emulating RIP emulator代码

目录

[1．前端界面代码 2](#_Toc113230912)

[1.1 主窗口代码 2](#_Toc113230913)

[1.2 运行结果窗口代码 31](#_Toc113230914)

[2．后端程序代码 33](#_Toc113230915)

# 1．前端界面代码

## 1.1 主窗口代码

import os

import sys

from PyQt5.QtCore import Qt

from PyQt5.QtWidgets import QApplication, QWidget, QDesktopWidget, QHBoxLayout, QVBoxLayout

from PyQt5.QtWidgets import QPushButton, QLineEdit, QTableWidget, QTableWidgetItem

from PyQt5.QtWidgets import QMessageBox, QMenu

from utils.dialog import ResultDialog

from qt\_material import apply\_stylesheet

import time

BASE\_DIR = os.path.dirname(os.path.realpath(sys.argv[0]))

RUNNING = 1

STOPPING = 2

STOP = 3

class MainWindow(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.switch = STOP

# 控件

self.txt\_asin1 = None

self.txt\_asin2 = None

self.txt\_asin3 = None

self.table\_widget1 = None

self.table\_widget2 = None

# 窗体标题和尺寸

self.setWindowTitle('RIP simulation')

# 窗体的尺寸

self.resize(1000, 450)

# 窗体位置

qr = self.frameGeometry()

cp = QDesktopWidget().availableGeometry().center()

qr.moveCenter(cp)

# 创建布局

layout = QVBoxLayout()

layout.addLayout(self.init\_form())

layout.addLayout(self.init\_table())

layout.addLayout(self.init\_footer())

# 给窗体设置元素的排列方式

self.setLayout(layout)

def init\_form(self):

# 2.创建上面标题布局

form\_layout = QHBoxLayout()

# 2.1 输入框 left

txt\_asin1 = QLineEdit()

# txt\_asin1.setText("N1 3 A")

txt\_asin1.setPlaceholderText("target\_net distance next\_hop, for example, N1 3 A")

self.txt\_asin1 = txt\_asin1

form\_layout.addWidget(txt\_asin1)

# 2.2 添加按钮

btn\_add = QPushButton("ADD")

btn\_add.clicked.connect(self.event\_add\_click1)

form\_layout.addWidget(btn\_add)

# # right

txt\_asin2 = QLineEdit()

txt\_asin2.setPlaceholderText("target\_net distance, for example, N2 4")

self.txt\_asin2 = txt\_asin2

form\_layout.addWidget(txt\_asin2)

# 2.2 添加按钮

btn\_add2 = QPushButton("ADD")

btn\_add2.clicked.connect(self.event\_add\_click2)

form\_layout.addWidget(btn\_add2)

return form\_layout

def init\_table(self):

# 3.创建中间的表格

table\_layout = QHBoxLayout()

# 3.1 创建表格 left

self.table\_widget1 = table\_widget1 = QTableWidget(0, 3)

table\_header = [

{"field": "target\_net", "text": "target net", 'width': 130},

{"field": "distance", "text": "distance", 'width': 130},

{"field": "next\_hop", "text": "next hop", 'width': 130},

]

# 用 table\_header 创建第一列

for idx, info in enumerate(table\_header):

item = QTableWidgetItem()

item.setText(info['text'])

table\_widget1.setHorizontalHeaderItem(idx, item)

table\_widget1.setColumnWidth(idx, info['width'])

# 3.1 创建表格 right

self.table\_widget2 = table\_widget2 = QTableWidget(0, 2)

table\_header = [

{"field": "target\_net", "text": "target net", 'width': 130},

{"field": "distance", "text": "distance", 'width': 130},

# {"field": "metric", "text": "metric", 'width': 150},

]

# 用 table\_header 创建第一列

for idx, info in enumerate(table\_header):

item = QTableWidgetItem()

item.setText(info['text'])

table\_widget2.setHorizontalHeaderItem(idx, item)

table\_widget2.setColumnWidth(idx, info['width'])

