

****

**J I A N G S U U N I V E R S I T Y**

**“无线传感网与识别技术”实验报告**

|  |  |
| --- | --- |
| 学院名称： | 计算机科学与通信工程学院 |
| 专业班级： | 物联网工程18级 |
| 学生姓名： | 张承楷 |
| 学生学号： | 3180611023 |
| 指导教师： | 熊书明 |

**2020年7月22日**

# 模拟AODV协议

## 一、功能介绍

 AODV是一种按需路由协议。当一个节点需要给网络中的其他节点传送信息时，如果没有到达目标节点的路由，则必须先以多播的形式发出RREQ(路由请求)报文。RREQ报文中记录着发起节点和目标节点的网络层地址，邻近节点收到RREQ，首先判断目标节点是否为自己。如果是，则向发起节点发送RREP;如果不是，则首先在路由表中查找是否有到达目标节点的路由，如果有，则向源节点单播RREP，否则继续转发RREQ进行查找。

## 二、运行步骤分析

/\* -\*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -\*- \*/

/\*

 \* Copyright (c) 2009 IITP RAS

 \*

 \* This program is free software; you can redistribute it and/or modify

 \* it under the terms of the GNU General Public License version 2 as

 \* published by the Free Software Foundation;

 \*

 \* This program is distributed in the hope that it will be useful,

 \* but WITHOUT ANY WARRANTY; without even the implied warranty of

 \* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the

 \* GNU General Public License for more details.

 \*

 \* You should have received a copy of the GNU General Public License

 \* along with this program; if not, write to the Free Software

 \* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA

 \*

 \* This is an example script for AODV manet routing protocol.

 \*

 \* Authors: Pavel Boyko <boyko@iitp.ru>

 \*/

#include "ns3/aodv-module.h"

#include "ns3/core-module.h"

#include "ns3/network-module.h"

#include "ns3/internet-module.h"

#include "ns3/mobility-module.h"

#include "ns3/point-to-point-module.h"

#include "ns3/wifi-module.h"

#include "ns3/v4ping-helper.h"

#include <iostream>

#include <cmath>

using namespace ns3;

/\*\*

 \* \ingroup aodv-examples

 \* \ingroup examples

 \* \brief Test script.

 \*

 \* This script creates 1-dimensional grid topology and then ping last node from the first one:

 \*

 \* [10.0.0.1] <-- step --> [10.0.0.2] <-- step --> [10.0.0.3] <-- step --> [10.0.0.4]

 \*

 \* ping 10.0.0.4

 \*/

class AodvExample

{

public:

  //声明函数，公有

  AodvExample ();   //构造函数

  /\*\*

   \* \brief Configure script parameters

   \* \param argc is the command line argument count

   \* \param argv is the command line arguments

   \* \return true on successful configuration

  \*/

  bool Configure (int argc, char \*\*argv);   //配置函数

  /// Run simulation

  void Run ();    //运行AODV

  /\*\*

   \* Report results

   \* \param os the output stream

   \*/

  void Report (std::ostream & os);    //打印报告

private:

  //私有变量

  // parameters

  /// Number of nodes

  uint32\_t size;    //节点的个数

  /// Distance between nodes, meters

  double step;    //节点间距

  /// Simulation time, seconds

  double totalTime;   //模拟总时间

  /// Write per-device PCAP traces if true

  bool pcap;    //是否输出pcap文件

  /// Print routes if true

  bool printRoutes;   //是否打印路由表

  // network

  /// nodes used in the example

  NodeContainer nodes;    //节点

  /// devices used in the example

  NetDeviceContainer devices;   //设备

  /// interfaces used in the example

  Ipv4InterfaceContainer interfaces;    //分配ip

private:

  //私有函数

  /// Create the nodes

  void CreateNodes ();   //生成节点

  /// Create the devices

  void CreateDevices ();    //生成设备

  /// Create the network

  void InstallInternetStack ();   //安装栈

  /// Create the simulation applications

  void InstallApplications ();    //安装应用

};

int main (int argc, char \*\*argv)

