实验检查点 1：将子字符串拼接成字节流

截止时间：1 月 19 日星期日晚上 11:59（延期截止时间：1 月 22 日星期三下午 7 点）

合作政策：与检查点 0 相同。

0 概述

在检查点 0 中，您使用了互联网流套接字从网站获取信息并发送电子邮件，利用了 Linux 内置的传输控制协议（TCP）实现。该 TCP 实现成功创建了一对可靠有序的字节流（一个从您到服务器，另一个方向相反），尽管底层网络仅提供“尽力而为”的数据报服务。这意味着：短数据包可能会丢失、乱序、被篡改或重复。您还在单台计算机的内存中自行实现了字节流抽象。在接下来的几周里，您将亲自实现 TCP，以便在通过不可靠数据报网络分隔的一对计算机之间提供字节流抽象。

⋆我为何要做这件事？在网络领域，许多引人入胜的问题都源于为另一个可靠性较低的服务提供封装或抽象服务。过去 40 年间，研究者和实践者们已成功实现了多种互联网传输方案——消息与电子邮件、超链接文档、搜索引擎、音频视频、虚拟世界、协作文件共享、数字货币等。其中，TCP 协议通过不可靠的数据报提供可靠的双向字节流，堪称经典范例。可以说，TCP 实现是地球上使用最广泛的非平凡计算机程序。

实验作业将指导你以模块化方式逐步实现 TCP 协议。还记得你在 Checkpoint 0 中实现的 ByteStream 吗？后续实验中，你将通过网络传输两个这样的数据流：一个是"出站"ByteStream，用于传输本地应用写入套接字、由你的 TCP 发送给对端的数据；另一个是"入站"ByteStream，用于传输来自对端、供本地应用读取的数据。

本次检查点包含“动手实践”部分和代码实现部分。建议在实验课前先开始代码实现部分，动手实践部分则安排在实验课期间完成。若您为 CGOE 学员，请通过 EdStem 平台协调与其他同学共同完成实践环节的时间。

动手实践环节为今年新增内容，涉及多个动态组件——可能会出现一些技术故障。请在实验课上耐心配合，我们将尽力确保每位同学都能顺利完成。若访问 https://cs144.net 网站时出现错误提示，请在 EdStem 平台公开发帖反馈，我们会及时处理。

1 开始使用

本次 TCP 实现将沿用检查点 0 中使用的 Minnow 库，并新增若干类与测试用例。开始前请：

1. 确保已提交检查点 0 的所有解决方案。请勿修改 src 目录及 webget.cc 以外的文件，否则可能导致与检查点 1 初始代码的合并冲突。

2. 在实验作业的代码仓库内，运行 git fetch 以获取最新版本的实验作业内容。

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

4. 确保构建系统正确配置：`cmake -S . -B build`

5. 编译源代码：`cmake --build build`

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

2 实践环节：为班级搭建的私有网络

我们已为 CS144 课程创建了一个私有网络。这将使你的虚拟机能够直接与班上其他同学的虚拟机互发数据报。要让你的虚拟机加入该网络：

1. 在你的虚拟机上，通过运行 `sudo apt install wireguard` 命令安装“wireguard”软件包

2. 访问 https://cs144.net/wg 并按照指示加入 CS144 私有网络。

3. 成功加入网络后，请根据该页面上的“ping”指引验证连接（这些指引会在您加入网络后显示）。

4. 每次重启虚拟机后，如需与本课程其他同学互发数据报，需重新加入网络。无需每次注册新公钥，但必须重新执行该网页上的命令。这些命令每次操作都相同。

2.1 向一位同学发送 ping 并观察数据报

1. 在您自己的电脑上（例如您的 Mac 或 Windows 机器——不是虚拟机），按照 https://www.wireshark.org/上的说明安装“wireshark”程序。（如果您使用的是 Debian 或 Ubuntu GNU/Linux 系统，安装命令为 sudo apt install wireshark。）

