Checkpoint 7 draft: making an Internet + something creative!

Due: end of class (Dec. 5, 11:59 p.m.)

0 Collaboration Policy

Collaboration Policy: Checkpoint 7 asks you to work with as many groupmates as possible—you'll need to connect together to build a class "Internet" and perhaps debug together. Talking about packets and ideas is strongly encouraged! As before, though, we're sticking to the clear line: please do not look at other students' *code* or ask for or receive code from anybody, including a chatbot or former student. As with previous checkpoints, please fully disclose collaborators, LLM prompts, or any gray areas in your writeup—this protects you.

1 Overview

By this point in the class, you've implemented a big portion of the Internet's infrastructure. From Checkpoint 0 (a reliable byte stream), to Checkpoints 1–3 (the Transmission Control Protocol), Checkpoint 5 (an IP/Ethernet network interface) and Checkpoint 6 (an IP router), you have mastered much of the Internet's core infrastructure!

To cap off your accomplishment, you're going to *use* all of your previous labwork to design and run a **"Checkpoint 7 Capstone Internet"** that includes your network stack (host and router) talking to the network stacks implemented by other students in the class. Please use the lab sessions to coordinate and connect, or EdStem to coordinate if you cannot attend.

2 Getting started

1. Make sure you have committed all your solutions. Please don't modify any files outside the top level of the `src` directory, or `webget.cc` and `ip_raw.cc`. You may have trouble merging the Checkpoint 7 starter code otherwise.
2. While inside the repository for the lab assignments, run `git fetch -all` to retrieve the most recent version of the lab assignment.
3. Download the starter code for Checkpoint 7 by running `git merge origin/check7-startercode`.
4. Make sure your build system is properly set up: `cmake -S . -B build`
5. Remember that if you have trouble, you can build "sanitizing" (bug-checking) versions of the applications with `cmake -S . -B build -DSANITIZED_APPS=True`
6. Compile the source code: `cmake -build build`
7. Reminder: please make frequent **small commits** in your local Git repository as you work. If you need help to make sure you're doing this right, please ask a classmate or the teaching staff for help. You can use the `git log` command to see your Git history.

检查点7草稿：制作一个互联网 + 有创意的东西！

截止日期：下课结束时（12月5日，晚上11:59）

0 协作政策

协作政策：检查点7要求你尽可能多地与组员合作——你需要一起连接起来构建一个班级"互联网"，并可能一起调试。强烈鼓励讨论数据包和想法！不过，我们仍然坚持明确的原则：请不要查看其他学生的代码，也不要向任何人索取或接收代码，包括聊天机器人或以前的学长学姐。和之前的检查点一样，请在你的报告书中充分披露合作者、LLM提示或任何灰色地带——这能保护你。

1 概述

此时，课程中你已经实现了一大大部分互联网的基础设施。从检查点0（可靠的字节流），到检查点1-3（传输控制协议），检查点5（IP/以太网网络接口）和检查点6（IP路由器），你已经掌握了互联网核心基础设施的许多内容！

为了圆满完成你的成就，你将使用所有之前的工作来设计和运行一个"检查点7综合互联网"，其中包括你的网络栈（主机和路由器）与其他学生在课程中实现的网络栈进行通信。请使用实验课时间进行协调和连接，如果你无法参加，也可以使用EdStem进行协调。

2 开始

1. 确保你已经提交了所有解决方案。请不要修改src目录顶层以外的任何文件，或者webget.cc和ip_raw.cc。否则，你可能会在合并检查点7的启动代码时遇到问题。
2. 在实验作业的代码库中，运行 `git fetch –all` 以获取最新版本的实验作业。
   the most recent version of the lab assignment.
3. 通过运行 `git merge origin/check7-startercode` 下载 Checkpoint 7 的初始代码。.
4. 确保您的构建系统已正确设置：`cmake -S . -B build`
5. 请记住，如果您遇到问题，可以构建"清理"（错误检查）版本的应用程序，使用 `cmake -S . -B build -DSANITIZED{v1}APPS{v2}True`
   of the应用程序使用cmake -S . -B build -DSANITIZED_APPS=True
6. 编译源代码：cmake –build build
7. 提醒：请在你本地的 Git 仓库中频繁提交小改动。如果你需要帮助以确保你这样做正确，请向同学或教学人员寻求帮助。你可以使用 `git log` 命令查看你的 Git 历史。

3 Read and Understand the Starter Code

As part of the checkpoint 7 starter code, we've given you two new "apps" that use your ByteStream, TCP implementation, NetworkInterface, and Router.