{

  AodvExample test;   //创建test对象

  test.Configure (argc, argv)

  if (!test.Configure (argc, argv))

    NS\_FATAL\_ERROR ("Configuration failed. Aborted.");    //报错

  test.Run ();    //运行

  test.Report (std::cout);    //打印报告

  return 0;

}

//-----------------------------------------------------------------------------

AodvExample::AodvExample () :

  size (10),    //10个节点

  step (10),    //距离10米

  totalTime (10),   //总时长10s

  pcap (true),    //打印pcap

  printRoutes (true)    //打印路由表

{

}

//在控制台上打印信息

bool

AodvExample::Configure (int argc, char \*\*argv)

{

  SeedManager::SetSeed (12345);

  CommandLine cmd;

  cmd.AddValue ("pcap", "Write PCAP traces.", pcap);

  cmd.AddValue ("printRoutes", "Print routing table dumps.", printRoutes);

  cmd.AddValue ("size", "Number of nodes.", size);

  cmd.AddValue ("time", "Simulation time, s.", totalTime);

  cmd.AddValue ("step", "Grid step, m", step);

  cmd.Parse (argc, argv);

  return true;

}

//运行AODV，分别调用需要的函数

void

AodvExample::Run ()

{

  CreateNodes ();

  CreateDevices ();

  InstallInternetStack ();

  InstallApplications ();

  std::cout << "Starting simulation for " << totalTime << " s ...\n";

  Simulator::Stop (Seconds (totalTime));

  Simulator::Run ();

  Simulator::Destroy ();

}

void

AodvExample::Report (std::ostream &)

{

}

//创建节点，打印信息后创建结点，并给节点设置位置信息，布局等等信息，再安装节点

void

AodvExample::CreateNodes ()

{

  std::cout << "Creating " << (unsigned)size << " nodes " << step << " m apart.\n";   //输出创建节点信息

  nodes.Create (size);

  // Name nodes

  for (uint32\_t i = 0; i < size; ++i)   //给节点命名

    {

      std::ostringstream os;

      os << "node-" << i;

      Names::Add (os.str (), nodes.Get (i));

    }

  // Create static grid

  //设置节点的位置，布局信息

  MobilityHelper mobility;

  mobility.SetPositionAllocator ("ns3::GridPositionAllocator",

                                 "MinX", DoubleValue (0.0),

                                 "MinY", DoubleValue (0.0),

                                 "DeltaX", DoubleValue (step),

                                 "DeltaY", DoubleValue (0),

                                 "GridWidth", UintegerValue (size),

                                 "LayoutType", StringValue ("RowFirst"));

  mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");

  mobility.Install (nodes);

}

//创建WiFi设备，通过物理助手与信道助手创建，安装节点，打印信息

void

AodvExample::CreateDevices ()

{

  //帮助创建WiFi的Mac及信道

  WifiMacHelper wifiMac;

  wifiMac.SetType ("ns3::AdhocWifiMac");

  YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();

  YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();

  wifiPhy.SetChannel (wifiChannel.Create ());

  WifiHelper wifi;

  wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager", "DataMode", StringValue ("OfdmRate6Mbps"), "RtsCtsThreshold", UintegerValue (0));

  devices = wifi.Install (wifiPhy, wifiMac, nodes);

  if (pcap)

    {

      wifiPhy.EnablePcapAll (std::string ("aodv"));

    }

}

//安装设备所需要的栈支持，如AODV协议，设置路由表，安装节点，设置ip地址

void

AodvExample::InstallInternetStack ()

{

  AodvHelper aodv;    //AODV协议

  // you can configure AODV attributes here using aodv.Set(name, value)

  InternetStackHelper stack;    //帮助安装栈的助手

  stack.SetRoutingHelper (aodv); // has effect on the next Install ()

  stack.Install (nodes);

  Ipv4AddressHelper address;

  address.SetBase ("10.0.0.0", "255.0.0.0");

  interfaces = address.Assign (devices);

  if (printRoutes)