# 3.2 初始化表格数据

# 读取数据文件

# left

import json

file\_path = os.path.join(BASE\_DIR, "db", "db1.json")

with open(file\_path, mode='r', encoding='utf-8') as f:

data = f.read()

data\_list1 = json.loads(data)

current\_row\_count1 = table\_widget1.rowCount() # 当前表格有多少行

for row\_list in data\_list1:

table\_widget1.insertRow(current\_row\_count1)

for i, ele in enumerate(row\_list):

cell = QTableWidgetItem(str(ele))

if i in [0, 1, 2]:

# 不可修改

cell.setFlags(Qt.ItemIsSelectable | Qt.ItemIsEnabled)

table\_widget1.setItem(current\_row\_count1, i, cell)

current\_row\_count1 += 1

# # right

file\_path = os.path.join(BASE\_DIR, "db", "db2.json")

with open(file\_path, mode='r', encoding='utf-8') as f:

data = f.read()

data\_list2 = json.loads(data)

current\_row\_count2 = table\_widget2.rowCount() # 当前表格有多少行

for row\_list in data\_list2:

table\_widget2.insertRow(current\_row\_count2)

for i, ele in enumerate(row\_list):

cell = QTableWidgetItem(str(ele))

if i in [0, 1]:

# 不可修改

cell.setFlags(Qt.ItemIsSelectable | Qt.ItemIsEnabled)

table\_widget2.setItem(current\_row\_count2, i, cell)

current\_row\_count2 += 1

table\_widget1.setContextMenuPolicy(Qt.CustomContextMenu)

table\_widget1.customContextMenuRequested.connect(self.table\_right\_menu1)

table\_widget2.setContextMenuPolicy(Qt.CustomContextMenu)

table\_widget2.customContextMenuRequested.connect(self.table\_right\_menu2)

table\_layout.addWidget(table\_widget1)

table\_layout.addWidget(table\_widget2)

return table\_layout

def init\_footer(self):

# 2.底部菜单 left

footer\_layout = QHBoxLayout()

footer\_layout.addStretch()

btn\_sort1 = QPushButton("sort")

btn\_sort1.clicked.connect(self.event\_sort\_click1)

footer\_layout.addWidget(btn\_sort1)

footer\_layout.addStretch()

btn\_reinit1 = QPushButton("reinit")

btn\_reinit1.clicked.connect(self.event\_reinit\_click1)

footer\_layout.addWidget(btn\_reinit1)

footer\_layout.addStretch()

btn\_delete1 = QPushButton("delete")

btn\_delete1.clicked.connect(self.event\_delete\_click1)

footer\_layout.addWidget(btn\_delete1)

footer\_layout.addStretch()

txt\_asin3 = QLineEdit()

txt\_asin3.setPlaceholderText("neighbour route")

self.txt\_asin3 = txt\_asin3

footer\_layout.addWidget(txt\_asin3)

btn\_run = QPushButton("run!")

btn\_run.clicked.connect(self.event\_run\_click)

footer\_layout.addWidget(btn\_run)

footer\_layout.addStretch()

btn\_sort2 = QPushButton("sort")

btn\_sort2.clicked.connect(self.event\_sort\_click2)

footer\_layout.addWidget(btn\_sort2)

footer\_layout.addStretch()

btn\_reinit2 = QPushButton("reinit")

btn\_reinit2.clicked.connect(self.event\_reinit\_click2)

footer\_layout.addWidget(btn\_reinit2)

footer\_layout.addStretch()

btn\_delete2 = QPushButton("delete")

btn\_delete2.clicked.connect(self.event\_delete\_click2)

footer\_layout.addWidget(btn\_delete2)

footer\_layout.addStretch()

return footer\_layout

def event\_add\_click1(self):

# 1.获取输入框中的内容

text = self.txt\_asin1.text()

# print(text)

text = text.strip()

if not text:

QMessageBox.warning(self, "ERROR", "input error!")

return

# # B07YN82X3B=100

self.txt\_asin1.clear()

target\_net, distance, next\_hop = text.split(" ")

