CS144：计算机网络导论 2025年秋季

实验检查点 1: 将子串拼接为字节流

截止日期：10月5日（周日）晚上11:59

合作政策：与检查点0相同。

0 概述

在检查点0中，你使用互联网流套接字从网站获取信息并发送电子邮件，使用的是Linux内置的传输控制协议（TCP）实现。该TCP实现设法产生了一对可靠的有序字节流（一个从你到服务器，另一个方向相反），尽管底层网络只能提供"尽力而为"的数据报。我们的意思是：短数据包可能会丢失、重新排序、改变或重复。你还在一台计算机的内存中自己实现了字节流抽象。在接下来的几周里，你将亲自实现TCP，以在通过不可靠数据报网络分隔的两台计算机之间提供字节流抽象。

★我为什么要做这个？　在不太可靠的服务之上提供服务或抽象，涵盖了许多有趣的网络问题。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各样的东西——消息和电子邮件、超链接文档、搜索引擎、音频和视频、虚拟世界、协作文件共享、数字货币。TCP自身的角色——使用不可靠数据报提供一对可靠字节流——是其中的经典范例之一。一种合理的观点认为，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方式构建TCP实现。还记得你在检查点0中刚实现的ByteStream吗？在接下来的实验中，你最终将跨网络传输两个ByteStream：一个"出站"ByteStream，用于本地应用程序写入套接字并由你的TCP发送到对端的数据；以及一个"入站"ByteStream，用于来自对端并将被本地应用程序读取的数据。

本检查点包含一个"动手实践"部分和一个实现部分。你可能更愿意在实验课之前先开始实现部分，然后在实验课上完成动手实践部分。如果你是CGOE学生，请使用EdStem与另一位学生协调时间来完成动手实践部分。

动手实践部分涉及多个活动部件——因此可能会出现一些小问题。请在实验课上耐心配合，我们会尽力让每个人都能顺利运行。如果你看到来自 https://cs144.net 网站的错误消息，请在EdStem上以公开帖子的形式报告，我们会查看。

1 开始

你的TCP实现将使用与检查点0中相同的Minnow库，并增加了额外的类和测试。开始步骤：

确保已提交检查点0的所有解决方案。请不要修改src目录之外的任何文件，也不要修改webget.cc。否则你可能无法合并检查点1的起始代码。

在实验作业仓库内，运行 `git fetch` 以获取实验作业的最新版本。

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

确保构建系统已正确设置：`cmake -S . -B build`

编译源代码：`cmake --build build`

打开并开始编辑 `writeups/check1.md` 文件。这是你实验报告的模板，将包含在你的提交中。

2 动手实践部分：课程专用网络

我们为CS144课程创建了一个专用网络。这将允许你的虚拟机直接与课程中其他学生的虚拟机之间发送数据报。要让你的虚拟机加入此网络：

在你的虚拟机上，通过运行 `sudo apt install wireguard apparmor-utils` 安装"wireguard"和"apparmor-utils"软件包。

运行 `sudo aa-complain /etc/apparmor.d/wg` 以移除专用网络软件的一些安全限制。

访问 https://cs144.net/wg 并按照说明加入CS144专用网络。

加入网络后，按照该页面上的"ping"说明验证你可以连接（加入网络后说明会出现）。

每次重启虚拟机时，你都需要重新加入网络（如果你希望能够与本课程的其它学生之间发送数据报）。你不需要每次都注册新的公钥，但确实需要重新运行该网页上的命令。每次命令相同。

2.1 Ping一个朋友并查看数据报

在你自己的电脑上（例如你的Mac或Windows机器——不是你的虚拟机），按照 https://www.wireshark.org/ 上的说明安装"wireshark"程序。（如果你使用的是Debian或Ubuntu GNU/Linux，命令是 `sudo apt install wireshark`。）

向一位组员询问他们的IP地址（https://cs144.net/wg 网页上为他们显示的地址）。使用 `ping` 命令，向你的朋友发送一些"echo请求"数据报，并确保你收到了一些"echo回复"数据报。

提示：

你可以通过按 `ctrl-C` 来结束"ping"程序。

你可以通过添加参数 `-i 0.2` 使"ping"命令发送更快。这将使其每0.2秒发送一次"echo请求"（每秒5次）。

你可以通过在另一个终端运行以下命令使"ping"命令打印出到目前为止的统计摘要（而不结束它）：`killall -QUIT ping`。

在你的报告中开始撰写报告，包含以下信息：

(a) 你的虚拟机发送"echo请求"到收到组员的"echo回复"之间的平均往返延迟是多少？

(b) 交付率是多少（"echo请求"中有多少百分比收到了对应的"echo回复"）？丢失率是多少（即100%减去交付率）？发送至少1,000次ping以获得可靠的估计。（如果使用 `ping -i 0.2`，这大约需要三分钟。）

(c) 你是否看到了任何重复的数据报（ping会打印"DUP"）？

(d) 在ping运行期间，你和你的组员可以通过运行以下命令捕获一些原始互联网数据报：

```
sudo rm /tmp/capture.raw; sudo tcpdump -n -w /tmp/capture.raw -i wg0 --print --packet-buffered
```