2. 向一位组员询问他们的 IP 地址（即在 https://cs144.net/wg 网页上显示的那个地址）。使用 ping 命令，向您的朋友发送一些“回显请求”数据包，并确保您能收到一些“回显应答”数据包。

3. 提示：□ 您可以通过输入 ctrl -C 来终止“ping”程序。）

□ 您可以通过添加参数-i 0.2 使“ping”命令运行得更快。
这将使其每 0.2 秒（每秒 5 次）发送一次“回显请求”。

你可以在另一个终端中运行以下命令，使“ping”命令打印出当前的统计摘要（而不终止它）：killall -QUIT ping。

4. 在报告中开始撰写，包括以下信息：(a) 你的虚拟机发送“回显请求”到接收到组员虚拟机的“回显回复”之间的平均往返延迟是多少?

(b) 投递率是多少（即“回显请求”中有多少百分比收到了对应的“回显回复”）？ 丢失率是多少（即 100%减去投递率）？ 发送至少 1000 次 ping 以获得可靠的估计（如果使用 ping -i 0.2，这将花费大约三分钟）。

(c) 你是否看到了重复的数据报（ping 命令会显示“DUP”）？

(d) 在 ping 命令运行期间，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
```

此命令将在“wg0”接口（私有类网络）上捕获数据报至文件“/tmp/capture.raw”，同时将其打印到屏幕上。请确保你看到一些“echo request”和“echo reply”行打印出来——这表明你的组员正在接收你的数据报并回复你。

(e) 使用 wireshark 程序检查每台虚拟机上的/tmp/capture.raw 文件。你可能需要先将 capture.raw 文件通过 scp 传输到自己的电脑（如 Mac 或 Windows 机器），再用 wireshark 打开该文件以便使用其图形界面。你能找到 1 月 10 日课程中讨论的互联网数据报字段吗（并与 https://www.rfc-editor.org/rfc/rfc791.html#page-11 上的图表匹配）？

(f) 在你的虚拟机上捕获的数据报与在你朋友的虚拟机上捕获的相同数据报之间是否存在差异？有哪些差异?

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，通过原始套接字按照 1 月 10 日课程中的相同方法向你的朋友发送互联网数据报。可以适当改编该课程中的代码。

1. 向你的组员发送一个 IP 协议为“5”的互联网数据报（需要使用“sudo”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。确保他们能收到！

2. 向你的组员发送一个用户数据报（使用 IP 协议“17”），采用 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。他们可以使用讲座中演示的“nc -u”程序，或使用 UDPSocket 类的 C++程序——随他们喜好！

3. 在你的提交中包含“ip raw.cc”的代码作为本次检查点的一部分。

4. 反向操作，接收来自组员的数据报。

3 实现：将子字符串按顺序排列

作为实验作业的一部分，你正在实现一个 TCP 接收器：该模块负责接收数据报并将其转换为可靠的字节流，供应用程序从套接字读取——就像你的 webget 程序在检查点 0 中从网络服务器读取字节流一样。

TCP 发送方将其字节流分割成较短的段（每个子字符串不超过约 1,460 字节），以便它们能各自装入一个数据报中。但网络可能会重新排序这些数据报，或丢弃它们，或多次传递它们。接收方必须将这些段重新组装成它们最初所处的连续字节流。

在本实验中，你将编写负责重组的数据结构：重组器（Reassembler）。它将接收由字节串组成的子字符串，以及该字符串在更大数据流中首个字节的索引。数据流的每个字节都有其唯一索引，从零开始递增。一旦重组器获知数据流的下一个字节，就会将其写入 ByteStream 的写入端——即你在检查点 0 实现的同一个 ByteStream。重组器的"客户"可以从同一 ByteStream 的读取端进行读取。

接口形式如下：

```
// 插入需要重组到 ByteStream 的新子字符串
void insert( uint64_t first_index, std::string data, bool is_last_substring );

