数据通信作业

姓名：刘浩文　　学号：517021911065　　日期：2020/5/27

一、 实验名称及内容

名称： ns3 Tracing 功能使用

内容：使用 **ns3** 的 **Tracing** 功能完成对 **WiFi** 移动节点仿真时位置的追踪。

```
//
//   Wifi 10.1.3.0
//                  AP
//  *    *    *    *
//  |    |    |    |          10.1.1.0
// n5   n6   n7   n0 ------------------------ n1   n2   n3   n4
//                          point-to-point      |    |    |    |
//                                              =================
//                                                LAN 10.1.2.0
//
```

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

首先完成了实验的第一和第二题，阅读了这两题的代码及解析，了解了 **ns3** 强大的 **Tracing** 功能：由追踪源和追踪接收器两个独立的部分组成，追踪源可以向模拟中发生的事件发出信号，并提供对感兴趣的底层数据的访问；追踪接收器接收并记录追踪源提供的信息。

然后仿照第一题和第二题，完成对 *third.cc* 中 **WiFi** 移动节点仿真时位置的追踪：在程序最开始创建追踪接收器函数，在程序末尾附近使用追踪源并生成记录文件。由于第一次实现时只输出了一个记录文件，三个移动节点的数据混在一起，无法使用 **gnuplot** 画出追踪图，因此在最后使用了三次追踪源，调用了三次追踪接收器，对每个节点都形成一个文件流，最后生成三个文件分别记录三个移动节点的位置追踪信息。

实验结果

运行程序：

```
test@ubuntu1804:~/workspace/ns-allinone-3.28/ns-3.28$ ./waf --run scratch/lab4
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 (1.694s)
/NodeList/5/$ns3::MobilityModel/CourseChange
/NodeList/6/$ns3::MobilityModel/CourseChange
/NodeList/7/$ns3::MobilityModel/CourseChange
/NodeList/5/$ns3::MobilityModel/CourseChange x = 0      y = 0
/NodeList/6/$ns3::MobilityModel/CourseChange x = 5      y = 0
/NodeList/7/$ns3::MobilityModel/CourseChange x = 10     y = 0
/NodeList/5/$ns3::MobilityModel/CourseChange x = -0.63917       y = -0.769065
/NodeList/7/$ns3::MobilityModel/CourseChange x = 10.3841        y = 0.923277
/NodeList/6/$ns3::MobilityModel/CourseChange x = 4.23241        y = -0.640938
/NodeList/5/$ns3::MobilityModel/CourseChange x = -0.376539      y = 0.195831
/NodeList/7/$ns3::MobilityModel/CourseChange x = 10.2049        y = 1.90708
/NodeList/6/$ns3::MobilityModel/CourseChange x = 5.21205        y = -0.44018
/NodeList/5/$ns3::MobilityModel/CourseChange x = -0.574674      y = 1.17601
/NodeList/6/$ns3::MobilityModel/CourseChange x = 5.03203        y = 0.543483
/NodeList/7/$ns3::MobilityModel/CourseChange x = 10.8136        y = 1.11368
/NodeList/5/$ns3::MobilityModel/CourseChange x = -1.5146        y = 0.834616
/NodeList/6/$ns3::MobilityModel/CourseChange x = 5.11926        y = -0.452705
/NodeList/7/$ns3::MobilityModel/CourseChange x = 10.8452        y = 2.11318
```

