“容量”是两者的上限:

1. 重组后的 ByteStream 中缓冲的字节数（以绿色显示）,以及
2. “未汇编”子串可以使用的字节数（以红色显示）



 bytes (substrings) in the Reassembler's internal storage

 bytes buffered in the ByteStream

 bytes that have been popped already

当您实现重新组装器并完成测试时,您可能会发现这张图很有用 “正确”的行为并不总是很自然。

2.2 常见问题解答

- 整个流中第一个字节的索引是多少?零。
- 我的实施应该有多高效?数据结构的选择在这里也很重要。请不要将此视为构建空间或时间效率极低的数据结构的挑战 - 重组器将成为 TCP实现的基础。您有很多选择可供选择。

我们为您提供了一个基准;任何大于 0.1 Gbit/s (每秒 100 兆比特)的数据都是可接受的。顶级重组器将达到 10 Gbit/s。

- 应如何处理不一致的子串?您可能会认为它们不存在。也就是说,您可以假设有一个唯一的底层字节流,并且所有子字符串都是它的 (准确的)切片。

- 我可以使用什么?您可以使用您认为有帮助的标准库的任何部分。在特别是,我们希望您至少使用一种数据结构。

- 何时应将字节写入流?尽快地。字节不应该出现在流中的唯一情况是当它之前有一个字节尚未被 “推送”时。

- 提供给insert() 函数的子字符串可以重叠吗?是的。
- 我需要将私有成员添加到重组器中吗?是的。子字符串可能以任何顺序到达,因此您的数据结构必须 “记住”子字符串,直到它们准备好放入流中,即直到它们之前的所有索引都被写入为止。
- 我们的重组数据结构可以存储重叠的子字符串吗?不。可以实现存储重叠子字符串的 “接口正确”重组器。但允许重新编译器执行此操作破坏了 “容量”作为内存限制的概念。如果调用者提供有关同一索引的冗余知识,则重组器应仅存储该信息的一份副本。

- 您预计有多少行代码?当我们运行./scripts/lines-of-code 在起始代码上,它打印:

```
字节流:          89行代码
重组器:22行代码
```

当我们在我们的解决方案上运行它时,它会打印:

```
ByteStream:134行代码
重组器:75行代码
```

因此,重组器的合理实现可能是大约 50-60 行重组器代码 (在起始代码之上)。

- 更多常见问题解答:更多请参见<https://cs144.github.io/lab faq.html>。

3 开发调试建议

1. 您可以使用 `cmake --build build --target check1` 测试您的代码（编译后）。
2. 请重新阅读 Lab 0 文档中有关“使用 Git”的部分,并记住将代码保存在主分支上分发的 Git 存储库中。进行小型提交,使用良好的提交消息来确定更改的内容和原因。
3. 请努力使您的代码对 CA 可读,CA 将对其风格和健全性进行评分。对变量使用合理、清晰的命名约定。使用注释来解释复杂或微妙的代码片段。使用“防御性编程”显式检查函数或不变量的先决条件,如果出现任何错误,则抛出异常。在设计中使用模块化 识别常见的抽象和行为,并在可能的情况下将它们分解出来。重复的代码块和庞大的函数会让你很难理解你的代码。
4. 还请遵循 Checkpoint 0 文档中描述的“Modern C++”风格。
cppreference 网站(<https://en.cppreference.com>)是一个很好的资源,尽管您不需要 C++ 的任何复杂功能来完成这些实验。（有时您可能需要使用 `move()` 函数来传递无法复制的对象。）
5. 如果您的构建陷入困境并且不确定如何修复它们,您可以删除您的构建目录（`rm -rf build` 请小心不要输入错误,因为这会删除您告诉它的任何内容）,然后运行 `cmake -S`。-B 再次构建。

4 提交

1. 在您的提交中,请仅对 `src` 目录中的 `.hh` 和 `.cc` 文件进行更改。在这些文件中,请根据需要随意添加私有成员,但请不要更改任何类的公共接口。
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`（标准化编码风格）
 - (c) `cmake --build build --target check0`（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. 在 `writeups/check1.md` 中编写报告。该文件应该是大约 20 到 50 行的文档,每行不超过 80 个字符,以便于阅读。报告应包含以下部分:

(a)程序结构 and 设计。描述代码中体现的高级结构和设计选择。您无需详细讨论从起始代码继承的内容。以此为契机,突出显示重要的设计方面,并提供这些领域的更多详细信息,以便您的评分助教能够理解。强烈建议您使用副标题和大纲,使本文尽可能具有可读性。请不要简单地将您的程序翻译成一段英文。(b)实施挑战。描述您发现最麻烦的代码部分并解释原因。反思一下你是如何克服这些挑战的,以及是什么帮助你最终理解了给你带来麻烦的概念。您如何尝试确保您的代码维持您的假设、不变量和先决条件?您认为这很容易还是很困难?你是如何调试和测试你的代码的?

(c)剩余的 error。尽可能指出并解释代码中残留的任何 error (或未处理的边缘情况)。

4. 在您的写作中,还请填写作业花费的小时数以及任何其他评论。

5. “如何上交”的机制将在截止日期前公布。

6. 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埃德的问题。祝你好运!