# 2.加入到表格中（型号、底价）

new\_row\_list = [target\_net, distance, next\_hop]

current\_row\_count = self.table\_widget1.rowCount() # 当前表格有多少行

self.table\_widget1.insertRow(current\_row\_count)

for i, ele in enumerate(new\_row\_list):

cell = QTableWidgetItem(str(ele))

if i in [0, 1, 2]:

# 不可修改

cell.setFlags(Qt.ItemIsSelectable | Qt.ItemIsEnabled)

self.table\_widget1.setItem(current\_row\_count, i, cell)

# QTableWidget.horizontalHeader().setSortIndicator(0, Qt.AscendingOrder);

pass

def event\_add\_click2(self):

# 1.获取输入框中的内容

text = self.txt\_asin2.text()

# print(text)

text = text.strip()

if not text:

QMessageBox.warning(self, "ERROR", "route information error!")

return

# # B07YN82X3B=100

target\_net, distance = text.split(" ")

self.txt\_asin2.clear()

# 2.加入到表格中（型号、底价）

new\_row\_list = [target\_net, distance]

current\_row\_count = self.table\_widget2.rowCount() # 当前表格有多少行

self.table\_widget2.insertRow(current\_row\_count)

for i, ele in enumerate(new\_row\_list):

cell = QTableWidgetItem(str(ele))

if i in [0, 1, 2]:

# 不可修改

cell.setFlags(Qt.ItemIsSelectable | Qt.ItemIsEnabled)

self.table\_widget2.setItem(current\_row\_count, i, cell)

pass

def event\_reinit\_click1(self):

row\_count = self.table\_widget1.rowCount() # 当前表格有多少行

if not row\_count:

QMessageBox.warning(self, "ERROR", "no available rows!")

return

for rowNum in range(0, row\_count)[::-1]: # 逆序删除，正序删除会有一些删除不成功

self.table\_widget1.removeRow(rowNum)

def event\_reinit\_click2(self):

row\_count = self.table\_widget2.rowCount() # 当前表格有多少行

if not row\_count:

QMessageBox.warning(self, "ERROR", "no available rows!")

return

for rowNum in range(0, row\_count)[::-1]: # 逆序删除，正序删除会有一些删除不成功

self.table\_widget2.removeRow(rowNum)

def event\_delete\_click1(self):

# 1.获取已经选中的行

row\_list = self.table\_widget1.selectionModel().selectedRows()

if not row\_list:

QMessageBox.warning(self, "ERROR", "no available rows!")

return

# 2.翻转

row\_list.reverse()

# 3.删除

for row\_object in row\_list:

index = row\_object.row()

self.table\_widget1.removeRow(index)

def event\_delete\_click2(self):

# 1.获取已经选中的行

row\_list = self.table\_widget2.selectionModel().selectedRows()

if not row\_list:

QMessageBox.warning(self, "ERROR", "no available rows!")

return

# 2.翻转

row\_list.reverse()

# 3.删除

for row\_object in row\_list:

index = row\_object.row()

self.table\_widget2.removeRow(index)

def event\_sort\_click1(self):

self.table\_widget1.sortItems(0, Qt.AscendingOrder)

def event\_sort\_click2(self):

self.table\_widget2.sortItems(0, Qt.AscendingOrder)

def table\_right\_menu1(self, pos):

# 只有选中一行时，才支持右键

selected\_item\_list = self.table\_widget1.selectedItems()

if len(selected\_item\_list) == 0:

return

menu = QMenu()

item\_copy = menu.addAction("copy")

item\_sort = menu.addAction("sort")

item\_delete = menu.addAction("delete")

item\_reinit = menu.addAction("reinit")

item\_run = menu.addAction("run")

# 选中了那个？

action = menu.exec\_(self.table\_widget1.mapToGlobal(pos))

if action == item\_copy:

# 赋值当前型号 B08166SLDF

clipboard = QApplication.clipboard()

clipboard.setText(selected\_item\_list[0].text())

if action == item\_delete:

# if not selected\_item\_list:

# QMessageBox.warning(self, "ERROR", "no available elements!")

# return

# 2.翻转

selected\_item\_list.reverse()

