—————————————————————————————

**修改**

修改：DTN2/servlib/bundling/BPQBlock

#include<configure>

Daemon.cc

handle\_bundle\_delivered注释掉

Bundle\* bundle = event->bundleref\_.object();

log\_info("BUNDLE\_DELIVERED id:%d (%zu bytes) -> regid %d (%s)",

DTN2/servlib/routing/TableBasedRouter

should\_fwd 注释掉，修改

"skip %s since bundle arrived from the same node",

bundle->bundleid(), route->link()->name());

return false;

}

}

return BundleRouter::should\_fwd(bundle, route->link(), route->action());

修改;DTN2/oasys/include/oasys/thread/LockDebugger

添加: #include"dtn-config.h"

修改;DTN2/oasys/include/oasys/thread/TLS

添加: #include<oasys/include/oasys/oasys-config.h>

修改;DTN2/oasys/include/oasys/thread.h

注释//#if OASYS\_DEBUG\_LOCKING\_ENABLED

// Locking debugging thread - local to each thread

**static** TLS<LockDebugger> *lock\_debugger\_*;

//#endif

修改后的程序运行到打开所有线程时，一共有6个线程

TCPServer，BundleDaemon，IPDiscovery，UDPConvergenceLayer

TCPServer，dtn::main

—————————————————————————————

dtnd.cc

初始化类 DTNStorageConfig/storage\_config\_

该类存储db的路径，payload路径

初始化DTNService

开始DTNService

开始DTNService:

开始Daemon

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**怎么在dtn加入线程?**

1.在需要成为线程的.h文件中加入#include <oasys/thread/Thread.h>

2.该类继承类thread

如:class CurrentLocationFromSimulator :public Thread

3.在构造函数中加入thread("线程名字"，CREATE\_JOINABLE)

如： CurrentLocationFromSimulator()

: Thread("Location", CREATE\_JOINABLE)；

4.重载void run()函数

5.打开该线程时，只需要在只需要在相应位置上声明一个对象，然后执行对象->start()就可以。当主程序执行oasys::Thread::release\_start\_barrier();这一语句后，就可打来线程

如:

CurrentLocationFromSimulator

\*c=CurrentLocationFromSimulator::Getinstance();

c->start();

oasys::Thread::release\_start\_barrier();

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**router:**

1.daemon成员router,在函数run()中初始化

router\_ = BundleRouter::create\_router(BundleRouter::config\_.type\_.c\_str());

2.形参BundleRouter::config\_.type\_.c\_str()决定router的类型，需要修改/etc/dtn.config

3.添加BundleRouter::create\_router

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

return new GeoHistoryRouter();

}

——————————————————————————————————

**配置文件**

/etc/dtn.config

在dtnd.cc文件中执行DTNServer::parse\_conf\_file,修改config\_

——————————————————————————————————

**如何序列化存储读取一个类？**

#include <boost/serialization/vector.hpp>

#include <boost/archive/text\_oarchive.hpp>

#include <boost/archive/text\_iarchive.hpp>

class Area{

private:

friend class boost::serialization::access;

template<class Archive>

void serialize(Archive & ar, const unsigned int version) {

ar & isCurrent;//序列化或反序列化&操作符比>>和<<更为方便

ar & level;

ar & id;

ar & vectorlist;

ar & fatherArea;

}

.....

}

main(){

Area fa1(1,2);

Area a1(3,4,&fa1);

std::fstream ofs;

ofs.open("./area.txt",ios::out);

boost::archive::text\_oarchive oa(ofs);

oa << a1;

ofs.close();

std::fstream ifs;

ifs.open("./area.txt",ios::in);

boost::archive::text\_iarchive ia(ifs);

Area a;

try {

while(!ifs.eof()) {

ia >>a;

cout<<a.level<<" "<<a.id<<endl;

}

}

catch(boost::archive::archive\_exception &e)

{ }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**GeohistoryRouter**

Geohistoryrouter没有单例模式，我们调用的对象都是Daemon中的成员变量router\_(BundleRouter)

1.router\_初始化是在Daeom run()中，

2.打开线程是在初始化后进行的.

————————————————————————————————

**加入链接库？**

在Rules.make中加入需要的链接库**-lboost\_serialization**

EXTRA\_CFLAGS = **-**lb**oost\_serialization**

EXTRA\_CXXFLAGS = **-**lb**oost\_serialization**

EXTRA\_LDFLAGS = **-**lb**oost\_serialization**

————————————————————————

**Area管理**

整个程序的Area都由AreaManager中的hash类型的成员areaMap来保存，其余的都是临时变量，不可对这些临时变量进行修改，因为没有用。

areaMap中的Area中的frequencyvector是真正的。TimeManager，AreaManager中保存的都是它的地址。

\*因此Area中的frequencyvector不能用vector类型，以免发生地址变化,改用list.

\*Area中的father，因为hash\_map的地址可能会改变，因此father不存储指针，而是存储id