- `apps/tcp_eth_udp.cc` implements a **"full stack" endpoint**: your TCP implementation creates TCP segments, which our code encapsulates in Internet datagrams. These are given to your NetworkInterface, which encapsulates them in Ethernet frames. The `tcp_eth_udp` tool encapsulates these Etheret frames in user datagrams and sends them over the normal Internet.
- `apps/fun_router` implement a compatible **IP router** by giving your Router a command-line interface. Command-line flags can create interfaces and add routes to the router's routing table. The Router's NetworkInterfaces also send Ethernet frames, encapsulated in user datagrams, over the Internet.

1. **Read through the source code** of these two tools and work to understand how they work and what they're doing.
2. One key point: both of these tools send IP-in-Ethernet-in-UDP-in-IP. As a result, there are *two layers* of IP addresses at work: the "physical" addresses used in the outermost layer of Internet datagrams sent over the real Internet, and the "virtual" addresses used in the "Checkpoint 7 Capstone Internet" that you'll design in this checkpoint.
3. When you use these tools, the "physical" addresses will correspond to real IP addresses that can reach another student's computer. This can include the CS144 virtual private network you previously used in checkpoints 1 and 3; from these apps' point of view, that's also the "real" Internet.
4. By contrast, the "virtual" addresses will be your own invention—**we want you to design an Internet** with an interesting topology, as many routers as possible routing packets, and the appropriate routing tables so that many students can simultaneously communicate with each other over this "virtual" Internet.

4 Make a Tiny Internet on Your Own VM

To get started, we're going to recreate the historic 1969 "first ARPANET connection" (the Internet's predecessor) between UCLA and Stanford, inside your computer. You will make a mini-Internet whose **virtual** topology looks like the below, by sending TCP segments between:

- a TCP server at "Stanford" with the IP address 50.9.8.7
- and a TCP client at "UCLA" with the IP address 80.6.5.4

3 阅读并理解 StarterCode

作为检查点 7 起始代码的一部分，我们为你提供了两个新的“应用程序”，它们使用你的 ByteStream、TCP 实现、NetworkInterface 和 Router。

- apps/tcp_eth_udp.cc 实现了一个“完整栈”的端点：你的 TCP 实现创建 TCP 段，我们的代码将其封装在互联网数据报中。这些数据报会交给你的 NetworkInterface，它将它们封装在以太网帧中。tcp_eth_udp 工具将这些以太网帧封装在用户数据报中，并通过普通互联网发送它们。
- apps/fun_router 通过为你的 Router 提供命令行界面来实plement 兼容的 IP 路由器。命令行标志可以创建接口并将路由添加到路由器的路由表中。Router 的 NetworkInterfaces 也会发送以太网帧，这些帧被封装在用户数据报中，并通过互联网发送。

1. 阅读这两个工具的源代码并努力理解它们
   工作和他们正在做什么。

2. 一个关键点：这两种工具都发送 IP-in-Ethernet-in-UDP-in-IP。因此，有两个 IP 地址层在工作：最外层互联网数据报在真实互联网上发送时使用的“物理”地址，以及“Checkpoint 7 Capstone 互联网”中你将在本次检查点设计的“虚拟”地址。

3. 当你使用这些工具时，“物理”地址将对应可以到达另一个学生计算机的真实 IP 地址。这包括你在检查点 1 和 3 中之前使用的 CS144 虚拟专用网络；从这些应用程序的角度来看，这也是“真实”互联网。

4. 相比之下，“虚拟”地址将是你自己的发明——我们希望你设计一个具有有趣拓扑结构的互联网，尽可能多的路由器路由数据包，以及适当的路由表，以便许多学生可以同时通过这个“虚拟”互联网相互通信。