// Reassembler 自身存储了多少字节？
// 此函数仅用于测试；请勿添加额外状态以支持它。
uint64_t count_bytes_pending() const;

// 访问输出流读取器
读取器& reader();
```

⋆为何我要这样做？TCP 对乱序和重复的鲁棒性源于其将字节流的任意片段重新拼接回原始流的能力。在一个可离散测试的模块中实现这一点，将使处理传入数据段变得更加容易。

重组器的完整（公开）接口由 reassembler.hh 头文件中的 Reassembler 类描述。你的任务是实现这个类。你可以向 Reassembler 类添加任何私有成员和成员函数，但不能更改其公开接口。

3.1 Reassembler 内部应存储什么？

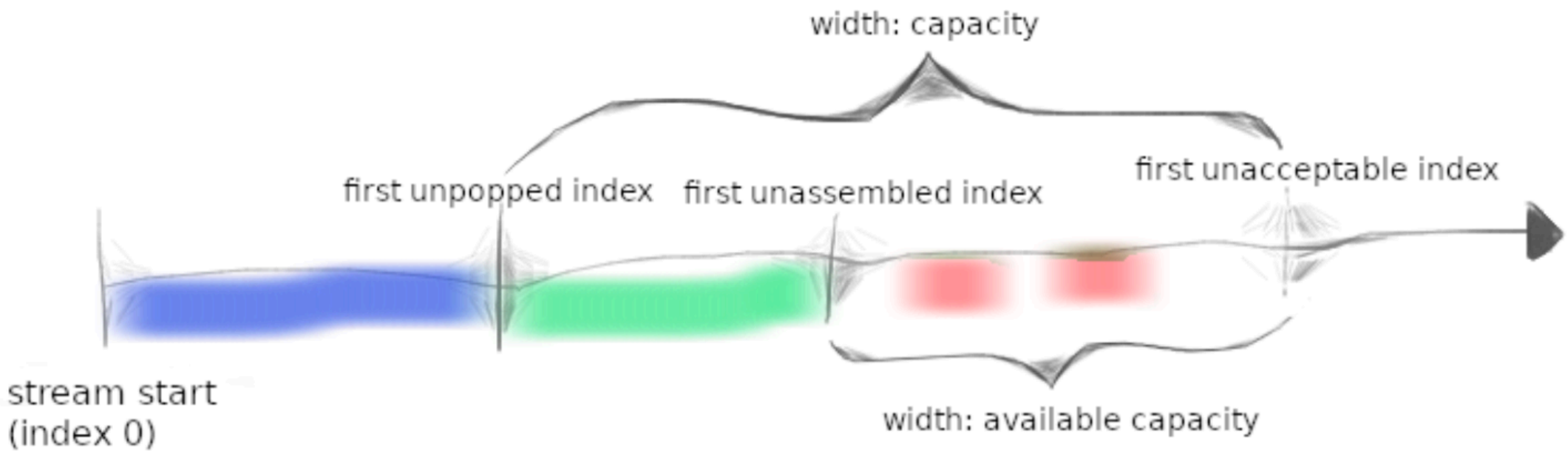
insert 方法向 Reassembler 通知 ByteStream 的一个新片段，以及它在整个流中的位置（子字符串起始的索引）。

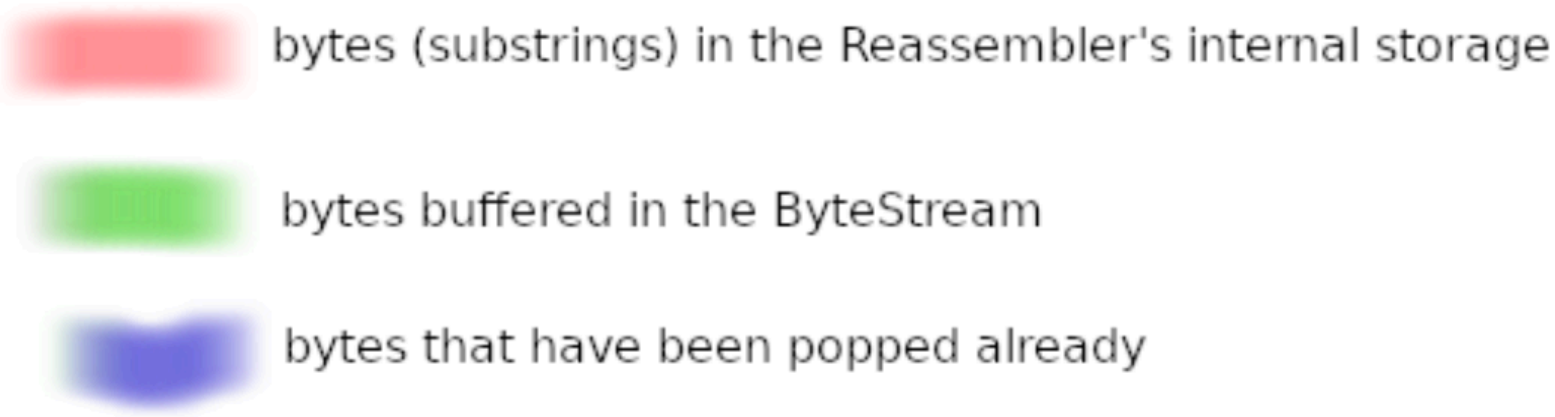
原则上，Reassembler 需要处理三类知识：

1. 流中的下一个字节。一旦这些字节被识别，重组器应立即将它们推送到流中（输出.writer()）。
2. 适合流可用容量但尚无法写入的字节，因为较早的字节仍未知。这些字节应在 Reassembler 内部存储。
3. 超出流可用容量的字节应被丢弃。重组器不会存储任何无法立即或待前期字节就绪后推送到字节流的数据。

此行为旨在限制重组器和字节流的内存使用量，无论传入的子字符串如何到达。我们已在下方图示中说明这一点。"容量"是以下两者的上限：

1. 重组字节流中缓冲的字节数（以绿色显示），以及
2. 可被"未重组"子字符串使用的字节数（以红色显示）





在实现重组器和进行测试时，你可能会发现这张图片很有用——因为“正确”的行为并不总是显而易见的。

