ns3网络构建

1. 预处理:

牢记，不同的设备之间不能有多个通信信道（点对点，有线csma，无线wifi），不同子网之间通信需要有AP（路由器）。

1.1 头文件准备

```
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/ssid.h"
#include "ns3/netanim-module.h"
#include "ns3/ipv4-global-routing-helper.h"

using namespace ns3;

//定义了一个日志组件，使得 ns-3 中的日志系统能够识别并输出该组件相关的日志信息
//名称可随意更改，一般为当前脚本的名称
NS_LOG_COMPONENT_DEFINE ("myExample");

int
main(int argc, char *argv[])
{
}
```

1.2 创建节点容器

```
NodeContainer nodes;
nodes.Create(2);
//设置网络设备助手
NetDeviceContainer device;
```

2. 设置网络模式

2.1 点对点模式

```
//设置点对点助手
PointToPointHelper pointtopoint;
//设置数据传输率
pointtopoint.SetDeviceAttribute("DataRate",StringValue("5Mbps"));
//设置时延
pointtopoint.SetChannelAttribute("Delay",StringValue("2ms"));
device=pointtopoint.Install(nodes);
```

2.2 有线模式

```
//设置有线助手
CsmaHelper csma;
//设置数据传输率
csma.SetChannelAttribute("DataRate",StringValue("100Mbps"));
//设置时延为6560纳秒，即6.56微秒
csma.SetChannelAttribute("Delay",TimeValue(NanoSeconds(6560)));
device=csma.Install(nodes);
```

2.3 无线模式（若一个节点同时配置了wifi和自组网，那么应另设置新的channel、phy和wifi）

```
//设置wifi信道助手,信道包含的各个参数值均为初始值default
YansWifiChannelHelper channel = YansWifiChannelHelper::Default();
//YansWifiPhyHelper设置信道物理层属性
YansWifiPhyHelper phy;

//设置信号发射功率,当AP离设备较远时应适当增大发射功率
//发射功率下限
phy.Set("TxPowerStart", DoubleValue(50.0));
//发射功率上限
phy.Set("TxPowerEnd", DoubleValue(50.0));
//明确通信范围
//channel.AddPropagationLoss("ns3::RangePropagationLossModel", "MaxRange", DoubleValue(1000.0));
//将配置好的信道与物理层关联起来
phy.SetChannel(channel.Create());
```

```
15
16
17  //配置mac层和SSID
18  WifiMacHelper mac;
19  //创建一个SSID，用于标识无线网络的名称，这里设置为Ulraman Tiga
20  Ssid ssid = Ssid("Ulraman Tiga");
21
22
23  //设置wifi助手
24  WifiHelper wifi;
25
26
27  //Ad-hoc（自组网）模式下，每个节点都可以充当中继节点，直接转发其他节点的数据，因此不需要
    单独的路由器设备
28  //mac.SetType("ns3::AdhocWifiMac");//适用于自组网
29  //配置并安装STA设备
30  //设置MAC层为StaWifiMac类型，即Wi-Fi终端MAC，并配置了SSID和ActiveProbing属性
31  //设置为false表示禁用主动探测
32  mac.SetType("ns3::StaWifiMac", "Ssid", SsidValue(ssid), "ActiveProbing",
    BooleanValue(false));
33  /*
34  如果安装MAC层为ApWifiMac类型，则为接入点（AP）
35  mac.SetType("ns3::ApWifiMac", "Ssid", SsidValue(ssid));
36  */
37  //将前面设置好的phy、mac属性安装到节点容器nodes中，并返回到设备容器device中
38  device = wifi.Install(phy, mac,nodes);
```