4 在自己的虚拟机中创建一个微型的互联网

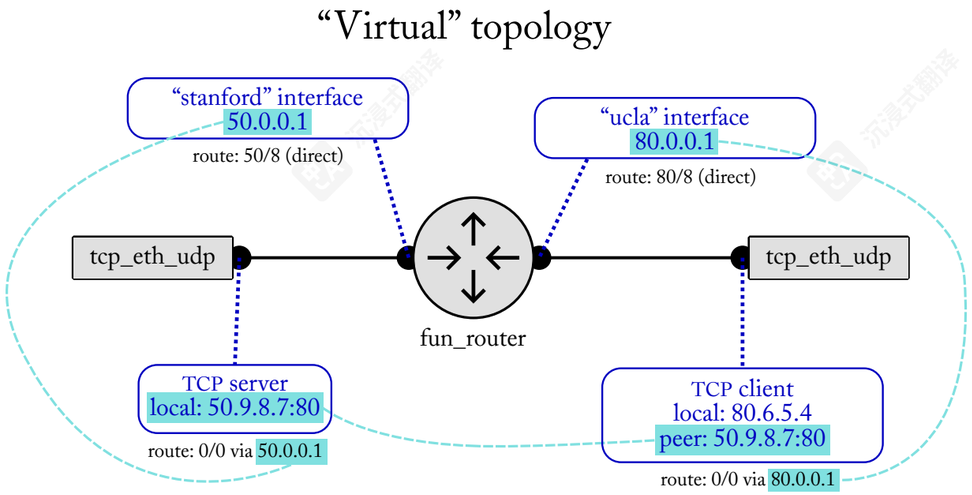
要开始，我们将要在您的计算机中重现历史性的1969年“首次ARPANET连接”（互联网的前身）——在加州大学洛杉矶分校和斯坦福大学之间。您将通过在以下设备之间发送TCP段来创建一个微型互联网，其虚拟拓扑结构如下：

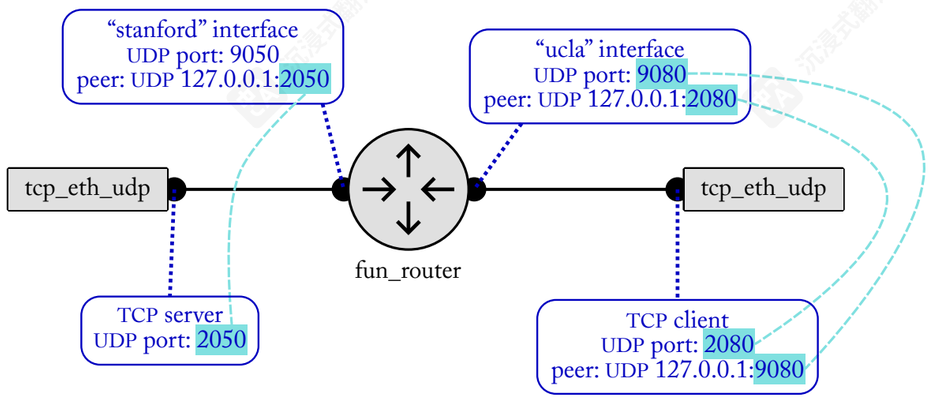
- 一台位于“斯坦福”的TCP服务器，IP地址为50.9.8.7
- 以及一个位于“加州大学洛杉矶分校”的TCP客户端，其IP地址为80.6.5.4

- through a router with two interfaces: one for Stanford's network (with a route pointing to it for the range 50/8) and one for UCLA's network (80/8)

These IP datagrams will be encapsulated inside Ethernet frames that are inside user datagrams that are, in turn, inside "physical" IP datagrams. For now, these "physical" IP datagrams will all stay inside your computer, whose internal IP address is 127.0.0.1.

The complete "virtual" and "physical" addresses will look like the below. Take some time to study this diagram and understand it—it's okay if it seems like a lot at first blush.