    {

      Ptr<OutputStreamWrapper> routingStream = Create<OutputStreamWrapper> ("aodv.routes", std::ios::out);

      aodv.PrintRoutingTableAllAt (Seconds (8), routingStream);

    }

}

//安装应用，通过ping助手测试相邻节点之间连通性，再将节点随移动

void

AodvExample::InstallApplications ()

{

  //测试连通性

  V4PingHelper ping (interfaces.GetAddress (size - 1));

  ping.SetAttribute ("Verbose", BooleanValue (true));

  ApplicationContainer p = ping.Install (nodes.Get (0));

  p.Start (Seconds (0));

  p.Stop (Seconds (totalTime) - Seconds (0.001));

  // 随机移动节点

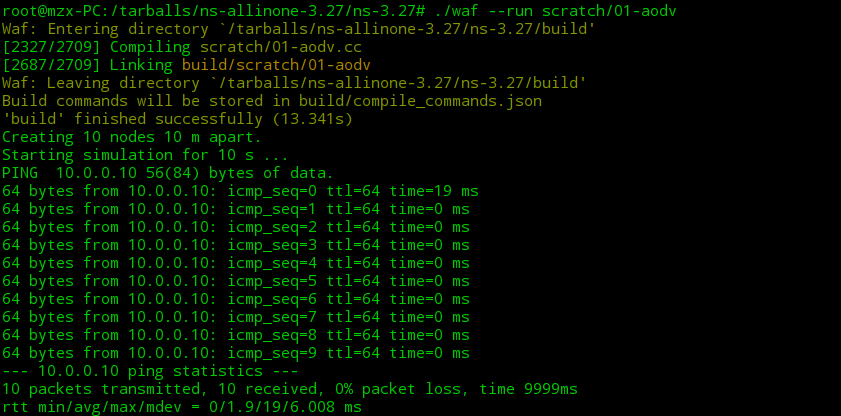
  Ptr<Node> node = nodes.Get (size/2);

  Ptr<MobilityModel> mob = node->GetObject<MobilityModel> ();

  Simulator::Schedule (Seconds (totalTime/3), &MobilityModel::SetPosition, mob, Vector (1e5, 1e5, 1e5));

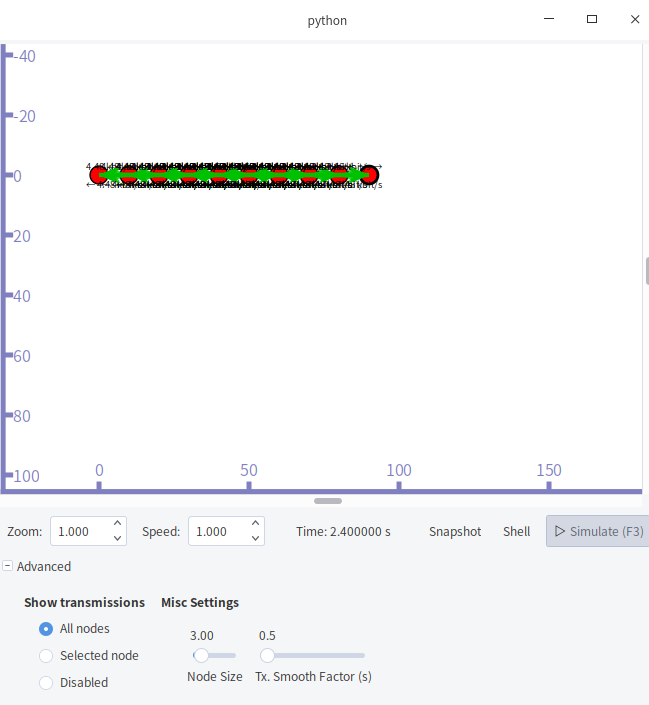
}

## 结果分析



AODV首先创建了10个节点，运行时间为10s,再由10.0.0.1向10.0.0.10

发送ping请求。AODV运行过程首先发RREQ请求帧，在通过路由发现算法构建路由表，因此ping信号通过，找到并成功发送消息。



通过模拟的图像，也可看到节点间相互通信，发送帧形成路由表。