此命令将捕获"wg0"接口（课程专用网络）上的数据报到文件（"/tmp/capture.raw"），同时还将它们打印到屏幕上。确保你看到一些"echo请求"和"echo回复"行被打印出来——这表明你的组员正在接收你的数据报并回复你。

(e) 使用wireshark程序检查你们各自虚拟机上的 `/tmp/capture.raw` 文件。你可能想要将 `capture.raw` 文件用 `scp` 拷贝到你自己的电脑上（例如Mac或Windows机器），然后使用wireshark打开此文件，以便使用其图形界面。你能否找到1月10日讨论过的互联网数据报的各字段（并与 https://www.rfc-editor.org/rfc/rfc791.html#page-11 上的图示匹配）？

(f) 同一个数据报在你的虚拟机上捕获时与在你朋友的虚拟机上捕获时是否有任何差异？差异是什么？确保你确实在分析同一个数据报，从两个不同的视角（发送方与接收方）捕获。

2.2 手动发送一个互联网数据报

在 `apps/ip_raw.cc` 文件中，编写一个程序，使用原始套接字向你的朋友发送一个互联网数据报，使用与9月24日课程相同的方法。可以修改该课程中的代码。

向你的组员发送一个IP协议号为"5"的互联网数据报（你需要使用"sudo"来运行 `./build/apps/ip_raw` 程序），并让你的朋友使用tcpdump确保他们收到了该数据报。他们可以运行 `sudo tcpdump -n -i wg0 'proto 5'` 来仅打印匹配协议"5"的数据报。确保他们收到了！

向你的组员发送一个用户数据报（IP协议号为"17"），使用 https://www.rfc-editor.org/rfc/rfc768 中的"用户数据报"头部格式。让你的组员在不使用"sudo"的情况下接收此数据报。他们可以使用课堂上演示的 `nc -u` 程序，或使用UDPSocket类的C++程序——随他们选择！

在你的提交中包含你的 `ip_raw.cc` 代码。

反向操作，接收来自组员的一个数据报。

3 实现：按顺序放置子串

作为实验作业的一部分，你正在实现一个TCP接收方：该模块接收数据报并将它们转换为可靠的字节流，供应用程序从套接字读取——就像你在检查点0中的webget程序从Web服务器读取字节流一样。

TCP发送方将其字节流分成短段（每个子串不超过约1,460字节），以便它们各自放入一个数据报中。但网络可能会重新排序这些数据报、丢弃它们或多次传递它们。接收方必须将这些段重新组装为它们最初组成的连续字节流。

在本实验中，你将编写负责此重新组装的数据结构：一个`Reassembler`。它将接收子串，子串由一串字节和该字符串的第一个字节在更大流中的索引组成。流的每个字节都有自己唯一的索引，从零开始向上计数。一旦Reassembler知道了流的下一个字节，它就会将其写入ByteStream的Writer端——与你在检查点0中实现的ByteStream相同。Reassembler的"客户"可以从同一ByteStream的Reader端读取。

接口如下所示：

```
// Insert a new substring to be reassembled into a ByteStream.
void insert( uint64_t first_index, std::string data, bool is_last_substring );

// How many bytes are stored in the Reassembler itself?
// This function is for testing only; don't add extra state to support it.
uint64_t count_bytes_pending() const;

// Access output stream reader
Reader& reader();
```

★我为什么要做这个？ TCP对重新排序和重复的鲁棒性来自于它将字节流的任意片段拼接回原始流的能力。在一个离散的可测试模块中实现这一点将使处理传入段变得更容易。

Reassembler的完整（公共）接口由 `reassembler.hh` 头文件中的Reassembler类描述。你的任务是实现此类。你可以向Reassembler类添加任何你想要的私有成员和成员函数，但不能更改其公共接口。

3.1 Reassembler内部应该存储什么？

`insert` 方法通知Reassembler有关ByteStream的新片段以及它在整个流中的位置（子串开头的索引）。

因此，原则上，Reassembler将需要处理三类信息：

流中的下一个字节。Reassembler应在知道这些字节后立即将它们推送到流中（`output.writer()`）。

适合流的可用容量但尚不能写入的字节，因为更早的字节仍然未知。这些应该在Reassembler内部存储。

超出流可用容量的字节。这些应该被丢弃。Reassembler不会存储任何无法立即推送到ByteStream或一旦更早字节变为已知时推送的字节。

此行为的目的是限制Reassembler和ByteStream使用的内存量，无论传入子串如何到达。我们在下图中说明了这一点。"容量"是以下两者的上限：

在重新组装的ByteStream中缓冲的字节数（以绿色显示），以及

可被"未组装"子串使用的字节数（以红色显示）

在你实现Reassembler并完成测试的过程中，你可能会发现这张图很有用——什么是"正确的"行为并不总是那么直观。

3.2 常见问题

整个流中第一个字节的索引是多少？ 零。

我的实现应该有多高效？ 数据结构的选择在这里同样重要。请不要把这当作构建极其浪费空间或时间的数据结构的挑战——Reassembler将是你TCP实现的基础。你有很多选择。

