

Lab5

Z5216214

Haojin Guo

Lab5 操作说明 - Congestion Control - NOAH Edu

Exercise0: Revision of TCP Congestion Control

Q1: TCP Reno 下的两次下降的区别是什么?

A1: 三次重复 ACK与 Timeout 的区别

三次重复 ACK - 快速重传 - TCP Timer (时间的范围) : 虽然我们没有 Timeout, 但是我们认为可以重传了, 基于累计确认机制。

Q2: 3 的位置发生了什么?

A2: 指数增长的情况下 - Slow Start 慢启动, CWND *= 2, CWND 初始值 = 1.

Q3: 4 的位置发生了什么?

A3: 线性增长, 加性增 - 乘性减 AIMD (Congestion Avoidance 拥塞避免) : 因为这是一个更佳谨慎的策略, 因为SS这个过程, 会发生指数性的增长: 从 8 - 16, 128 - 256, 范围会很大, 我们很难确认到底是在哪里是我们的一个 CWND 极限值。

所以我们一般说, 到了阈值 Threshold, 我们进行了从 SS - AIMD 的变化。既保证了 CWND 大小的增加, 同时, 降低了直接 Timeout 的风险。

Q4: 为什么 CWND 的增加过程; 3比4快?

A4: 因为在每一个 Round, RTT, Round Trip Time, 秒。Slow Start 是 *2, AIMD 是 +1

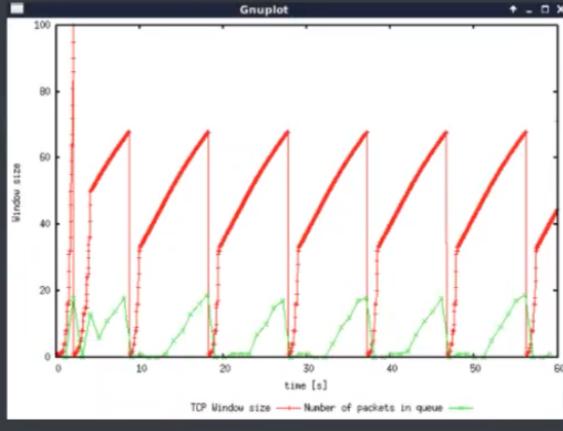
Q5: 你能解释一下 2 之后发生了什么吗?

A5: 当 TCP Reno 发生 Timeout 后, 首先 CWND 恢复到 1, 然后 Threshold 改变成发生 Timeout 时 CWND 的一半。在之后进行 Slow Start, 当 CWND 抵达 Threshold 后, 进行 AIMD。

Exercise1: Understanding TCP Congestion Control using ns-2

Q1 请保证在 LAB 中运行，本地图片不显示

1. 下载 tpWindow.tcl Window.plot
2. 进入文件目录，执行命令 ns tpWindow.tcl 150 100ms
3. 执行命令 gnuplot Window.plot
4. 得到gnuplot

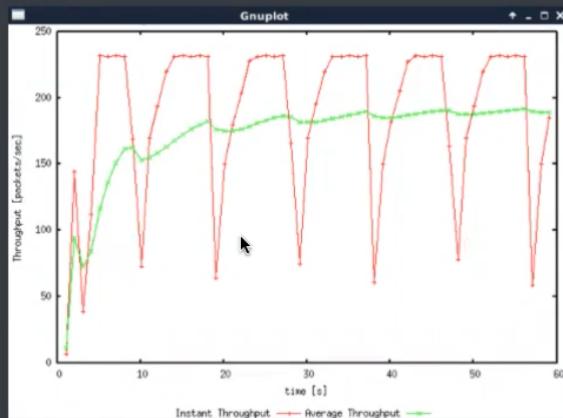


在这种情况下，TCP流达到的拥塞窗口的最大大小是多少？当拥塞窗口达到这个值时，TCP流会怎么做？为什么要这样做？接下来会发生什么？在你的提交报告中加入该图。

- Reno 当出现了3 ACK CWND /= 2, 然后 Threshold = CWND/2
- 如果出现了timeout。我们重新启动。SS。把threshold set 出现timeout的CWND/2
- Tahoe 无论三次ACK还是timeout 都会发生回到CWND = 1时进行SS。
- 在threshold位置从SS -> AIMD （拥塞避免）
- Tahoe 无论是timeout 还是3ACK都会恢复到1的时候重新进行SS

Q2

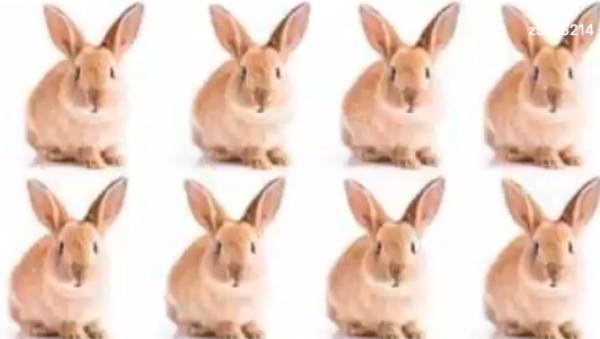
1. 下载 WindowTPut.plot
2. 执行命令 gnuplot WindowTPut.plot
- 3.



