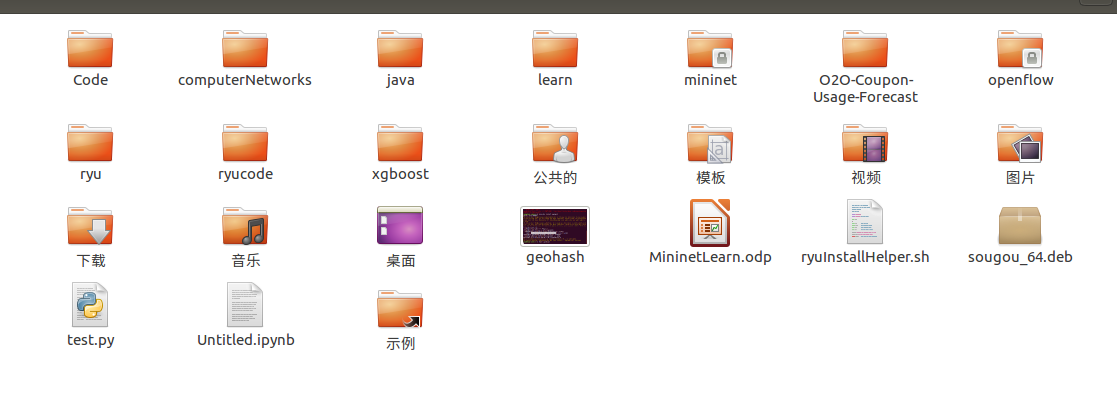
**SDN中动态负载均衡与节能机制的程序文档**

**一、整体介绍**

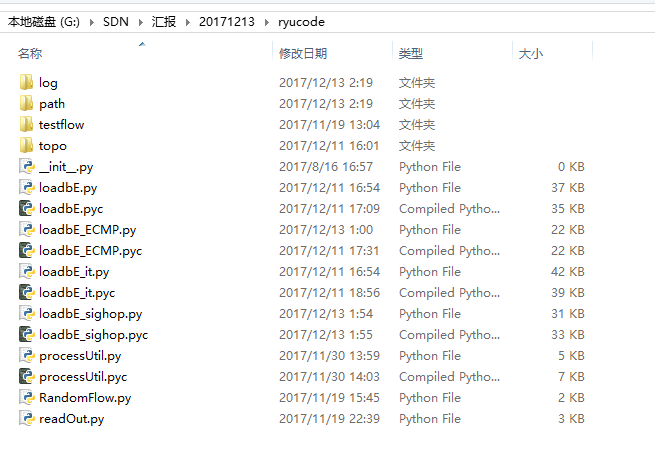
整个程序分为四部分：Ryu部分、Mininet发包部分、拓扑构建部分和 数据处理部分（ryucode mininet learn xnpj在另一个地方)



（1）Ryu部分

Ryu部分位于ryucode文件夹，是整个机制的核心，包含4种算法（LoadbE, LoadbE-it, IECMP和ILABERIO）

文件结构如图所示



1）Log文件夹存放每次算法的输出，文件名是开启ryu的时间戳。

2）path文件夹存放每个数据流FlowMod消息的安装路径、调度路径和FlowRemove消息删除路径。

3）testflow是模拟流量文件夹，修改后的mininet通过调用iperfread 命令读取testflow里的文件，可以通过Iperf自动发包自动化模拟连续数据流的到达。其中testflow里的文件可以通过RandomFlow.py文件生成，也可以修改参数对流模型修改。

4）topo文件夹，存放拓扑配置文件，是为了保证ryu控制器得到和Mininet一样的拓扑，替换了ryu中拓扑发现模块，同时也解决了环路广播风暴问题。

5）LoadbE.py 为LoadbE机制的核心文件

6）LoadbE\_it.py 为LoadbE-it机制的核心文件

7）LoadbE\_ECMP.py 为IECMP机制的核心文件

8）LoadbE\_sighop.py 为ILABERIO机制的核心文件（其中sighop为原LABERIO中的一个算法）

9）readOut.py文件为对log文件进行图形化的程序，显示了6个指标随时间变化曲线

（2）Mininet 文件部分主要包含修改版的Mininet：（添加自定义命令参考SDNLAB，只需要修改如下三个文件，可以先安装Mininet，然后替换里面的三个文件，再重新编译安装）