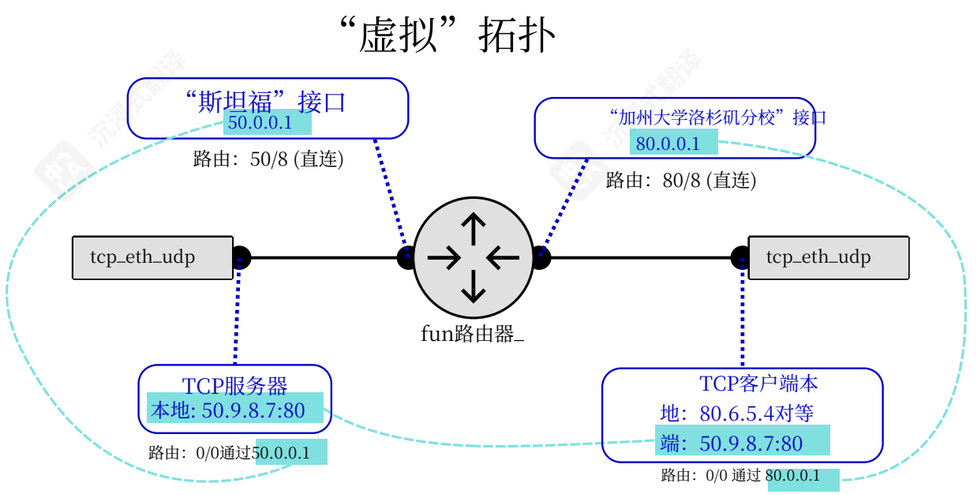


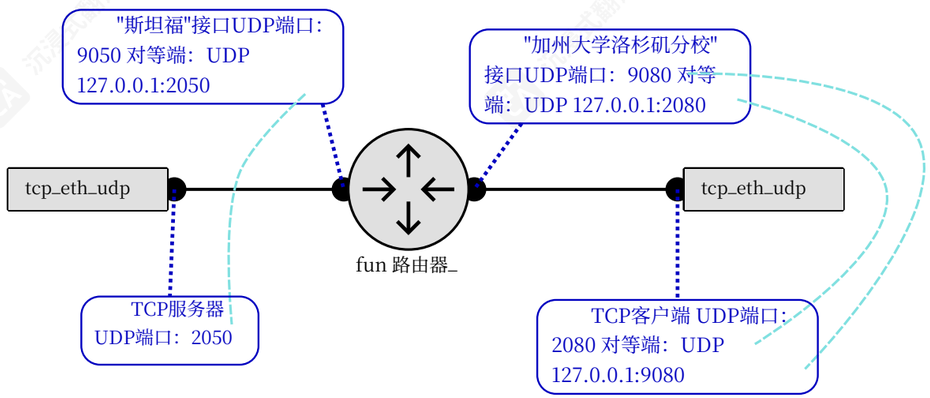
- 通过一个具有两个接口的路由器：一个用于斯坦福的网络（有一个路由指向它，范围为50/8），另一个用于加州大学洛杉矶分校的网络（80/8）

一个用于斯坦福的网络（有一个路由指向它，范围为50/8）和一个用于加州大学洛杉矶分校的网络（80/8）

这些IP数据报将被封装在以太网帧中，而以太网帧又封装在用户数据报中，用户数据报再封装在"物理"IP数据报中。目前，这些"物理"IP数据报将都保留在你的计算机内部，其内部IP地址为127.0.0.1。

完整的"虚拟"和"物理"地址将如下所示。花些时间研究这个图表并理解它——一开始看起来很多也是正常的。





To run this topology with the `fun_router` and `tcp_eth_udp` apps, you'll want to open **four** terminals on your VM.

- On the first terminal, run `sudo tshark -ni lo -d udp.port==2080,eth 'port 2080'` to **monitor** the packets in flight. The `-d udp.port==2080,eth` tells `tshark` to interpret the user-datagram payload as Ethernet, possibly with IP and TCP inside it.
- On the second terminal, run

```
./build/apps/fun_router \
interface:stanford:50.0.0.1:9050:127.0.0.1:2050 \
interface:ucla:80.0.0.1:9080:127.0.0.1:2080 \
route:50.0.0.0:8:stanford route:80.0.0.0:8:ucla
```

  . This command has all the information (virtual and physical) for the router in the above diagram, including the interface names. You can run `./build/apps/fun_router` by itself to see the help message, which describes the arguments.
- On the third terminal, run the TCP server (at "Stanford"): `./build/apps/tcp_eth_udp server 2050 50.0.0.1 50.9.8.7:80`. (You can run `./build/apps/tcp_eth_udp` by itself to see the help message.)
- On the fourth terminal, run the TCP client (from "UCLA"): `./build/apps/tcp_eth_udp client 2080 127.0.0.1:9080 80.0.0.1 80.6.5.4 50.9.8.7:80`

If everything works as intended, your "tshark" terminal will show an ARP exchange and then a TCP connection establishment. Your "router" terminal will show lots of packets in both directions. Your TCP client and server terminals will show a connection established (as you saw on checkpoint 3).