# 3.删除

for row\_object in selected\_item\_list:

index = row\_object.row()

self.table\_widget1.removeRow(index)

if action == item\_reinit:

row\_count = self.table\_widget1.rowCount() # 当前表格有多少行

if not row\_count:

QMessageBox.warning(self, "ERROR", "no available rows!")

return

for rowNum in range(0, row\_count)[::-1]: # 逆序删除，正序删除会有一些删除不成功

self.table\_widget1.removeRow(rowNum)

if action == item\_run:

route = self.txt\_asin3.text()

if not route:

QMessageBox.warning(self, "ERROR", "no available route!")

return

row1 = self.table\_widget1.rowCount()

column1 = self.table\_widget1.columnCount()

if not row1:

QMessageBox.warning(self, "ERROR", "no available route1!")

return

with open('./res/route1.txt', 'wt') as f:

for i in range(0, row1):

for j in range(0, column1):

if j != column1 - 1:

f.write(self.table\_widget1.item(i, j).text() + " ")

else:

f.write(self.table\_widget1.item(i, j).text())

if i != row1 - 1:

f.write("\n")

row2 = self.table\_widget2.rowCount()

column2 = self.table\_widget2.columnCount()

if not row2:

QMessageBox.warning(self, "ERROR", "no available route2!")

return

with open('./res/route2.txt', 'wt') as f:

for i in range(0, row2):

for j in range(0, column2):

if j != column2 - 1:

f.write(self.table\_widget2.item(i, j).text() + " ")

else:

f.write(self.table\_widget2.item(i, j).text())

if i != row2 - 1:

f.write("\n")

with open('./res/route.txt', 'wt') as f:

f.write(route)

os.system("rip.exe")

with open('./res/res.txt', 'rt') as f:

data = f.read()

f.close()

dialog = ResultDialog(data)

dialog.setWindowModality(Qt.ApplicationModal)

dialog.exec\_()

if action == item\_sort:

self.table\_widget1.sortItems(0, Qt.AscendingOrder)

def table\_right\_menu2(self, pos):

# 只有选中一行时，才支持右键

selected\_item\_list = self.table\_widget2.selectedItems()

if len(selected\_item\_list) == 0:

return

menu = QMenu()

item\_copy = menu.addAction("copy")

item\_sort = menu.addAction("sort")

item\_delete = menu.addAction("delete")

item\_reinit = menu.addAction("reinit")

item\_run = menu.addAction("run")

# 选中了那个？

action = menu.exec\_(self.table\_widget2.mapToGlobal(pos))

if action == item\_copy:

# 赋值当前型号 B08166SLDF

clipboard = QApplication.clipboard()

clipboard.setText(selected\_item\_list[0].text())

if action == item\_delete:

# 2.翻转

selected\_item\_list.reverse()

# 3.删除

for row\_object in selected\_item\_list:

index = row\_object.row()

self.table\_widget2.removeRow(index)

if action == item\_reinit:

row\_count = self.table\_widget2.rowCount() # 当前表格有多少行

if not row\_count:

QMessageBox.warning(self, "ERROR", "no available rows!")

return

for rowNum in range(0, row\_count)[::-1]: # 逆序删除，正序删除会有一些删除不成功

self.table\_widget2.removeRow(rowNum)

if action == item\_run:

route = self.txt\_asin3.text()

if not route:

QMessageBox.warning(self, "ERROR", "no available route!")

return

row1 = self.table\_widget1.rowCount()

column1 = self.table\_widget1.columnCount()

if not row1:

QMessageBox.warning(self, "ERROR", "no available route1!")

return

with open('./res/route1.txt', 'wt') as f:

for i in range(0, row1):

for j in range(0, column1):

if j != column1 - 1:

f.write(self.table\_widget1.item(i, j).text() + " ")

else:

f.write(self.table\_widget1.item(i, j).text())

if i != row1 - 1:

f.write("\n")

row2 = self.table\_widget2.rowCount()

column2 = self.table\_widget2.columnCount()

if not row2:

QMessageBox.warning(self, "ERROR", "no available route2!")