mininet/bin/mn :命令文件

mininet/mininet/net.py

mininet/mininet/cli.py

（3）拓扑构建部分 使用命令行在Mininet中创建拓扑（learn文件夹）。

learn文件中有三个拓扑构建文件，其中topo-7sw-4-4.py和topo-10sw-8-datacenter.py文件是论文中使用的拓扑1和拓扑2。

（4）性能评价数据处理部分主要是对数据进行可视化

这里面的文件主要是按照指标文件可视化显示，如文件名所示。

将Log文件夹每一对应列提取出来，放到Excel中处理（数据对齐），然后复制到1.txt等文件，再调用1draw.py显示。

**二、机制代码介绍**

（1）主要数据结构

表4.4 主要数据结构与类

Table 4.4 The main data structure and class

|  |  |
| --- | --- |
| 数据结构 | 描述 |
| CoreEdge | 核心链路类，链接两个交换机的链路 |
| Edge | 接入链路类，链接交换机与主机的链路 |
| Host | 主机节点类，存储主机IP等信息 |
| Node | 交换机节点类，存储节点状态信息 |
| Flow | 数据流，存储数据流的相关信息 |
| Checks | 数据流监测点集合 |
| BwTM | 网络链路带宽占用矩阵（CoreEdge中） |
| RwTM | 网络链路剩余带宽矩阵（CoreEdge中） |
| FavorTM | 网络链路偏好矩阵（CoreEdge中） |
| RatioTM | 网络链路利用率矩阵（CoreEdge中） |

在动态负载均衡与节能机制的设计中，本文分别对网络的节点、链路以及数据流进行了建模，以此定义了节点、链路、数据流和主机的类。同时为了存储网络的状态，本文构建了网络的剩余带宽矩阵、链路偏好矩阵、链路利用率矩阵等，具体见表4.4所示。

（2）流量监测机制的主要函数

在流量监测机制中涉及的功能是控制器定时对数据流发送轮询请求，并且在收到响应后，对链路的剩余带宽矩阵进行更新，计算链路偏好矩阵，更新数据流的状态，为数据流设置新的监测点。当收到FlowRemove消息后需要对数据流产生的数据结构进行清理。流量监测机制的主要函数如表4.5所示。

表4.5 流监测机制的主要函数

Table 4.5 The main function of traffic monitoring

|  |  |
| --- | --- |
| 函数名 | 功能描述 |
| \_monitor() | 轮询函数 |
| \_record() | 记录每个时刻网络的能耗、吞吐量等 |
| \_request\_stats() | 发送轮询请求 |
| \_add\_check() | 添加监测点 |
| updateLinksRate() | 更新链路状态 |
| getWeight1() | 计算链路偏好 |
| getNewInterval() | 更新轮询间隔 |
| recordFlows() | 流信息登记 |
| flow\_removed\_handler() | FlowRemove消息的处理 |

（3）流表项的安装与更新

在流表安装和更新阶段，主要涉及流表的增加、删除和修改功能，其主要函数如表4.6所示。

表4.6 流表项的安装与更新的主要函数

Table 4.6 The main function of installing and updating for flow entries

|  |  |
| --- | --- |
| 函数名 | 功能描述 |
| change\_path() | 将原路径修改为新路径 |
| add\_flow() | 安装指定的流表项 |
| del\_path() | 删除原路径上多余的流表项 |
| del\_flow() | 删除指定交换机的指定流表项 |
| modify\_flow() | 修改指定交换机的指定流表项 |

（4）路由机制的主要函数

路由机制中主要包含对PacketIn消息的处理、获取流的头部信息、基于偏好的路径计算和流路径的安装等功能，其主要函数如表4.7所示。

表4.7 路由机制的主要函数

Table 4.7 The main function of routing

|  |  |
| --- | --- |
| 函数名 | 功能描述 |
| \_packet\_in\_handler() | PacketIn消息处理，提取包头信息 |
| receiveARP() | ARP消息单独处理，避免广播风暴 |
| getFirstPath() | 网络中只有一个数据流时的路径计算 |
| getInitMatrix() | 根据拓扑类型构建流量矩阵和偏好矩阵 |
| PbRR() | 基于链路偏好进行路由 |
| dijkstra() | 根据拓扑类型计算最短路径 |
| BFS() | 判断目的节点是否可达 |
| addPathFlow() | 路径安装 |
| getOutport() | 获取与目的主机相连的交换机端口 |