Try typing "LO" into the UCLA terminal and hitting Enter. Does it get to Stanford?

5 If you have trouble...

- Use the tshark terminal (or wireshark), and the debugging output from the router, to debug.
- Consider building the "sanitizing" (bug-checking) version of the `endtoend` program. It will find many instances of undefined behavior and use of invalid addresses in your code. (See above for directions.)
- Run the entire unit-test suite (including new tests your classmates have contributed this quarter) with `cmake -build build -target test`

要使用 fun_ 路由器和 tcp_eth_udpapps 运行此拓扑，您需要在您的虚拟机（VM）上打开四个终端。

- 在第一个终端上，运行 sudo tshark -ni lo -d udp.port==2080,eth ' port 2080' 来监控正在传输的数据包。-d udp.port==2080,eth 命令告诉 tshark 将用户数据报载荷解释为以太网，其中可能包含 IP 和 TCP 数据包。
- On the second terminal, run

```
./build/apps/fun_路由 \ 接口:斯坦福:
50.0.0.1:9050:127.0.0.1:2050 \ 接口:加州大学洛杉矶分校:
80.0.0.1:9080:127.0.0.1:2080 \ 路由:50.0.0.0:8:斯坦福 路由:
80.0.0.0:8:加州大学洛杉矶分校
```

  . 这个命令包含了上述图中路由器的所有信息（虚拟和物理），包括接口名称。你可以单独运行 ./build/apps/fun_router 来查看帮助信息，其中描述了参数。
- 在第三个终端上，运行TCP服务器（在"斯坦福"处）：./build/apps/tcp_eth_udp server 2050 50.0.0.1 50.9.8.7:80 . (您可以通过单独运行 ./build/apps/tcp_eth_udp 来查看帮助信息。)
- 在第四个终端上，运行TCP客户端（从"加州大学洛杉矶分校"处）： ./build/apps/tcp_eth_udp client 2080 127.0.0.1:9080 80.0.0.1 80.6.5.4 50.9.8.7:80

如果一切按预期工作，您的"tshark"终端将显示ARP交换，然后是TCP连接建立。您的"路由器"终端将在双向显示大量数据包。您的TCP客户端和服务器终端将显示连接已建立（正如您在检查点3中看到的那样）。

尝试在加州大学洛杉矶分校终端中输入"LO"并按回车。能到达斯坦福吗?

5 如果遇到问题...

- 使用tshark终端（或wireshark），以及路由器的调试输出，来调试。
- 考虑构建端到端程序的"清理"版本（即bug检查版本）。它将发现代码中许多未定义行为和无效地址的使用。（参见上文说明。）
- 使用cmake –build build –target test运行整个单元测试套件（包括本学期同学们贡献的新测试）

6 An Internet of three people

Now that you have a working mini-Internet inside your own computer, try doing the same exercise with two partners, over the CS144 class network. One person should run the "router," one of you runs the TCP client, and the third person runs the TCP server. You may have to rejoin the CS144 class network (use the directions from checkpoint 1, and visit https://cs144.net).

Change the *physical* addresses to match your actual physical addresses (starting with `10.144`). You shouldn't have to change any of the *virtual* addresses.

7 Make a bigger Internet

During the next three weeks and two lab sessions, work to design the **biggest** virtual Internet you can (most routers and hosts). Each student will run a network, and we're hoping many of you will connect to create one big Inter-network. Assign IP blocks to one another, and subnetworks of IP blocks, and set up appropriate routing tables to connect several student-run routers in an "interesting" topology. Arrange for several TCP connections to be occurring at the same time over this network. In your lab report, include a diagram of the Internet's topology with all of the relevant addresses, routing tables, and connections—and the names of the students running each router or endpoint.