return

with open('./res/route2.txt', 'wt') as f:

for i in range(0, row2):

for j in range(0, column2):

if j != column2 - 1:

f.write(self.table\_widget2.item(i, j).text() + " ")

else:

f.write(self.table\_widget2.item(i, j).text())

if i != row2 - 1:

f.write("\n")

with open('./res/route.txt', 'wt') as f:

f.write(route)

os.system("RIP.exe")

with open('./res/res.txt', 'rt') as f:

data = f.read()

f.close()

dialog = ResultDialog(data)

dialog.setWindowModality(Qt.ApplicationModal)

dialog.exec\_()

if action == item\_sort:

self.table\_widget2.sortItems(0, Qt.AscendingOrder)

def event\_run\_click(self):

route = self.txt\_asin3.text()

if not route:

QMessageBox.warning(self, "ERROR", "no available route!")

return

row1 = self.table\_widget1.rowCount()

column1 = self.table\_widget1.columnCount()

if not row1:

QMessageBox.warning(self, "ERROR", "no available route1!")

return

with open('./res/route1.txt', 'wt') as f:

for i in range(0, row1):

for j in range(0, column1):

if j != column1 - 1:

f.write(self.table\_widget1.item(i, j).text() + " ")

else:

f.write(self.table\_widget1.item(i, j).text())

if i != row1 - 1:

f.write("\n")

row2 = self.table\_widget2.rowCount()

column2 = self.table\_widget2.columnCount()

if not row2:

QMessageBox.warning(self, "ERROR", "no available route2!")

return

with open('./res/route2.txt', 'wt') as f:

for i in range(0, row2):

for j in range(0, column2):

if j != column2 - 1:

f.write(self.table\_widget2.item(i, j).text() + " ")

else:

f.write(self.table\_widget2.item(i, j).text())

if i != row2 - 1:

f.write("\n")

with open('./res/route.txt', 'wt') as f:

f.write(route)

os.system("RIP.exe")

with open('./res/res.txt', 'rt') as f:

data = f.read()

f.close()

dialog = ResultDialog(data)

dialog.setWindowModality(Qt.ApplicationModal)

dialog.exec\_()

self.txt\_asin3.clear()

def style():

mm = time.strftime('%m', time.localtime())

m = int(mm)

hh = time.strftime('%H', time.localtime())

h = int(hh)

if 3 <= m <= 5:

season = 1

elif 6 <= m <= 8:

season = 2

elif 9 <= m <= 11:

season = 3

else:

season = 4

if season == 1 or season == 3:

if 6 <= h <= 18:

apply\_stylesheet(app, theme='light\_blue.xml')

else:

apply\_stylesheet(app, theme='dark\_blue.xml')

elif season == 2:

if 5 <= h <= 19:

apply\_stylesheet(app, theme='light\_blue.xml')

else:

apply\_stylesheet(app, theme='dark\_blue.xml')

else:

if 7 <= h <= 17:

apply\_stylesheet(app, theme='light\_blue.xml')

else:

apply\_stylesheet(app, theme='dark\_blue.xml')

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

window = MainWindow()

style()

# apply\_stylesheet(app, theme='dark\_blue.xml')

window.show()

sys.exit(app.exec\_())

## 1.2 运行结果窗口代码

from PyQt5.QtWidgets import QVBoxLayout, QDialog, QTextEdit

class ResultDialog(QDialog):

def \_\_init\_\_(self, data, \*args, \*\*kwargs):

super().\_\_init\_\_(\*args, \*\*kwargs)

self.field\_dict = {}

self.data = data

# print(self.data)

self.init\_ui()

def init\_ui(self):

"""

初始化对话框

:return:

"""

self.setWindowTitle("result")

self.resize(200, 200)

layout = QVBoxLayout()

text\_edit = QTextEdit()

text\_edit.setText("")

layout.addWidget(text\_edit)

self.setLayout(layout)

text\_edit.setText(self.data)

# 2．后端程序代码

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <algorithm>

using namespace std;

//定义路由表结构RTable

typedef struct node{

    char dstNet[5];

    int  distance;

    char nextSkip[5];

}RTable;