3.2 常见问题解答

整个流中第一个字节的索引是什么？零。

我的实现应该有多高效？数据结构的选择再次

此处内容重要。请不要将此视为构建一个极度浪费空间或时间的数据结构的挑战——重组器将是您 TCP 实现的基础。您有多种选择可供考虑。

我们已为您提供了一个基准；任何高于 0.1 Gbit/s（100 兆比特每秒）的速率都是可接受的。顶级的重组器将达到 10 Gbit/s。

不一致的子字符串应如何处理？您可以假设它们不存在。也就是说，您可以认为存在一个唯一的底层字节流，所有子字符串都是其（精确的）切片。

~~不一致的子字符串应如何处理？您可以假设它们不存在。也就是说，您可以认为存在一个唯一的底层字节流，所有子字符串都是其（精确的）切片。~~

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

应该在什么时候将字节写入流？尽快。唯一的情况是
当一个字节前面的字节尚未被“推送”时，该字节不应出现在流中。

提供给 insert()函数的子字符串可以重叠吗？可以。

我需要向 Reassembler 添加私有成员吗？是的。子字符串可能会以
任何顺序，因此你的数据结构必须“记住”子字符串，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被写入。

我们的重新组装数据结构是否可以存储重叠的子字符串？不行。
可以实现一个“接口正确”的重组器来存储重叠的子字符串。但允许重组器这样做会削弱“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，重组器应仅存储该信息的一个副本。

重组器会使用字节流的读取端吗？不会——那是给外部客户用的。重组器只使用写入端。

你预计有多少行代码？当我们运行 ./scripts/lines-of-code
在初始代码上时，它会打印：

