实验室检查点 1：将子字符串拼接成字节流

到期的：10 月 8 日星期五下午 5 点
最迟截止期限：10 月 10 日晚上 11:59（接收反馈的最后一天；风格评分上限为 2/3）

0 合作政策

编程作业必须是你自己的作品：您必须编写您提交的编程作业的所有代码，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 或其他来源复制粘贴代码。如果您根据在 Web 或其他地方找到的示例编写自己的代码，请在您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，但不得抄袭任何人的代码。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，请向 Ed 询问。

埃德：请随意向 Ed 提问，但请不要发布任何源代码。

1 概述

建议：实施前请阅读整个实验文档。

在实验 0 中，你使用了*互联网流套接字*使用 Linux 内置的传输控制协议 (TCP) 实现从网站获取信息并发送电子邮件消息。此 TCP 实现设法生成一对*可靠的有序字节流*（一个是从您到服务器，另一个是从服务器到服务器），即使底层网络只提供“尽力而为”的数据报。我们指的是：可能丢失、重新排序、更改或重复的短数据包。您还在一台计算机的内存中自己实现了字节流抽象。在接下来的四周内，您将实现 TCP，以在由不可靠的数据报网络分隔的两台计算机之间提供字节流抽象。

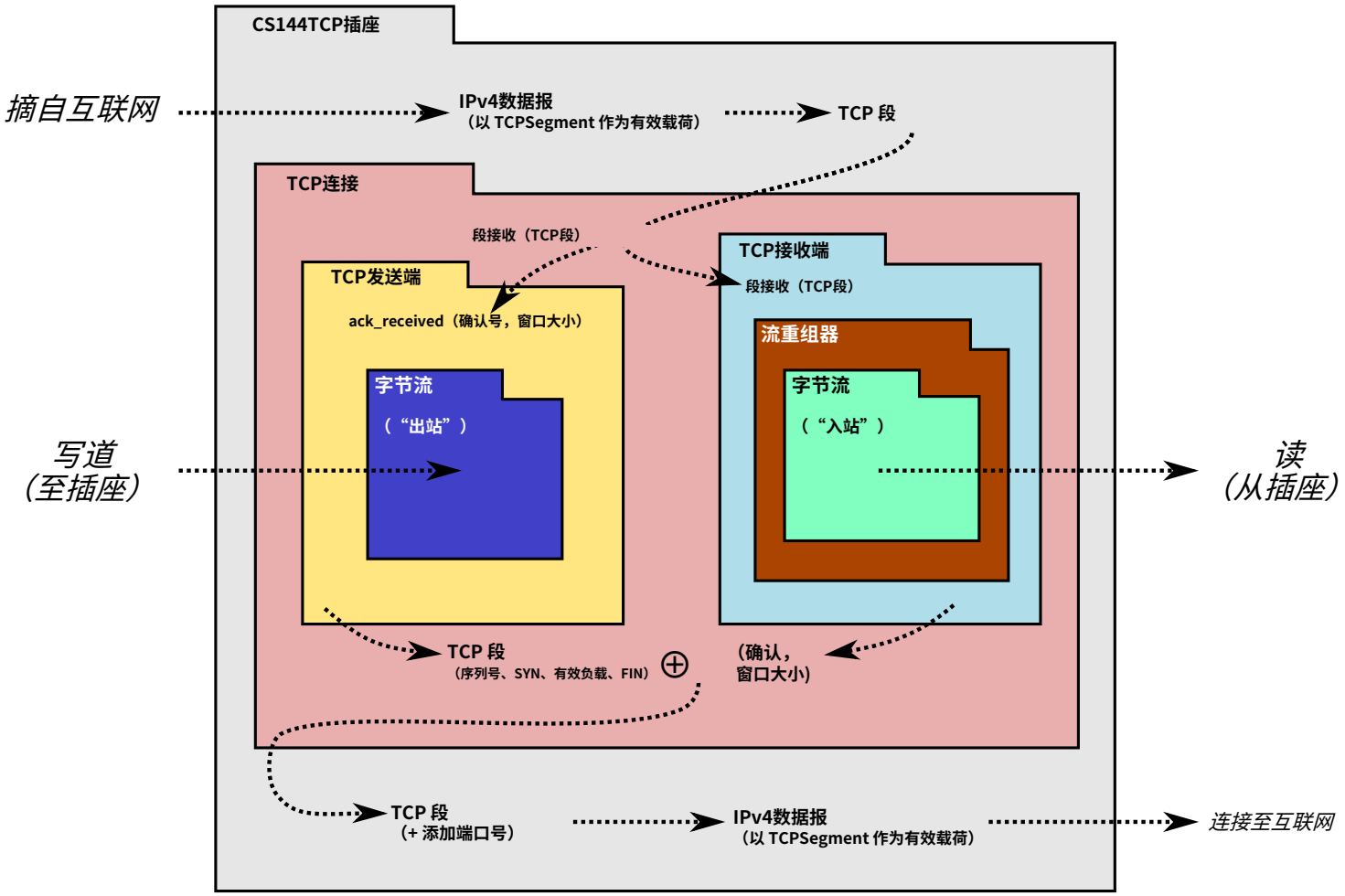


图 1：TCP 实现中的模块和数据流的排列。 字节流是实验 0。TCP 的工作是传递两个字节流s（每个方向一个）在不可靠的数据报网络上，这样写入连接一端套接字的字节就会出现在对端可以读取的字节中，反之亦然。实验 1 是 流重组器，在实验 2、3 和 4 中，你将实现TCP接收器，TCP发送器， 然后TCP连接把所有东西联系在一起。

> *“我为什么要这么做？”* 在另一种不太可靠的服务之上提供服务或抽象，这可以解释网络中许多有趣的问题。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递各种东西——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 自身的作用是使用不可靠的数据报提供一对可靠的字节流，这是其中的一个典型例子。一种合理的观点是，TCP 实现是地球上使用最广泛的非平凡计算机程序。

实验作业将要求你以模块化的方式构建 TCP 实现。记住字节流你刚刚在实验 0 中实现了什么？在接下来的四个实验中，你将最终通过网络传达其中两个：“出站”字节流，本地应用程序写入套接字的数据以及 TCP 将发送的数据*到*对等体，以及“入站” 字节流用于数据传入*从*本地应用程序将读取的对等体。图1 显示了各个部分如何组合在一起。