从我们使用的模拟脚本中，我们知道数据包的有效载荷 payload 是500 Bytes。请记住，IP Header和 TCP Header的大小各为20 Bytes。忽略任何其他头文件。在这种情况下，TCP的平均吞吐量是多少 calculating the throughput in bps? (包括每秒的数据包数和bps)

吞吐量bps₂₅₂₁₆₂₁₄，单位 bps = bit/sec

Rabbit



Rabbbyte

What is the average throughput of TCP in this case? (both in number of packets per second and bps)

找到图像中的绿色线 -> 趋近于 190 左右

(500TCP payload + 20 TCP header) + 20 IP Header

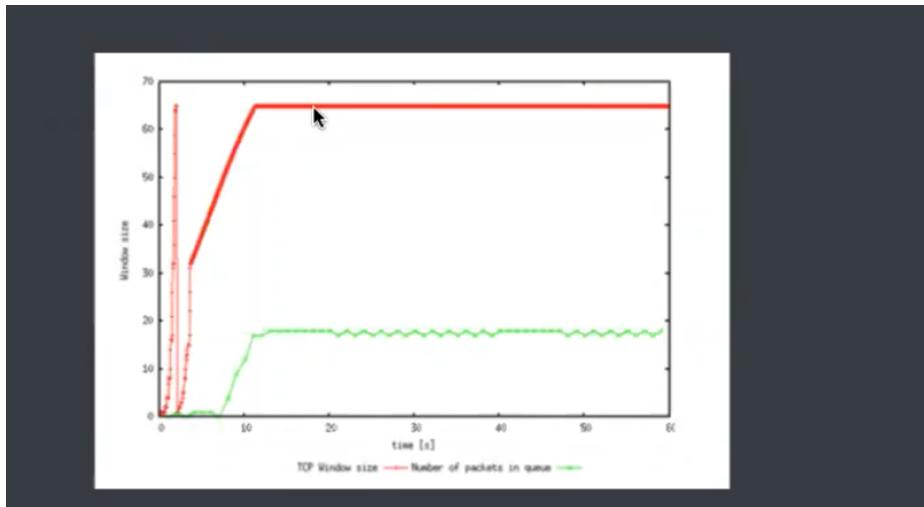
(540 * 8bit/Bytes) * 190packet/sec = XXXXbps