```
ByteStream：82 行代码 Reassembler：
26 行代码
```

而当我们在我们的解决方案上运行它时，它会打印：

```
字节流：111 行代码 重组器：85 行代
码
```

因此，重组器的一个合理实现可能需要在初始代码基础上再编写约 50 至 60 行代码。

更多常见问题解答：详情请访问 https://cs144.github.io/lab faq.html。

4 开发与调试建议

1. 你可以在编译后使用命令 `cmake --build build --target check1` 来测试你的代码。
2. 请重新阅读实验 0 文档中关于“使用 Git”的部分，并记得将代码保留在 Git 仓库的主分支上，该仓库是代码最初分发的位置。进行小规模的提交，并使用良好的提交信息，明确说明更改内容及原因。
3. 请努力使你的代码对负责评分风格和健壮性的课程助理（CA）来说易于阅读。为变量使用合理且清晰的命名约定。用注释解释复杂或微妙的代码段。采用“防御性编程”——即明确地

检查函数的前置条件或不变量，一旦发现任何问题就抛出异常。设计时采用模块化原则——识别常见的抽象和行为，尽可能将其提取出来。重复的代码块和冗长的函数会使代码难以理解。

4. 请同时遵循“现代 C++”风格，如 Checkpoint 0 文档所述。cppreference 网站（https://en.cppreference.com）是一个很好的资源，尽管完成这些实验并不需要用到 C++的高级特性。（有时可能需要使用 move()函数来传递无法复制的对象。）

5. 如果构建过程卡住且不确定如何修复，可以删除构建目录（rm -rf build——请注意不要输错命令，因为它会删除你指定的任何内容），然后重新运行 cmake -S . -B build。

5 提交

1. 在提交时，请仅对 src 目录中的.hh 和.cc 文件进行修改。在这些文件中，可以根据需要添加私有成员，但请不要更改任何类的公共接口。

2. 提交任何作业前，请按顺序执行以下步骤：(a) 确保已将全部更改提交至 Git 仓库。

(b) 运行 cmake --build build --target format （以统一代码风格）

(c) 运行 cmake --build build --target check0 （确保自动化测试通过）

(c) cmake --build build --target check1（确保自动化测试通过）(d) 可选步骤：cmake --build build --target tidy（建议改进以遵循良好的 C++编程实践）

3. 在 writeups/check1.md 中撰写报告。该文件应为约 20 至 50 行的文档，每行不超过 80 个字符以便阅读。报告需包含以下部分：

(a) 结构与设计。描述代码中体现的高层结构和设计选择。无需详细讨论从初始代码继承的部分。借此机会突出重要设计方面，并为评分助教提供更详细的理解。你在头文件中选择了哪些数据结构？是否存在非严格必要的部分？我们希望你尽可能避免冗余状态，除非你认为并能够证明这样做会带来严重的性能损失。

强烈建议您通过使用小标题和提纲使这份撰写内容尽可能易读。请不要简单地将您的程序直译成一段英文。

(b) 您考虑过的替代设计方案，最好能从性能、编写难度（例如，实现无缺陷代码所需的小时数）、阅读难度（例如，代码行数及其微妙性或非显而易见正确性的程度）以及您认为对读者（或完成作业前的自己）有趣的其他维度进行评估。如适用，请包含任何测量数据。

(c) 实现过程中的挑战。描述您在代码编写过程中遇到的最大困难部分，并解释原因。反思您是如何克服这些挑战的，以及最终帮助您理解困扰您的概念的因素是什么。您如何尝试确保代码保持您的假设、不变量和前提条件，以及在此过程中您发现哪些方面容易或困难？您是如何调试和测试代码的？

(d) 遗留的缺陷。尽可能指出并解释代码中仍存在的任何缺陷（或未处理的边缘情况）。

4. 在您的报告中，请同时填写完成作业所花费的小时数以及其他任何评论。

5. 关于“如何提交”的具体操作将在截止日期前公布。

6. 如实验课中遇到任何问题，或需通过 Ed 平台提问，请尽快告知课程工作人员。祝好运！