1. 在实验 1 中，你将实现一个*流重组器*— 一个将字节流的小片段（称为子字符串或段）返回到正确顺序的连续字节流。

2. 在实验 2 中，你将实现 TCP 中处理入 站字节流的部分： TCP接收器。这需要思考 TCP 如何表示每个字节在流中的位置——即所谓的“序列号”。TCP接收端负责告诉发送方（a）它能够成功组装多少入站字节流（这称为“确认”）以及（b）发送方现在允许发送多少字节（“流控制”）。

3. 在实验 3 中，你将实现 TCP 中处理出站字节流的部分： TCP发送方。当发送方怀疑自己发送的某个数据段在途中丢失，并且从未到达接收方时，发送方应该如何反应？何时应该再次尝试重新发送丢失的数据段？

4. 在实验 4 中，你将结合前两个实验中的工作来创建一个有效的 TCP 实现：TCP连接包含一个TCP发送端和TCP接收器。 您将使用它与世界各地的真实服务器进行对话。

2 入门

您的 TCP 实现将使用与实验 0 中相同的 Sponge 库，并附加一些类和测试。开始之前：

1. 确保你已经将所有解决方案提交给实验室 0。请不要修改自由海绵目录，或webget.cc。否则，您可能无法合并实验室 1 启动代码。

2. 在实验室作业存储库中，运行实验室作业的最新版本。 git 获取 检索

3. 运行以下命令下载实验 1 的起始代码 git 合并 origin/lab1-startercode 。

4. 在您 的建造目录，编译源代码： 制作 （例如，你可以运行 制作-j4 在编译时使用四个处理器）。

5. 外面建造目录，打开并开始编辑writeups/lab1.md fi这是您的实验室报告模板，将包含在您的提交内容中。

3 按顺序排列子字符串

在这个和下一个实验中，你将实现一个 TCP 接收器：该模块接收数据报并将其转换为可靠的字节流，以供应用程序从套接字读取——就像你的网络获取程序从实验室 0 中的 Web 服务器读取字节流。

TCP 发送方将其字节流分成短*段*（子字符串（每个不超过 1,460 字节）以便每个子字符串都适合一个数据报。但网络可能会重新排序这些数据报，或丢弃它们，或多次传送它们。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，你将编写负责此重组的数据结构： StreamReassembler。它将接收由一串字节组成的子字符串，以及较大流中该字符串第一个字节的索引。流中的每个字节有自己独特的索引，从零开始向上计数。流重组器 将拥有字节流对于输出：重组器一旦知道流的下一个字节，它就会将其写入字节流。所有者可以访问并读取 字节流只要它愿意。