（5）流调度机制的主要函数

表4.8 非迭代流调度机制的主要函数

Table 4.8 The main function of non-iterative flow scheduling mechanism

|  |  |
| --- | --- |
| 函数名 | 功能描述 |
| scheduleFlows() | 调度指定的数据流 |
| getNewPath() | 为数据流计算多条备选路径 |
| getRerouteMatrix() | 根据拓扑类型构建流量矩阵和偏好矩阵 |
| evalue() | 路径评分预处理 |
| score4() | 路径评分 |

表4.9 迭代流调度机制的主要函数

Table 4.9 The main function of iterative flow scheduling mechanism

|  |  |
| --- | --- |
| 函数名 | 功能描述 |
| getNewSchedule\_it () | 迭代地调度数据流直到满足终止条件 |
| getRerouteMatrix\_it () | 根据拓扑类型构建流量矩阵和偏好矩阵 |
| getpaths\_it () | 为调度计算多条备选路径 |
| selectpath\_it () | 从备选路径中选择评分最高的路径 |
| score4\_it() | 路径评分 |

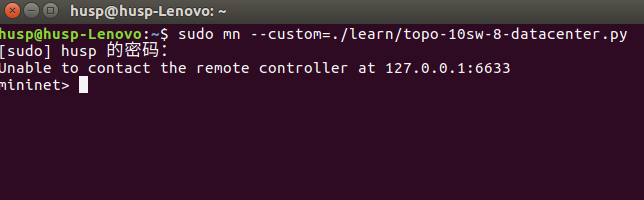
非迭代的流调度的主要函数见表4.8所示，其主要包含数据流的调度、多路径计算和路径评估功能。迭代的流调度的主要函数见表4.9所示，其主要包含迭代式数据流的调度、迭代式多路径计算和路径选择功能。

