数据通信作业

姓名：刘浩文　　学号：517021911065　　日期：2020/6/3

一、 实验名称及内容

名称： ns3 记录点到点 **TCP** 通信的拥塞窗口变化

内容：

1. 使用 **using SocketObject->TraceConnect(.....)** 方法记录 *mysixth.cc* 拥塞窗口变化
2. 使用 **using Config::Connect(.....)** 方法记录 *tcp-bulk-send.cc* 拥塞窗口变化

```
//
// ============================================================================
//
//         node 0                  node 1
//   +----------------+    +----------------+
//   |    ns-3 TCP    |    |    ns-3 TCP    |
//   +----------------+    +----------------+
//   |    10.1.1.1    |    |    10.1.1.2    |
//   +----------------+    +----------------+
//   | point-to-point |    | point-to-point |
//   +----------------+    +----------------+
//           |                      |
//           +----------------------+
//                5 Mbps, 2 ms
//
//
```

二、实验过程和结果

环境

物理主机系统：*macOS Catalina 10.15.4*

虚拟机系统：*Ubuntu 18.04.4 LTS*

虚拟机软件：*VMware Fusion专业版 11.5.3 (15870345)*

实验软件：*ns-3.28*

实验过程

首先完成了实验的教程部分，阅读了代码及解析，复习了计算机通信网络课程中学习的拥塞窗口知识，了解了如何使用 **ns3** 去追踪拥塞窗口变化。

然后仿照教程，完成对 *mysixth.cc* 和 *tcp-bulk-send.cc* 的拥塞窗口变化。

实验结果

运行程序：

```
test@ubuntu1804:~/workspace/ns-allinone-3.28/ns-3.28$ ./waf --run scratch/lab5
Waf: Entering directory `/home/test/workspace/ns-allinone-3.28/ns-3.28/build'
Waf: Leaving directory `/home/test/workspace/ns-allinone-3.28/ns-3.28/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.159s)
PhyRxEnd at 1.00209
1.00419 536
PhyRxEnd at 1.00627
PhyRxEnd at 1.00722
1.0093  1072
PhyRxEnd at 1.01225
PhyRxEnd at 1.01319
1.01528 1608
PhyRxEnd at 1.01812
PhyRxEnd at 1.01958
```

```
test@ubuntu1804:~/workspace/ns-allinone-3.28/ns-3.28$ ./waf --run scratch/lab6
Waf: Entering directory `/home/test/workspace/ns-allinone-3.28/ns-3.28/build'
Waf: Leaving directory `/home/test/workspace/ns-allinone-3.28/ns-3.28/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.991s)
PhyRxEnd at 0.005928
0.011856        536
PhyRxEnd at 0.01772
PhyRxEnd at 0.02716
0.033024        1072
PhyRxEnd at 0.047464
PhyRxEnd at 0.056904
0.062768        1608
PhyRxEnd at 0.077208
PhyRxEnd at 0.086648
```

