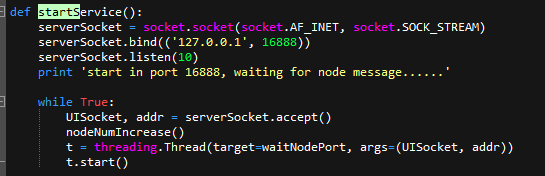
调用tracehandler.py，启动性能分析模块。

首先，调用线程startService，监听16888端口。

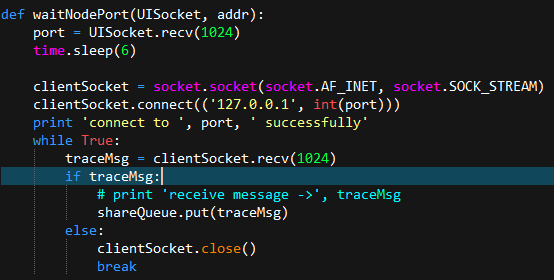




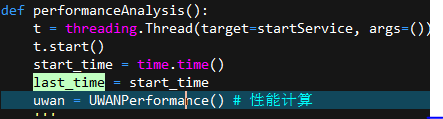
等待testNode.py与这个端口建立TCP连接，然后发送一个port过来（9080等），收到这个端口以后，再与这个port建立TCP连接。（如果成功连接一个节点的协议栈，则nodeNum自加一，作为全局的节点数目。同时调用waitNodePort线程）



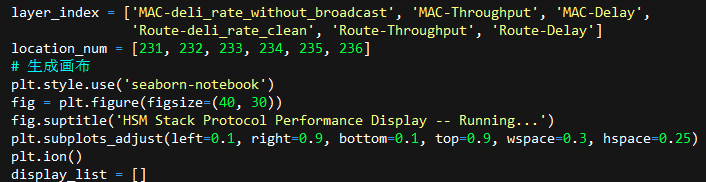
建立成功之后，再创建一个waitNodePort线程，用于给该套接字通信。（即5个节点就建立了5个这样的线程。）收到trace数据就放到shareQueue这个队列中。（这个队列中的数据是从协议栈传来的原始数据，需要拆分。）

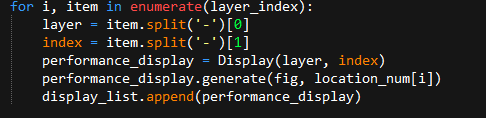


回到主进程中，这里获得一个进行性能计算的对象，利用这个对象对数据进行处理：



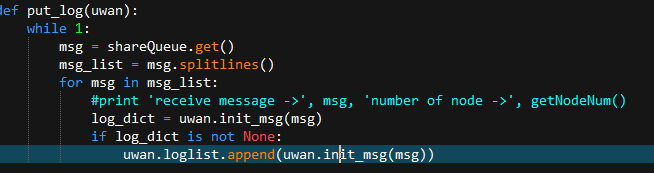
然后是生成画布相关的：



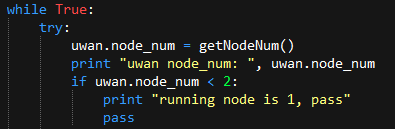


接下来开一个线程，主要是对shareQueue中的原始数据进行处理。首先调用splitlines根据\n隔开数据，然后调用uwan对象中的init\_msg函数，把分开的数据交给这个函数处理。





接下来进入主循环（每10秒循环一次），首先判断有多少个节点参与了，如果少于2个则不进行统计。



如果多于一个节点，先调用uwan的函数performance\_detail进行统计（根据put\_log线程传入的所有数据）。所有的结果放到一个json返回，说明只要根据result的数据进行统计和画图即可。

1 如果当前时间和上次消息的时间差超过100s（说明太长时间没有数据），不添加新的点，只是改标题。

2 否则，检查一下数据是否正确，开始结束时间是否为0。如果不正确，说明还没数据开始发。如果正确，就添加新的点，并且时间设置为当前时间– starttime。

