CS144：计算机网络导论 2025年秋季

检查点 4: 测量真实世界

截止日期：10月26日（周日）晚上11:59

0 合作政策

合作政策：与检查点0相同。请在你的报告中完全披露任何合作者或任何灰色地带——坦诚是最好的策略。

1 概述

到目前为止，你已经以几乎完全符合标准的方式实现了传输控制协议。TCP实现可以说是世界上最流行的单一计算机程序，存在于数十亿设备中。大多数实现使用与你不同的策略，但因为所有TCP实现共享一种通用语言，它们都是可互操作的——每个TCP实现都可以与互联网上任何其他TCP实现成为对等方。本检查点不会使用你的TCP实现：它关于测量一些真实互联网路径的长期统计数据。

要完成本检查点，我们希望你选择并表征至少三条"有趣的"互联网路径。每条路径将在一台计算机（例如你的笔记本电脑，或myth/cardinal机器）和另一台主机之间。要使路径"有趣"，它要么(1)跨越相当大的距离（RTT大于100毫秒），要么(2)沿途包括至少一条"有趣的"链路（例如Wi-Fi、系留蜂窝或卫星链路）。理想情况下，你的最终报告将包括混合——至少一条长距离路径（理想情况下没有任何无线链路）和至少一条包括"有趣"链路的路径。

对于每条路径，请至少测量以下统计数据：

2 收集数据

在互联网上选择一个远程主机进行ping（从你的计算机或VM用ping测量）。一些远距离路径的可能性：

www.cs.ox.ac.uk（牛津大学计算机科学系Web服务器，英国）

162.105.253.58（北京大学计算中心，中国）

www.canterbury.ac.nz（坎特伯雷大学Web服务器，新西兰）

41.186.255.86（MTN卢旺达）

CS144专用网络上的朋友的VM

首选：距你至少100毫秒RTT的原创选择，或需要跨越无线链路的选择

使用 `mtr` 或 `traceroute` 命令跟踪你的VM与此主机之间的路由。

运行至少一小时的ping来收集此互联网路径的数据。使用类似 `ping -D -n -i 0.2 hostname | tee data.txt` 的命令将数据保存在"data.txt"文件中。（`-D` 参数使ping记录每行的时间戳，`-i 0.2` 使其每0.2秒发送一个"echo请求"ICMP消息。`-n` 参数使其跳过使用DNS将回复IP地址反向查找为主机名。）

注意：每0.2秒发送默认大小的ping是可以的，但请不要以比这更快的速度向任何人发送洪泛流量超过几秒钟。

3 分析数据

如果你每秒发送五个ping持续一小时，你将发送大约3,600个echo请求（= 5 x 3600），其中我们期望绝大多数在ping输出中收到了回复。使用你选择的编程语言和图表工具，请至少计算并绘制以下信息：

整个时间间隔内的总体交付率是多少？换句话说：收到了多少echo回复，除以发送了多少echo请求？（注意：GNU/Linux上的ping不会打印任何关于未收到的echo回复的消息。你必须通过查找缺失的序列号来识别缺

失的回复。）

最长的连续成功ping串（全部依次收到回复）是多少？

最长的连续丢失突发（全部依次未收到回复）是多少？

你能从 `mtr` 或 `traceroute` 结果中的名称弄清楚路径去了哪里（地理上）吗？它走了你预期的路径还是没有预期到的路径？

生成一个图表，显示"数据包丢失"随时间的自相关性。换句话说：

给定echo请求 #N收到了回复，echo请求 #(N+k)也成功收到回复的概率是多少？在你的图表中，为k在-10到10（含）之间的每个值包括一个条形。

给定echo请求 #N没有收到回复，echo请求 #(N+k)也没有收到回复的概率是多少？

这些数字（条件交付率）与第一个问题中的整体"无条件"数据包交付率相比如何？丢失有多独立或"突发性"？

整个时间间隔内观察到的最小RTT是多少？（这可能是真实MinRTT的合理近似...）

整个时间间隔内观察到的最大RTT是多少？

制作RTT作为时间函数的图表。用实际时间标注x轴（覆盖超过一小时的时间段），y轴应为RTT的毫秒数。

绘制观察到的RTT分布的累积分布函数图。这是一个图表，其中x轴是每个观察到的RTT值，y轴是小于或等于此数值的样本比例（因此y轴将从0到1）。分布大致是什么形状？

制作"ping #N的RTT"与"ping #N+1的RTT"之间相关性的散点图。x轴应为第一个RTT的毫秒数，y轴应为第二个RTT的毫秒数。RTT随时间的相关性如何？

做一些简短的（少于10秒）实验，通过增加数据包大小和频率来发送更高数据速率的ping。在Linux上，你可以用"-s"参数增加echo请求（和回复）的大小（不要使用大于1400的值），你可以通过减少给"-i"参数的间隔来增加echo请求的频率。你可以用"-c"参数告诉ping在一定数量的请求后停止。制作回复的总体数据速率（数据包大小 x 回复数 / 总持续时间）作为你的echo请求数据速率函数的图表。它是否在某个值处趋于平稳？你能获得的最大吞吐量是多少？

你从数据中得出什么结论？网络路径是否按你预期的方式表现？从图表和汇总统计数据中看，有什么（如果有的话）让你感到惊讶？

你能在你表征的这条路径和其他网络路径之间做出什么有趣的比较？

请通过Gradescope以PDF格式提交你的报告。