RTable RT1[1000];//当前路由器路由表

RTable RT2[1000];//相邻路由器的路由表

int i,l1,l2;

char nearR1[5],nearR2[5];

bool cmp(node a, node b)

{

    return strcmp(a.dstNet, b.dstNet) < 0;

}

//当前路由初始化

void InitRTable( RTable\* RT ){

    // char message

    //读 route1.txt

    FILE \*ft = fopen("./res/route1.txt", "r");

    // fscanf(ft, "%s", message);

    for(i = 0; !feof(ft); ++ i)

        fscanf(ft, "%s%d%s",RT[i].dstNet,&RT[i].distance,RT[i].nextSkip);

    fclose(ft);

    l1 = i;//记录RT1[]的长度

}

//添加路由信息

void AddNearRouter(){

    //读 route.txt

    FILE \*ft = fopen("./res/route.txt", "r");

    fscanf(ft, "%s", nearR1);

}

//相邻路由初始化

void InitNearRTable(){

    //读 route2.txt

    FILE \*ft = fopen("./res/route2.txt", "r");

    // fscanf(ft, "%s %d %s", RT[i].dstNet, &RT[i].distance, RT[i].nextSkip);

    for(i = 0; !feof(ft); ++ i)

    {

        fscanf(ft, "%s%d\n",RT2[i].dstNet, &RT2[i].distance);

    }

    fclose(ft);

    l2 = i;//记录RT2[]的长度

}

//路由信息修改

void UpdateNearRTable ( RTable\* RT2,char\* nearR ){

    int p;

    for( p=0;p<l2;++p ){

    RT2[p].distance = RT2[p].distance + 1;

    strcpy( RT2[p].nextSkip,nearR );

    }

}

//路由表更新

void UpdateRTable( RTable\* RT1,RTable\* RT2 ){

    int p,q;//p——RT2[]，q——RT1[]

    for( p=0;p<l2;++p ){

        int finded=0;

        for( q=0;q<l1;++q ){

            if( strcmp( RT2[p].dstNet,RT1[q].dstNet )==0 ){//当前表中找到与发来的表目的网络相同的一条路由信息

                finded = 1;

                if( strcmp( RT1[q].nextSkip,RT2[q].nextSkip )==0 ){//下一跳路由器正好是这个相邻路由器

                    RT1[q].distance = RT2[p].distance;

                }

                else{//下一跳路由器不是这个

                    if( RT2[p].distance+1<RT1[q].distance ){

                        RT1[q].distance = RT2[p].distance;

                        strcpy( RT1[q].nextSkip,RT2[q].nextSkip );

                    }

                }

            }

        }

        if( !finded ){//当前表中没有这条路由信息，就加上

            strcpy( RT1[l1].dstNet,RT2[p].dstNet );

            RT1[l1].distance = RT2[p].distance;

            strcpy( RT1[l1].nextSkip,RT2[p].nextSkip );

            ++l1;

        }

    }

}

//打印

void PrintRTable( RTable\* RT,int len ){

    FILE \*ft1=fopen("./res/res.txt","wb");

    for( i=0;i<len;++i ){

        fprintf(ft1,"%s %d %s\n", RT[i].dstNet, RT[i].distance, RT[i].nextSkip);

    }

    fclose(ft1);

    FILE \*ft2=fopen("./res/record.txt","ab");

    for( i=0;i<len;++i ){

        fprintf(ft2,"%s %d %s\n", RT[i].dstNet, RT[i].distance, RT[i].nextSkip);

    }

    fprintf(ft2, "\n");

    fclose(ft2);

}

// 打印更新的路由表

void Print\_Update(){

    FILE \*ft=fopen("./res/res.txt","wb");

    UpdateRTable(RT1,RT2);

    sort(RT1, RT1 + l1, cmp);

    PrintRTable(RT1,l1);

    fclose(ft);

}

int main()

{

    //初始化当前路由器

    InitRTable(RT1);

    //添加相邻路由器

    AddNearRouter();

    //初始化相邻路由器

    InitNearRTable();

    //打印修改后的路由信息

    UpdateNearRTable(RT2,nearR1);

    //进行路由表更新

    Print\_Update();

}