我们为你提供了一个基准测试；任何大于0.1
Gbit/s（100兆位每秒）的结果都是可接受的。顶级的Reassembler可以达到10 Gbit/s。

应该如何处理不一致的子串？ 你可以假设不存在不一致。也就是说，你可以假设存在一个唯一的底层字节流，所有子串都是它的（准确）切片。

我可以使用什么？ 你可以使用标准库中任何有帮助的部分。特别是，我们希望你至少使用一种数据结构。

字节应该在什么时候写入流？ 尽快。字节不应在流中的唯一情况是，在它之前存在尚未被"推送"的字节。

提供给`insert()`函数的子串可以重叠吗？ 可以。

我是否需要向`Reassembler`添加私有成员？ 是的。子串可能以任意顺序到达，因此你的数据结构必须"记住"子串，直到它们准备好被放入流中——也就是说，直到它们之前的所有索引都已被写入。

我们的重新组装数据结构可以存储重叠的子串吗？ 不可以。虽然可以实现一个存储重叠子串的"接口正确"的Reassembler。但允许Reassembler这样做会破坏"容量"作为内存限制的概念。如果调用者提供了关于同一索引的冗余信息，Reassembler应该只存储一份此信息的副本。

`Reassembler`会使用`ByteStream`的`Reader`端吗？
不会——那是给外部客户的。Reassembler只使用Writer端。

你们期望多少行代码？ 当我们在起始代码上运行 `./scripts/lines-of-code` 时，它打印：

```
ByteStream: 90 lines of code
Reassembler: 26 lines of code
```

当我们在我们的解决方案上运行时，它打印：

```
ByteStream: 109 lines of code
Reassembler: 87 lines of code
```

因此，Reassembler的合理实现大约需要50-60行代码（在起始代码的基础上）。

更多常见问题：请访问 https://cs144.github.io/lab_faq.html。

4 开发与调试建议

你可以在编译后用 `cmake --build build --target check1` 测试你的代码。

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分发的Git仓库的main分支上。进行小的提交，使用好的提交消息来标识更改了什么以及为什么。

请努力使你的代码对评分的CA可读，他们将根据风格和健全性进行评分。使用合理且清晰的变量命名约定。使用注释来解释复杂或微妙的代码段。使用"防御性编程"——显式检查函数的前置条件或不变量，如果有任何错误则抛出异常。在你的设计中使用模块化——识别共同的抽象和行为，并在可能时将它们提取出来。重复的代码块和巨大的函数会使你的代码难以理解。

请同时遵循检查点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，虽然你不需要使用C++的任何高级特性来完成这些实验。（你有时可能需要使用 `move()` 函数来传递无法复制的对象。）

如果你的构建卡住了且不确定如何修复，可以删除你的构建目录（`rm -rf build`——请注意不要打错字，因为这会删除你告诉它删除的任何内容），然后重新运行 `cmake -S . -B build`。

5 提交

在你的提交中，请仅更改 `src` 目录中的 `.hh` 和 `.cc` 文件。在这些文件中，请随意添加必要的私有成员，但请不要更改任何类的公共接口。

在提交任何作业之前，请按顺序运行以下命令：

(a) 确保已将所有更改提交到Git仓库。你可以运行 `git status` 来确保没有未提交的更改。记住：在编码时进行小的提交。

(b) `cmake --build build --target format`（规范化编码风格）

(c) `cmake --build build --target check1`（确保自动化测试通过）

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

在 `writeups/check1.md` 中撰写报告。此文件应为大约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） 结构与设计。 描述你代码中体现的高层结构和设计选择。你不需要详细讨论从起始代码继承的内容。利用这个机会突出重要的设计方面，并为评分的TA提供这些领域的更多细节。你在头文件中选择了什么数据结构？其中有哪些不是严格必要的？我们希望你尽可能避免冗余状态，除非你认为并可以证明这样做会带来严重的性能损失。我们强烈建议你通过使用子标题和大纲使这份报告尽可能可读。请不要简单地将你的程序翻译成一段英文。

（b） 替代设计选择——你考虑过的或理想情况下在性能、编写难度（例如，生成无错误实现所需的小时数）、阅读难度（例如，代码行数及其微妙程度或非显而易见的正确性）以及你认为对读者（或做此作业之前的你自己）有趣的其他维度方面进行了评估的。如适用，请包括任何测量数据。我们非常关心帮助你提升想象和描述现实实现选项并权衡其取舍的能力。

（c） 实现挑战。 描述你觉得最棘手的代码部分并解释原因。反思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。你是如何尝试确保你的代码维护你的假设、不变量和前置条件的，你觉得这样做容易还是困难？你是如何调试和测试你的代码的？

（d）残余错误。 尽可能指出并解释代码中仍然存在的任何错误（或未处理的边界情况）。

在你的报告中，请同时填写完成作业所花的小时数和任何其他评论。

"如何提交"的具体方式将在截止日期前公布。

如果有任何问题，请尽快在实验课上通知课程工作人员，或在Ed上发帖提问。祝你好运！