8 Something creative

Finally, please take the `webget`, `ip_raw`, `tcp_ipv4`, `tcp_udp_eth`, and `fun_router` programs as a jumping-off point to write a program that does **something creative** that involves computer networking. This could be a tiny game, a chat program, a tiny email client or Web browser, a traceroute/mtr program, a Web server, some kind of crazy router, or anything that inspires you that relates to any of the subject matter of this class.

9 Submit

Write a report in the form of a PDF. Please include:

1. A description of your three-person Internet. Who participated (please include SUnet IDs) in what roles? Were you able to exchange data in all directions? Can you send a 1-MB file and have it arrive exactly the same (using the same procedure as in checkpoint 3)? Did anything interesting happen (or did you find any bugs you had to fix)?
2. A description of your "bigger Internet." Please diagram the topology (hand-drawn and photographed is fine). Include the relevant address ranges, routing tables, and connections, and the names/SUNet IDs of the students running each router or endpoint. What workloads/flows did you run over this network? Were there any surprises?

三人互联网

现在，你已经在自己的电脑里搭建了一个迷你互联网，试着和两个伙伴在CS144课程网络中做同样的练习。一个人应该运行“路由器”，你们中的一个运行TCP客户端，第三个人运行TCP服务器。你可能需要重新加入CS144课程网络（使用检查点1的说明，并访问https://cs144.net）。

将物理地址更改为与您的实际物理地址匹配（以10.144开头）
您不应该需要更改任何虚拟地址。

7 创建更大的互联网

在接下来的三周和两次实验课中，工作以设计你能设计的最大虚拟互联网（最多的路由器和主机）。每位学生将运行一个网络，我们希望许多同学能连接起来创建一个大的互联网络。互相分配 IP 块和 IP 块的子网络，并设置适当的路由表以连接几个学生运行的路由器，形成一个“有趣”的拓扑结构。安排在这个网络上同时进行多个 TCP 连接。在你的实验报告中，包括互联网拓扑结构的图，以及所有相关的地址、路由表和连接——以及运行每个路由器或端点的学生姓名。

8 有创意的内容

最后，请以 webget、ip_raw、tcp_ipv4、tcp_udp_eth 和 fun_router 程序为起点，编写一个程序，完成一些涉及计算机网络的有创意的内容。这可以是一个小游戏、一个聊天程序、一个微型的电子邮件客户端或网页浏览器、一个 traceroute/mtr 程序、一个网页服务器、某种疯狂的路由器，或者任何与本课程主题相关的能激发你的内容。

9 提交

以PDF格式撰写报告。请包括：

1. 描述你的三人互联网。谁参与了（请包含SUnet ID），在什么角色中？你们是否能在所有方向上交换数据？能否发送一个1-MB文件并确保它完全相同地到达（使用检查点3中相同的步骤）？是否发生了什么有趣的事情（或者你是否发现了需要修复的任何错误）？

2. 描述你的“更大的互联网”。请绘制拓扑图（手绘并拍照即可）。包括相关的地址范围、路由表和连接，以及运行每个路由器或端点的学生的姓名/SUNet ID。你们在这个网络上运行了哪些工作负载/流量？是否有任何意外？

3. Describe the creative thing you did! Show screenshots or describe what it does, as appropriate. Include an excerpt of the most salient or most interesting *piece of code* that you wrote for this purpose. And... why is this cool (in your view)?
4. Any reflections you have on the class or recommendations for future material to include in the CS144 lab in future years. Do you want to see a checkpoint on congestion control, or real-time audio or video?

The CAs and Keith are eager to read what you all come up with. Thank you for a great quarter, and happy holidays!

3. 描述你做的创意项目！根据需要展示截图或说明其功能。包含你为此目的编写的最突出或最有趣代码片段的摘录。还有……你认为这为什么酷（从你的角度看）？

4. 你对课程的任何反思，或对今后CS144实验课应包含的未来材料的建议。你想看到关于拥塞控制的检查点，还是实时音频或视频?

CAs和Keith都迫不及待想看你们想出的内容。感谢这季度的工作，祝节日快乐！