或者只算 payload (500 * 8bit/Bytes) * 190packet/sec = XXXXbps

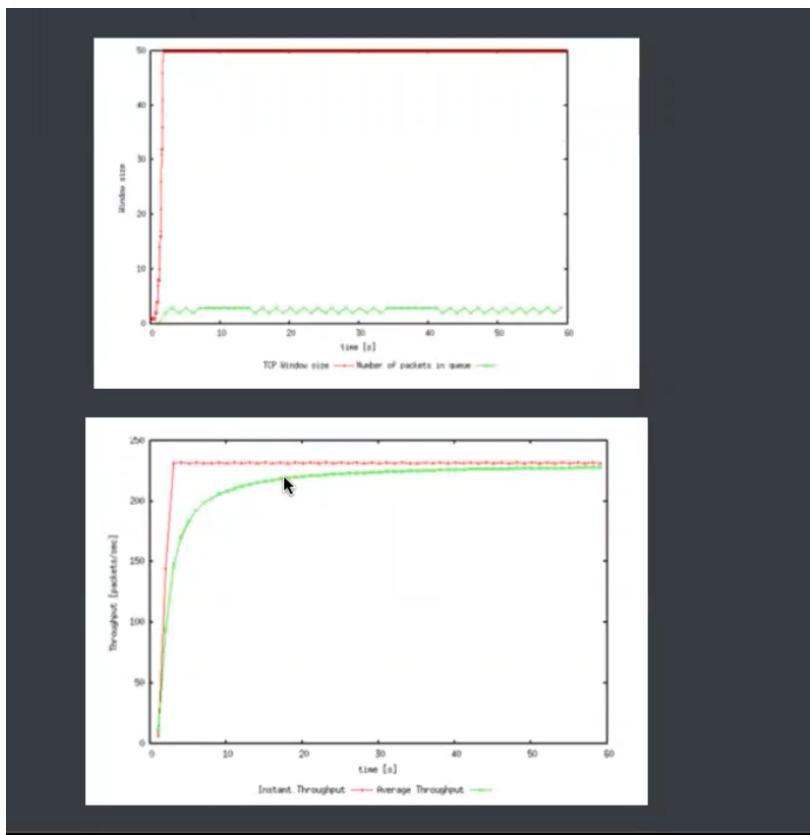
Q3

重新运行上面的脚本，每次使用不同的最大拥塞窗口大小值，但RTT相同（即100ms）。TCP是如何响应这个参数的变化的？找出最大拥塞窗口的值，在这个值上，TCP停止振荡（即不再上下移动），达到稳定的行为。此时的平均吞吐量（以包和bps为单位）是多少？实际平均吞吐量与链路容量（1Mbps）相比如何？

1. 改变第一问中的winodw size，尝试多个不同的window size，最终得出结论
2. 当最大拥塞窗口大小<= 66时，在窗口大小首次返回1后，我们将阈值降低到最大窗口大小的一半，之后开始慢启动，拥塞窗口增大到我们设置的最大拥塞窗口大小。
3. 例如：winodw size = 65
4. ns tpWindow.tcl 65 100ms
5. gnuplot Window.plot



1. 如果我们将最大拥塞窗口大小设置小于等于50,TCP以慢速启动快速启动，然后窗口大小增加到最大拥塞窗口大小，之后趋于稳定。
2. 令window size = 50
3. ns tpWindow.tcl 50 100ms
4. gnuplot Window.plot
5. gnuplot WindowTPut.plot

