**请求区域：**

**1.请求经纬度：**

/geohistorydtn/ areaConnectiveSimulation/CurrentLocationFromSimulator.h

绑定10004端口，向10003端口发送数据，10003端口被Realsimulator绑定

从10004端口获得数据。

发送的数据由4部分组成，ip,经度，纬度，距离。因为是请求当前的经纬度，所有发送的数据中的经纬度距离都为0，表示是请求当前的经纬度。

将从10004端口获得的数据，即当前的经纬度传递给请求区域模块。

**2.请求区域：**

/geohistorydtn/questAreaInfo/ QuestAreaInfo.h

将被传递过来的经纬度传递给63301端口，即传递给MapInterface，判断该经纬度所属哪些区域。

从10004端口获得区域，再传递给请求经纬度模块

**3.将获得的区域信息传递给GeohistoryRouter进行处理**

**日志**

/geohistorydtn/Log/GeohistoryLog.h

使用时，加入该头函数，再初始化一个指针对象，便可利用该指针对象使用。

在dtnd.cc文件中第一次初始化，并也在该文件中关闭。

使用时：GeohistoryLog \*log = GeohistoryLog::GetInstance();

**计时器**

/geohistorydtn/timeManager/TimeManager.h

作用：一分钟倒计时后，查看是否需要执行相应的任务

1.每达到一分钟，执行分钟级的频率向量增加操作

2.每达到一小时，执行小时级的频率向量增加操作和衰减操作

3.Interest包传递后的计时随后处理

**压缩移动表，将移动表按一定时间间隔写入到文件**

使用boost库去序列化移动表中的每一个Area,并存到文件中。两个邻居相遇时交换的也是该文件。

如果系统中没有boost库，则执行

apt-cache search boost

搜到所有的boost库

然后：

sudo apt-get install libboost-all-dev

安装相应的库

即可

**移动表的维护过程：**

在dtnd.cc中初始化AreaManager，并将historyarea.txt中记录的历史移动规律存储到程序中

CurrentLocationFromSimulator将当前时间当前结点所处的三层(多层)区域id和当前时间存储在一个AreaInfo对象中。将该对象存储到一个阻塞队列中，geoHistoryRouter定时进行处理。

geoHistoryRouter接受到AreaInfo对象后，观察是否它所存储的底层区域与原来的底层区域一样，如果一样，说明结点没有离开该底层区域，不进行任何动作。

如果不一样。

首先，检查AreaLayerInfo对象中存储的区域是否已经形成一个Area存储在AreaManager的成员areaMap中。如果没有，生成一个Area和该Area的FrequencyVecter。并将这些FrequencyVecter加入到FrequencyVecterManager中的各个频率队列中，用来执行衰减操作。最后返回底层区域id所在的Area.

其次，根据AreaInfo存储的当前时间，修改当前的底层区域的区域向量(加1操作)，并递归的将父亲区域进行修改。

接下来，将原来的底层区域的频率向量从TimeManager中撤出，并修改他们的频率向量的所有标记位为false，保证下一次可以正常执行加一操作。

然后将当前底层区域的频率向量加入到TimeManager中，用来维护当前区域频率在不同时间段的加一操作。

最后，递归的将原来的父区域删除，新的父区域加入，不过如果父区域与原来底层区域的父区域一样，则说明父区域并没有改变，就不再执行任何动作。

**每隔一个时间点的动作**

对当前区域的频率向量执行加一操作，并将该时段的频率标记改为true,防止一个时间段内修改两次。

对所有的频率向量执行衰减操作

如果时间点循环够一次(比如过了一小时，分钟频率需要重置)，则将所有的分钟频率的所有标记为改为false。

将historyarea.txt规律重新写入

**决定每个Area,Neighbor中频率类型**

编写mydtn.conf文件，如下：

##########################################

#

# Set the frequency vector used for Area and Neighbor table.

#

# frequency vector set type [minute | hour | monafteve | week | month]

#

frequency vector set type minute & hour

#

##########################################

在dtnd.cc中执行HandleMydtnConfig::handleMydtnConfig();

如果没有该文件，或文件格式出错的话，则默认类型为minute和hour

**修改**

**dtnd.cc main()**

GeohistoryLog \*geohistoryLog=GeohistoryLog::*GetInstance*();

HandleMydtnConfig \*myconfig=HandleMydtnConfig::*Getinstance*();

AreaManager \*areamanager= AreaManager::Getinstance();

areamanager->init();//用于将原来的历史移动规律读入该程序

CurrentLocationFromSimulator \*location=CurrentLocationFromSimulator::Getinstance();

location->start();

TimeManager \*timemanager=TimeManager::GetInstance();

timemanager->start();

**BundleDaemon.h**

public:

//添加 by gaorui

BundleRouter\* getrouter()

{

return router\_;

}

static BundleDaemon \*GetInstance();

**BundleDaemon.cc**

1.//添加 by gaorui

BundleDaemon \* BundleDaemon::GetInstance()

{

ASSERT(instance\_ != NULL);

return instance\_;

}

//end by gaorui

2.run()

BundleRouter::startThread(router\_);

**BundleRouter.cc**

1. void BundleRouter::startThread(BundleRouter \*router)

{

GeoHistoryRouter \*geoHistoryRouter=dynamic\_cast<GeoHistoryRouter \*>(router);

geoHistoryRouter->start();

}

2. BundleRouter::create\_router(const char\* type)

else if (!strcmp(type, "geohistory")) {

return new GeoHistoryRouter();

}