由于数据结构不同，所以使用 *python* 脚本处理输出的数据文件，使之后能顺利画出三种图像。

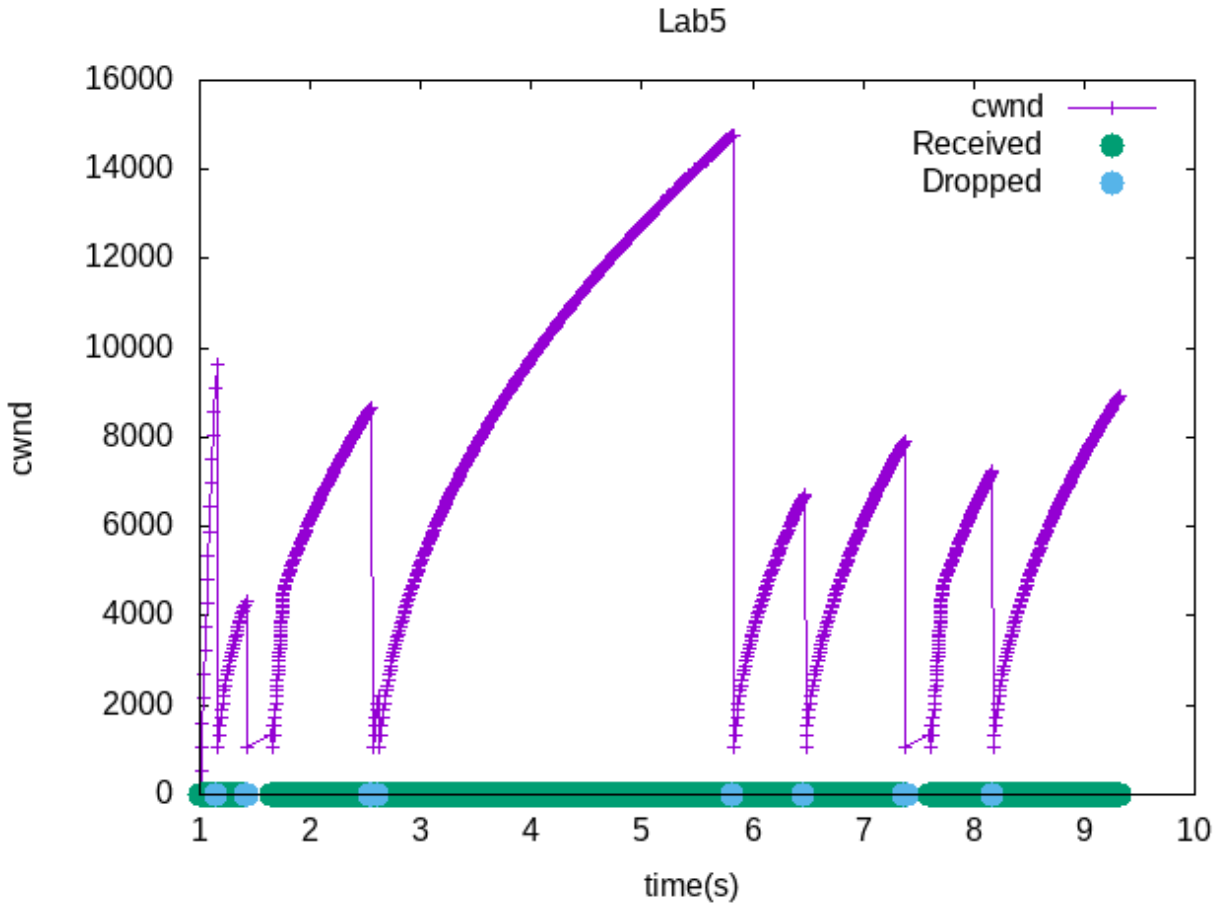
```python
f.open('Lab5_drop.dat', 'r')
str1 = f.read()
f.close()
str1 = str1.replace('\n', '\t0\n')
f.open('lab5_drop.dat', 'w')
f.write(str1)
f.close()
```