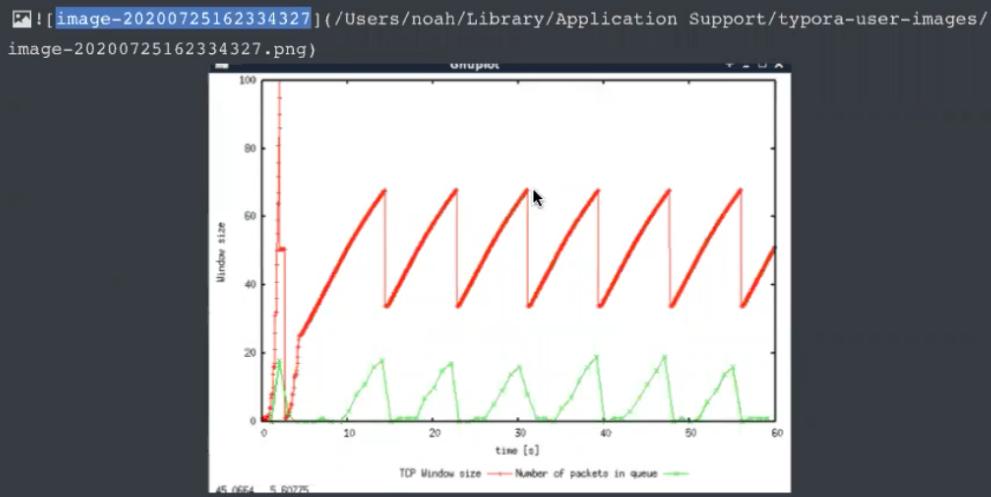


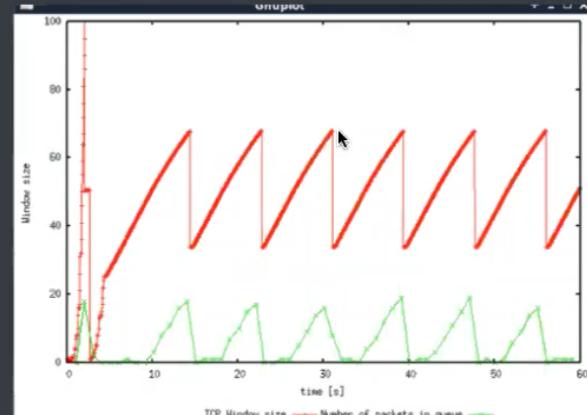
查看绿色线 220~220 * 540 * 8 = 1Mbps 相差不大的。

Q4

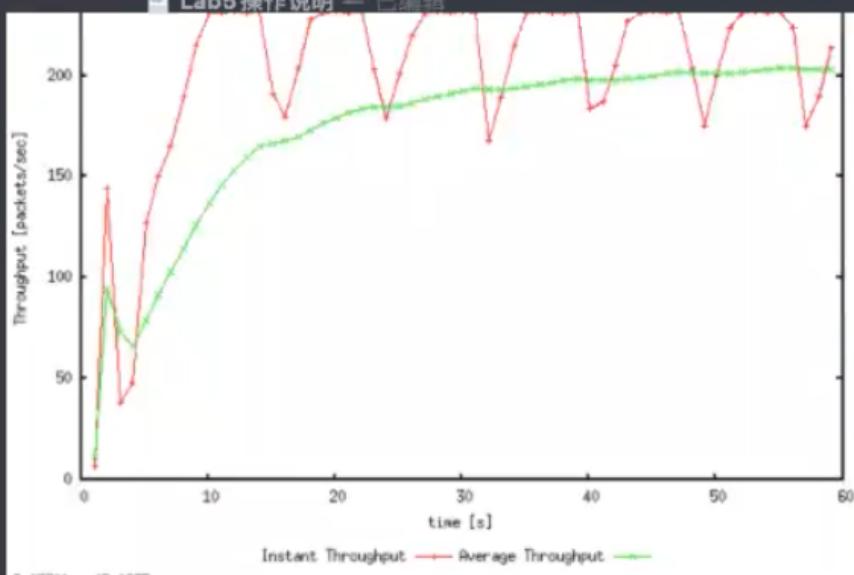
重复问题1和2（不是问题3）中概述的步骤，但对于TCP Reno来说。比较这两种实现的图表，并解释其差异。提示：比较每种情况下拥塞窗口回到零的次数）。两种实现中的平均吞吐量有何不同？

1. 更改tpWindow.tcl，找到`set tcp0 [new Agent/TCP]`，改成`set tcp0 [new Agent/TCP/Reno]`
2. 执行命令
3. `ns tpWindow.tcl 150 100ms`
4. `gnuplot Window.plot`
5. `gnuplot WindowTPut.plot`

 ! [image-20200725162334327] (/Users/noah/Library/Application Support/typora-user-images/image-20200725162334327.png)



Lab5 操作说明 — 已编辑

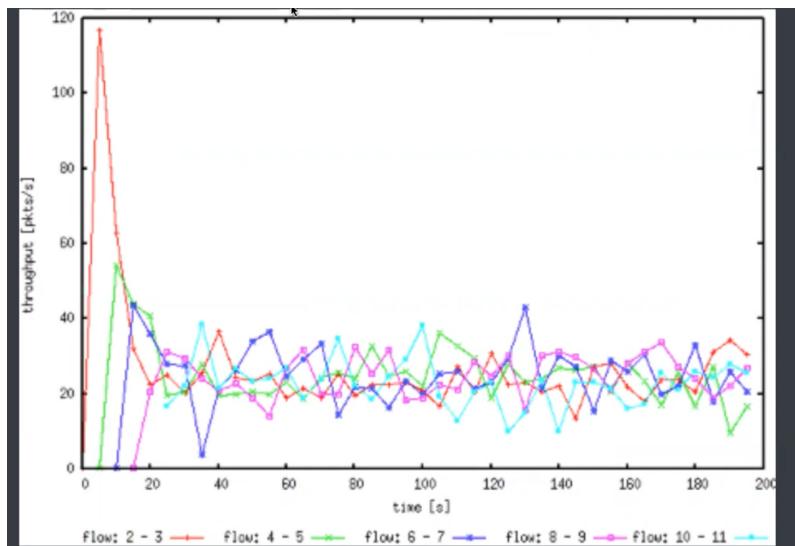


TCP Reno 的优势就是当出现了 3次ACK时（快速重传机制），回到 CWND 的一半进行 AIMD（拥塞避免）。

观察绿色线条的数值（200 左右），理论上是大于 TCP Tahoe 的（190 左右）。

Exercise2 – Congestion control的fairness

1. 下载tp_fairness.tcl, fairness_pkt.plot
2. 执行
3. `ns tp_fairness.tcl`
4. gnuplot fairness_pkt.plot