界面如下所示：

```
// 构造一个 `StreamReassembler`，它将存储最多 `capacity` 个字节。 流重组器(常量 size_t容量）；

// 接收子字符串并将任何新的连续字节写入流中，// 同时保持在“容量”的内存限制内。超出容量的字节将被默默丢弃。

//
// `data`：子字符串
// `index` 表示 `data` 中第一个字节的索引（按顺序排列） // `eof`：此子字符串的最后一个字节将是整个流中的最后一个字节 空白推送子字符串（常量细绳&数据，常量uint64_t指数，常量布尔值结束符;


// 访问重新组装的 ByteStream（来自实验室 0 的代码） 字节流&流输出 （）；

// 子字符串中已存储但尚未重组的字节数 size_t未组装的字节 （）常量;
```

```
    // 内部状态是否为空（除输出流外）？ 布尔值空的 () 常量;
```

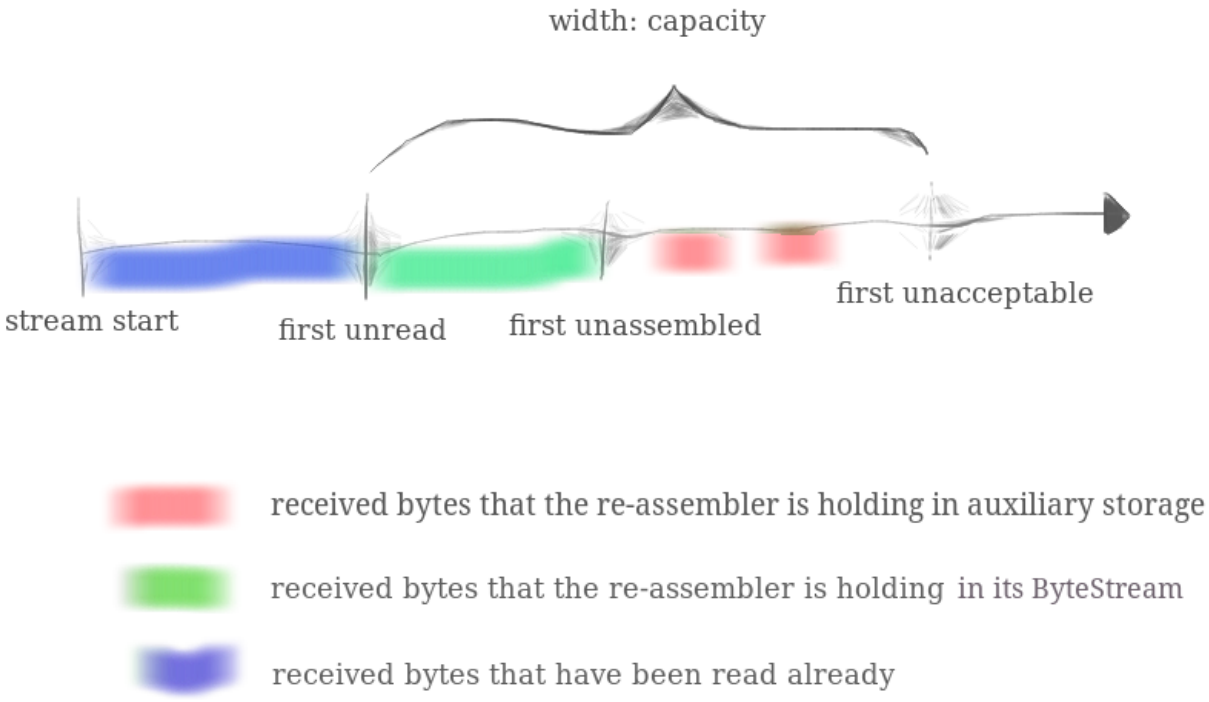
> *“我为什么要这么做？”* TCP 对重新排序和重复的稳健性来自于它能够将字节流的任意摘录缝合回原始流。在离散的可测试模块中实现此功能将使处理传入的段*很多*更轻松。

重组器的完整（公共）接口由流重组器类中的流重组器.hh标头。您的任务是实现此类。您可以将任何所需的私有成员和成员函数添加到流重组器类，但你不能改变它的公共接口。

3.1 什么是“容量”？

你的推送子字符串方法将忽略字符串中可能导致 流重组器超过其“容量”：内存使用量的限制，即允许存储的最大字节数。这可以防止重组器使用无限量的内存，无论 TCP 发送方决定做什么。我们在下图中说明了这一点。“容量”是*两个都*：

1. 重组后的字节数字节流（如下图绿色部分所示），以及
2. “未组装”子字符串可使用的最大字节数（以红色显示）



你可能会发现这张图片在你实现流重组器并通过测试——什么是“正确”的行为并不总是自然而然的。

3.2 常见问题

- *整个流中第一个字节的索引是什么？* 零。
- *我的实施效率应该有多高？* 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——此数据结构将成为 TCP 实现的基础。大致的期望是，每个新的 Lab 1 测试都可以在不到半秒的时间内完成。
- *如何处理不一致的子字符串？* 您可以假设它们不存在。也就是说，您可以假设存在一个唯一的底层字节流，并且所有子字符串都是它的（准确）切片。
- *我可以使用什么？* 您可以使用标准库中您认为有用的任何部分。特别是，我们希望您至少使用一个数据结构。
- *何时应将字节写入流？* 尽快。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”。