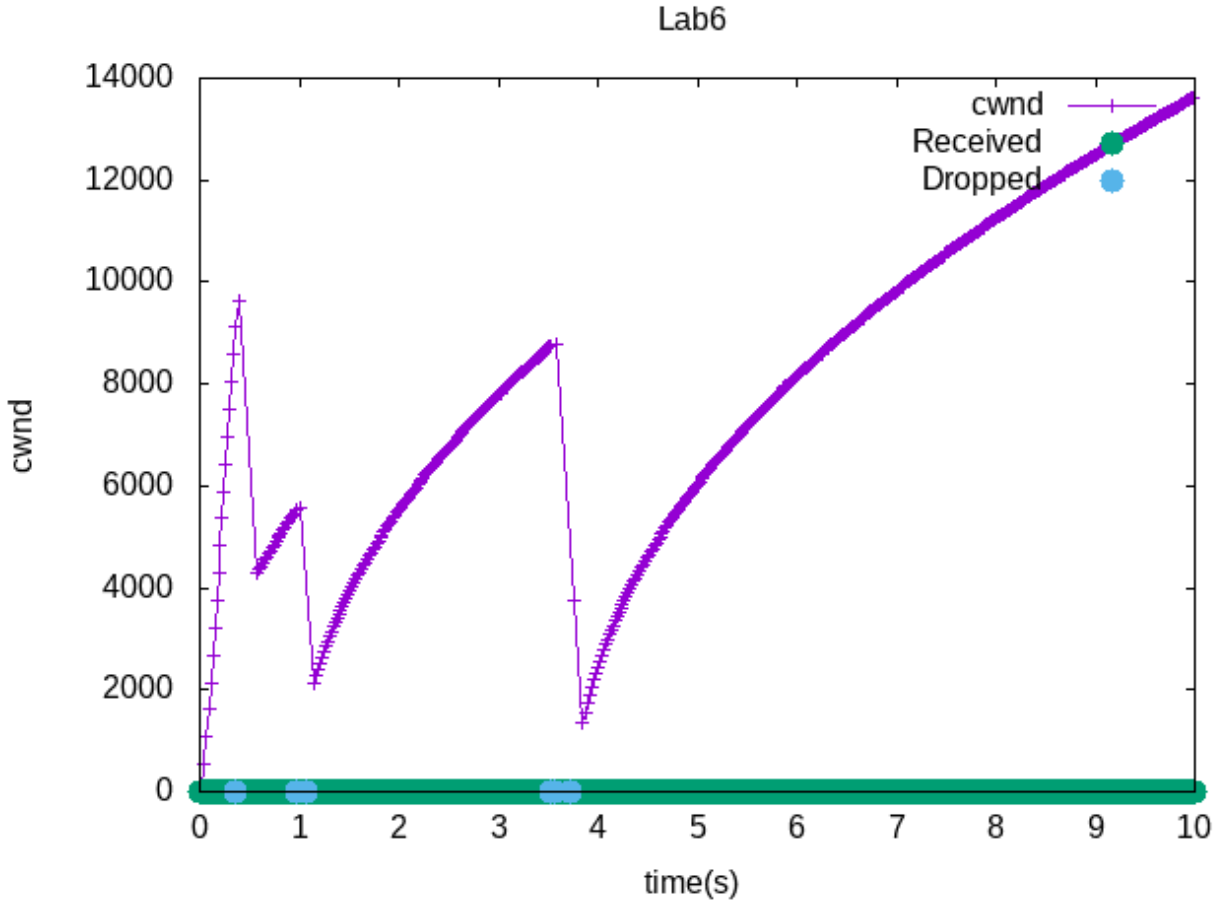
```
f.open('Lab5_recv.dat', 'r')
str1 = f.read()
f.close()
str1 = str1.replace('\n', '\t0\n')
f.open('lab5_recv.dat', 'w')
f.write(str1)
f.close()
f.open('Lab6_drop.dat', 'r')
str1 = f.read()
f.close()
str1 = str1.replace('\n', '\t0\n')
f.open('lab6_drop.dat', 'w')
f.write(str1)
f.close()
f.open('Lab6_recv.dat', 'r')
str1 = f.read()
f.close()
str1 = str1.replace('\n', '\t0\n')
f.open('lab6_recv.dat', 'w')
f.write(str1)
f.close()
```

使用 **gnuplot** 画图：

```
$ gnuplot
gnuplot> set xlabel "time(s)"
gnuplot> set ylabel "cwnd"
gnuplot> set title "Lab5"
gnuplot> set output "lab5.png"
gnuplot> set terminal png size 640,480
gnuplot> plot "Lab5_cwnd.dat" using 1:2 title "cwnd" with linespoints, "lab5_recv.dat" using 1:2 title "Received" with points pointtype 7 pointsize 2, "lab5_drop.dat" using 1:2 title "Dropped" with points pointtype 7 pointsize 2
gnuplot> exit
$ gnuplot
gnuplot> set xlabel "time(s)"
gnuplot> set ylabel "cwnd"
gnuplot> set title "Lab6"
gnuplot> set output "lab6.png"
gnuplot> set terminal png size 640,480
gnuplot> plot "Lab6_cwnd.dat" using 1:2 title "cwnd" with linespoints, "lab6_recv.dat" using 1:2 title "Received" with points pointtype 7 pointsize 2, "lab6_drop.dat" using 1:2 title "Dropped" with points pointtype 7 pointsize 2
gnuplot> exit
```

得到图像：


Lab5
cwnd
Received
Dropped
16000
14000
12000
10000
8000
6000
4000
2000
0
1
2
3
4
5
6
7
8
9
10
cwnd
time(s)


Lab6
cwnd
Received
Dropped
14000
12000
10000
8000
6000
4000
2000
0
0
1
2
3
4
5
6
7
8
9
10
cwnd
time(s)

三、问题与思考

这次实验帮我很好的复习了计算机通信网络中学到的 **TCP** 拥塞窗口机制。当时只是在书本上学习理论知识，这次实际进行模拟，对其理解上了一层楼。