问题1：每个流量是否都能平等地获得共同链路的容量（即TCP是否公平）？请解释你从哪些观察中得出这个结论。

观察不同颜色图像的变化规律以及趋近值。

问题2：TCP的吞吐量会怎样？当一个新的流被创建时，原有的TCP流的吞吐量会发生什么变化？解释导致这种行为的TCP机制。论证你认为这种行为是公平还是不公平。

Exercise3 – TCP – UDP 的公平性问题

问题1：如果链路的容量是5Mbps，你希望TCP流和UDP流的表现如何？
z5216214

观察谁会回到横坐标轴，这代表了他发生了（Congestion Control - 回到 CWND = 1）

问题2：为什么一个流量比另一个流量的吞吐量高？试解释是什么机制迫使这两个流稳定到观察到的吞吐量。

UDP best effort - flow control 不存在 congestion control （一直在发）

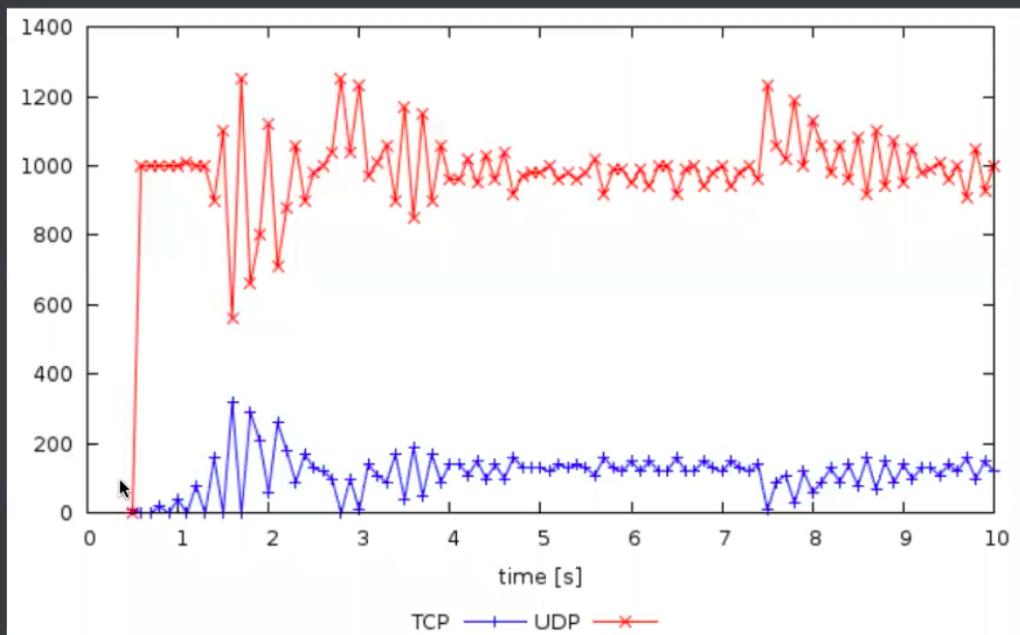
TCP 因为存在 flow control congestion control

问题3：当我们的连接必须与其他流竞争同一链路时，请列出使用UDP而不是TCP进行文件传输的优点和缺点。如果大家都因为同样的原因开始使用UDP而不是TCP，会发生什么？

UDP pros cons

快。保证速度。不可靠。上层application

我们互相都在竞争，但是没有人控制。网络是不是会出现congestion。TCP，根据congestion control 越发越少。最后逐渐不发了。



问题1：如果链路的容量是5Mbps，你希望TCP流和UDP流的表现如何？

观察谁会回到横坐标轴，这代表了他发生了（Congestion Control - 回到 CWND = 1）

问题2：为什么一个流量比另一个流量的吞吐量高？试解释是什么机制迫使这两个流稳定到观察到的吞吐量。

UDP 基于 IP 协议的，非常简单的一个 User Datagram Protocol。best effort（尽力而为的机制）

UDP 不考虑 flow control congestion control（一直在发）

TCP 因为存在 flow control（个人问题：点对点的）congestion control（群体问题：管道链路的），收到速度限制。

问题3：当我们的连接必须与其他流竞争同一链路时（统计多路复用：谁需要的多，分配的带宽更多），请列出使用UDP而不是TCP进行文件传输的优点和缺点。如果大家都因为同样的原因开始使用UDP而不是TCP，会发生什么？

UDP pros cons : Best-effort 的特性

快。保证速度。不可靠。可靠的功能：上层 application layer 来实现。

我们互相都在竞争，但是没有人控制。个人角度看，大家都很融洽，都尽力而为的发送。

但是从链路角度出发，网络是不是会出现 Congestion。如果说一个 Router 的Congestion，会造成其他路由器的拥塞，也就是最后可能导致网络瘫痪。TCP，根据congestion control 越发越少。最后逐渐不发了。

UDP best effort 一直在发。高速的发。存在越来越多的包堆积在一起router。congestion 越来越严重。Queue 越来越满 Router 产生了网络瘫痪。会影响他的上游节点。最后大量的网络瘫痪。