- *可以提供给推送子字符串（）功能重叠？* 是的。
- *我是否需要向 StreamReassembler 添加私有成员？* 是的。子字符串可以按任何顺序到达，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 - 也就是说，直到它们之前的所有索引都已写入。
- *我们的重新组装数据结构可以存储重叠的子字符串吗？* 不。可以实现一个存储重叠子字符串的“接口正确”重组器。但允许重组器这样做会破坏“容量”作为内存限制的概念。评分时，我们会将存储重叠子字符串视为风格违规。
- *更多常见问题*：更多详情请参阅https://cs144.github.io/lab常见问题.html。

4 开发调试建议

1. 您可以使用以下方法测试代码（编译后） `检查实验室1` 。
2. 请重新阅读 Lab 0 文档中关于“使用 Git”的部分，并记住将代码保存在它发布的 Git 存储库中掌握分支。进行小规模的提交，使用良好的提交消息来识别更改的内容及其原因。
3. 请努力使代码易于 CA 理解，CA 将根据代码风格对其进行评分。使用合理、清晰的变量命名约定。使用注释来解释复杂或微妙的代码片段。使用“防御性编程”——明确检查函数或不变量的先决条件，如果出现任何问题，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其分解出来。重复的代码块和庞大的函数会使您的代码难以理解。
4. 请同时遵守 Lab 0 文档中描述的“现代 C++”风格。cppreference 网站 (https://en.cppreference.com）是一个很好的资源，尽管你不需要任何复杂的 C++ 功能来完成这些实验。（有时你可能需要使用移动（）函数传递一个不可复制的对象。）
5. 如果遇到分段错误，那一定出了问题！我们希望您能够采用安全的编程实践来编写代码，从而让分段错误变得极为罕见（不会malloc()，不新的，没有指针，安全检查会在你不确定的地方抛出异常，等等）。也就是说，为了调试，你可以使用以下命令配置你的构建目录：
   `cmake .. -DCMAKE 构建类型=RelASan` 启用编译器的“清理器”检测内存错误和未定义行为，并在发生时为您提供良好的诊断。您还可以使用瓦尔格林德工具。您还可以使用
   `cmake .. -DCMAKE 构建类型=调试` 并使用 GNU 调试器（您可以使用 gdb 命令来配置 .但请

请记住，这些选项（尤其是清理器）会减慢 编译和执行的速度 - 您不会想意外地将它们留在原处！

6. 你可以使用以下命令重置构建系统 使清洁 和 cmake .. -DCMAKE 构建类型=发布 。或者如果你的构建真的卡住了，并且不确定如何修复它们，你可以删除你的建造目录（ rm-rf 构建 — 请注意不要输入错误，因为会擦除你告诉它的所有内容），创建一个新的建造目录，以及 制作... 再次。

5 提交

1. 在您提交的作品中，请仅对 进行更改。时和 。cc fi位于顶层的libsponge。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*民众*任何类的接口。

2. 在提交任何作业之前，请按顺序运行以下内容：

   （一）制作格式 （规范化编码风格）

   （二）制作 （确保代码可以编译）

   （三）检查实验室1 （确保自动化测试通过）

3. 撰写报告writeups/lab1.md。该文件应为大约 20 到 50 行的文档，每行不超过 80 个字符，以方便阅读。报告应包含以下部分：

   （一个）程序结构和设计。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中继承的内容。利用这个机会突出重要的设计方面，并更详细地介绍这些方面，以便您的评分助教理解。强烈建议您使用小标题和大纲使本文尽可能易于阅读。请不要简单地将您的程序翻译成一段英文。

   （二）实施挑战。描述您发现最麻烦的代码部分并解释原因。反思您如何克服这些挑战以及是什么帮助您最终理解了让您感到困难的概念。您如何尝试确保您的代码保持您的假设、不变量和先决条件，以及在哪些方面您觉得这很容易或很难？您如何调试和测试您的代码？

   （三）餘下的错误。尽可能指出并解释代码中存在的任何错误（或未处理的边缘情况）。

4. 在您 的报告中，请填写完成作业所花费的小时数以及其他任何评论。

5. 准备提交时，请按照以下说明操作https://cs144.github.io/提交。提交前请确保您已完成所有想要完成的工作。

6. 如果在周二晚上的实验课上遇到任何问题，请尽快告知课程工作人员，或者在 Ed 上发布问题。祝你好运！