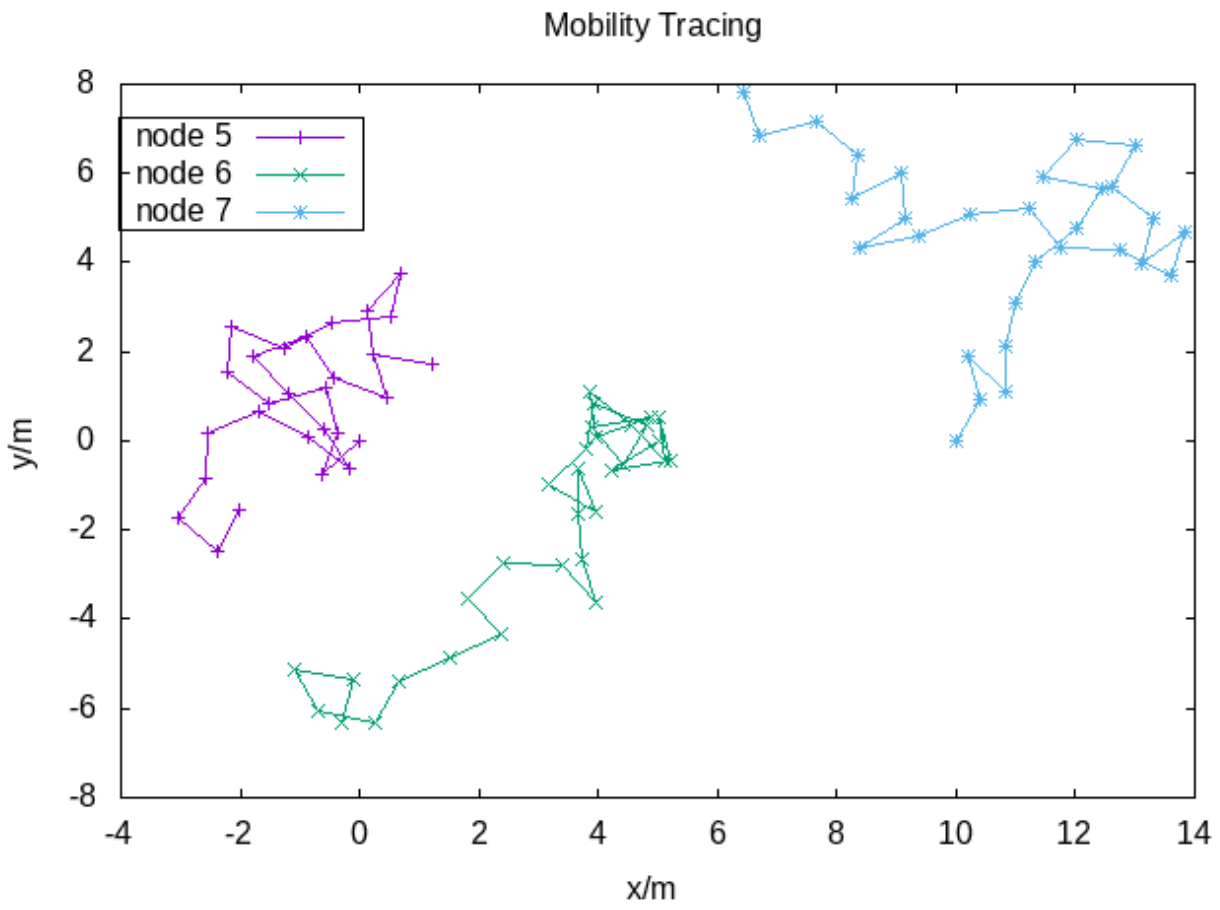
生成三个记录文件：

```
test@ubuntu1804:~/workspace/ns-allinone-3.28/ns-3.28$ ls
AUTHORS        DlMacStats.txt       doc          lab4.png       scratch         UlInterferenceStats.txt
bindings       DlPdcpStats.txt      examples     LICENSE        src             UlMacStats.txt
build          DlRlcStats.txt       lab4-0.dat   Makefile       test.py         UlPdcpStats.txt
CHANGES.html   DlRsrpSinrStats.txt  lab4-1.dat   README         testpy-output   UlRlcStats.txt
contrib        DlTxPhyStats.txt     lab4-2.dat   RELEASE_NOTES  testpy.supp     UlSinrStats.txt
```

使用 **gnuplot** 画图：

```
$ gnuplot
gnuplot> set xlabel "x/m"
gnuplot> set ylabel "y/m"
gnuplot> set title "Mobility Tracing"
gnuplot> set output "lab4.png"
gnuplot> set terminal png size 640,480
gnuplot> set key box
gnuplot> set key center at -2,6
gnuplot> plot "lab4-0.dat" using 2:3 title "node 5" with linespoints,\
>"lab4-1.dat" using 2:3 title "node 6" with linespoints,\
>"lab4-2.dat" using 2:3 title "node 7" with linespoints
gnuplot> exit
```

得到图像：



三、问题与思考

这次实验并没有继续以往的网络通信仿真，而是转而学习 **ns3** 的追踪机制。这种简便快捷的追踪机制确实让我印象深刻，但对它在程序层面的原理还是不清楚。但不管怎么说，这种简便快捷的追踪方法确实大大提高了实验效率，难怪 **ns3** 如此受人欢迎。