（注：对比机制中的预计算路径采用的是固定采用字典存储，直接写成了固定的）

**三、运行（如何使用本机制）**

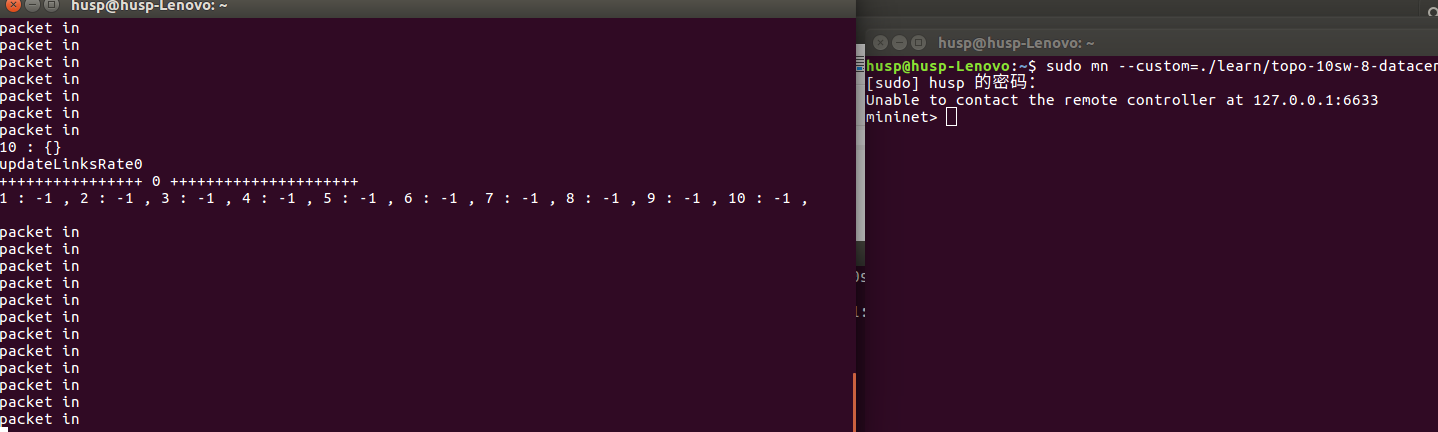
**1）启动Mininet端**

***husp@husp-Lenovo:~$ sudo mn --custom=./learn/topo-10sw-8-datacenter.py***

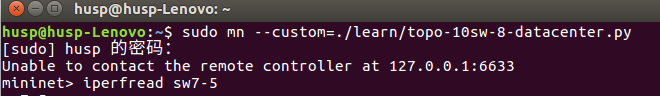
**

**2）启动Ryu端**

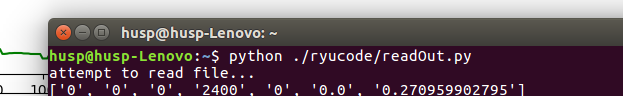
***husp@husp-Lenovo:~$ ryu-manager ./ryucode/loadbE\_sighop.py***

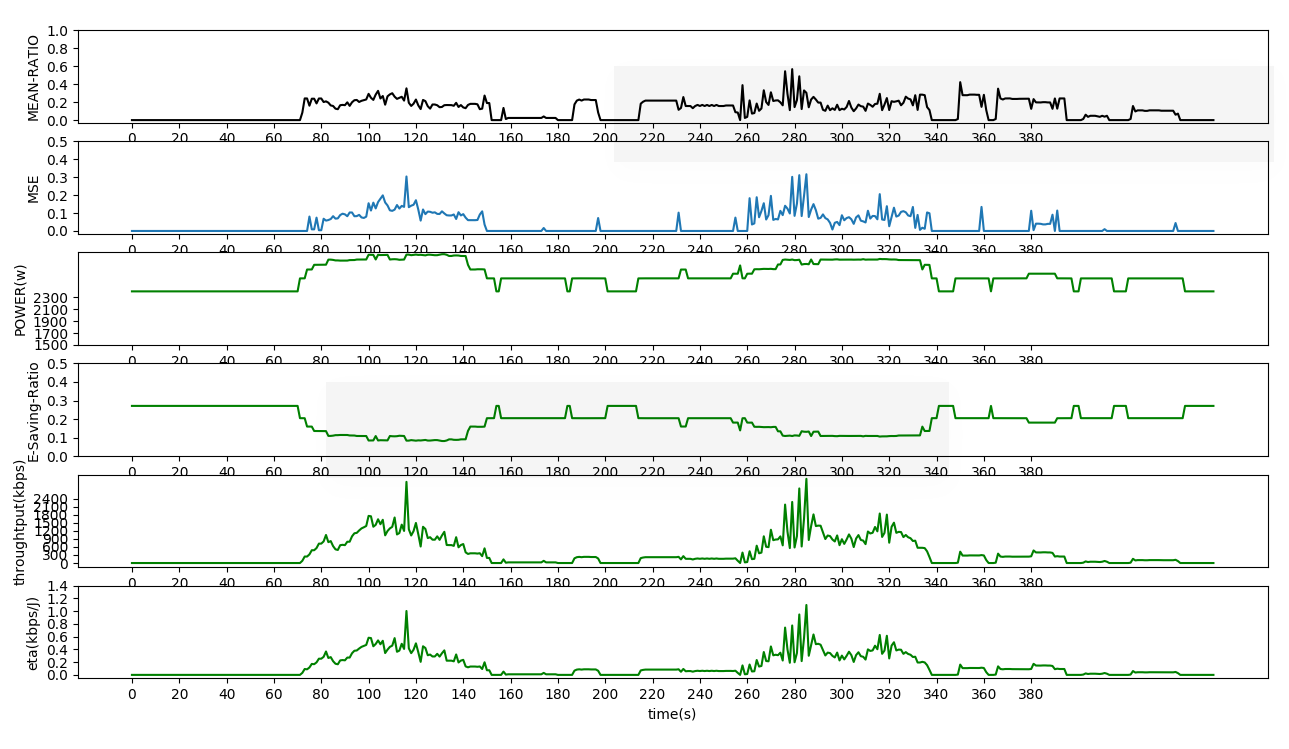
**

**3）开始模拟数据流*mininet> iperfread sw7-5***

**

**4）读取结果并显示*husp@husp-Lenovo:~$ python ./ryucode/readOut.py***

**

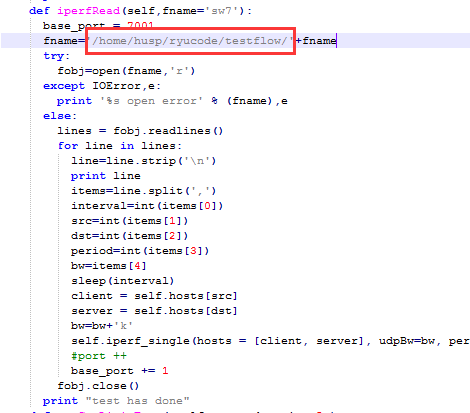
**

**四、修改（如何在本机制的基础上做简单修改）**

（1）修改流输入模型

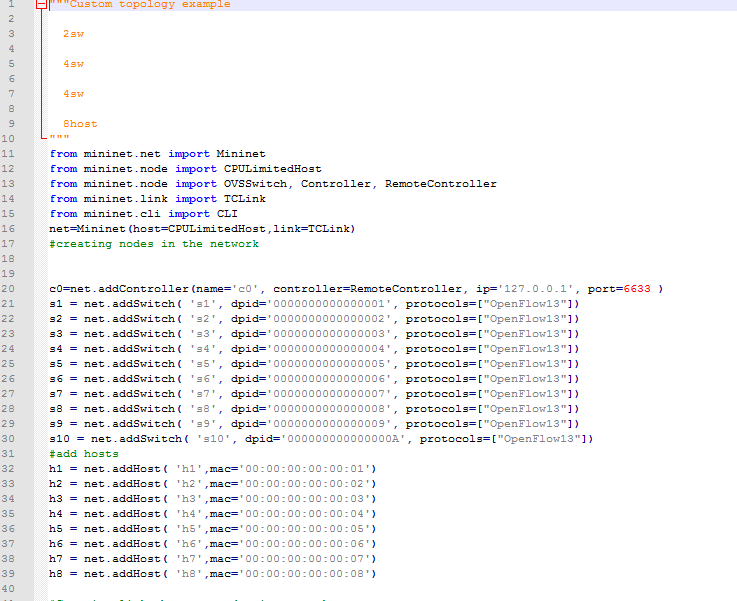
1）使用RandomFlow.py生成数据流文件，或者自己手动设置，生成文件后放到testflow文件夹中。

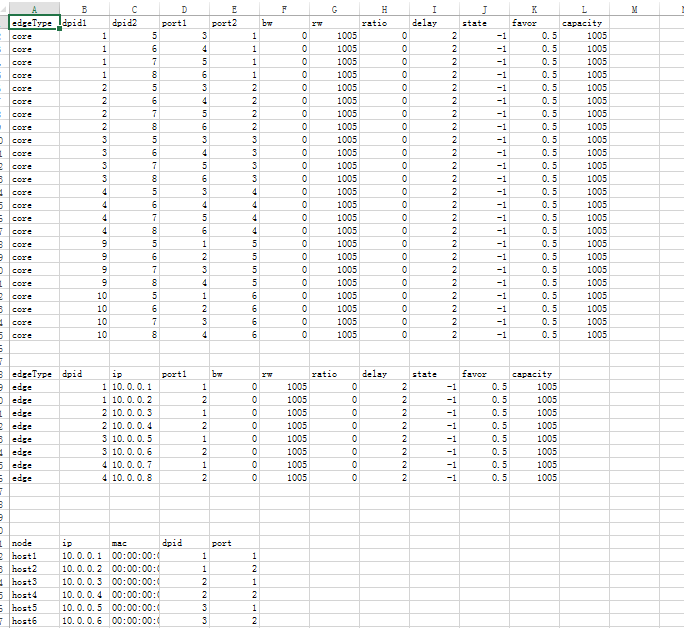
2）修改Mininet中net.py文件，修改相应的读取路径



（2）拓扑替换

1）在learn中配置Mininet拓扑文件，用于Mininet构建拓扑

2）配置好后，topo文件夹配置拓扑文件，用于Ryu读取拓扑



3）修改相应机制中初始化读取路径

self.topoFile='/home/husp/ryucode/topo/topo-10sw-dc.csv'