3. 设置移动模型

3.1 设置移动模型

```
1   //初始化移动助手
2   MobilityHelper mobility;
3   //配置位置分配器
4   //"ns3::GridPositionAllocator": 使用网格位置分配器
5   //MinX,MinY表示网格的起始位置为(0,0)
6   //DeltaX,DeltaY表示节点在网格中相邻位置的水平和垂直间距
7   //GridWidth表示每行的节点个数
8   //LayoutType表示排版类型，RowFirst表示行优先
9   mobility.SetPositionAllocator("ns3::GridPositionAllocator",
10                                "MinX", DoubleValue(0.0),
11                                "MinY", DoubleValue(0.0),
12                                "DeltaX", DoubleValue(5.0),
13                                "DeltaY", DoubleValue(10.0),
```

```
                                "GridWidth", UintegerValue(3),
                                "LayoutType", StringValue("RowFirst"));
//设置节点的移动模型
//设置节点二维随机漫步模型，移动边界为x轴和y轴上(-50,50)
mobility.SetMobilityModel("ns3::RandomWalk2dMobilityModel",
                          "Bounds", RectangleValue(Rectangle(-50, 50, -50,
50)));
//将模型安装到节点容器中
mobility.Install(nodes);

//设置静态移动模型
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(wifiApNode);
```

4. 安装协议与通信设备

4.1 设置网络栈

```
//初始化网络栈助手
InternetStackHelper stack;
//将助手安装到节点上
stack.Install(nodes);
```

4.2 分配ip地址

```
//设置ipv4地址助手
Ipv4AddressHelper address;
//设置子网地址与子网掩码,助手会从192.168.1.1开始给节点分配地址
address.SetBase("192.168.1.0", "255.255.255.0");
//设置子网容器
Ipv4InterfaceContainer Interfaces;
//分配地址，并用子网容器记录节点和对应ipv4地址
Interfaces = address.Assign(Device);
Interfaces.Add(address.Assign (apDevices));//给路由器网络设备分配地址

//全局路由
Ipv4GlobalRoutingHelper::PopulateRoutingTables();


/*
如何动态分配IP地址范围
```

```
address.SetBase("10.0.0.0","255.255.255.252");
address.Assign(nodes);
address.NewNetwork();//切换下一子网
//例如当前address分配IP范围为10.0.0.0-10.0.0.3(即10.0.0.1和10.0.0.2),执行该代码后
//切换到10.0.0.4-10.0.0.7(即10.0.0.5和10.0.0.6)
*/
```

4.3 安装udp客户机/服务器

```
//设置udp服务器助手，收发端口为9
UdpEchoServerHelper echoServer(9);
//将服务器安装到通信双方其中之一上
ApplicationContainer serverApps = echoServer.Install(nodes.Get(0));
//设置服务器的运行开始与结束的时间
serverApps.Start(Seconds(1.0));
serverApps.Stop(Seconds(10.0));
//设置udp客户机助手，告诉客户机助手服务器的ip地址与端口
UdpEchoClientHelper echoClient(csmaInterfaces1.GetAddress(0), 9);
//设置回显请求参数
//设置发送回显请求个数
echoClient.SetAttribute("MaxPackets", UintegerValue(1));
//设置每两个回显请求时间发送间隔
echoClient.SetAttribute("Interval", TimeValue(Seconds(1.0)));
//回显请求大小
echoClient.SetAttribute("PacketSize", UintegerValue(1024));

//将客户机安装到两个通信设备中的另一台设备上
ApplicationContainer clientApps = echoClient.Install(nodes.Get(1));

//设置客户机的运行时间
clientApps.Start(Seconds(2.0));
clientApps.Stop(Seconds(10.0));

//以LOG_INFO级别输出udp日志
LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
```

4.4 让节点保持静止

```
//获取移动助手
MobilityHelper mobility;
//设置地址分配器
```

```
Ptr<ListPositionAllocator> positionAlloc =
CreateObject<ListPositionAllocator> ();
//给地址分配器添加要分配的地址
positionAlloc->Add (Vector (99.0, 97.0, 0.0));
positionAlloc->Add (Vector (94.0, 97.0, 0.0));
positionAlloc->Add (Vector (99.0, 92.0, 0.0));

//将地址分配器加入到移动助手中，此时会分别设置各个节点的初始值
mobility.SetPositionAllocator (positionAlloc);
//设置完初始值，要让节点保持静止，设置移动模型为ConstantPositionMobilityModel
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
//将该移动助手应用到nodes中
mobility.Install (nodes);
```

4.5 启动与结束仿真

```
AnimationInterface anim("examplecsma.xml");
Simulator::Stop(Seconds(10.0));
Simulator::Run();
Simulator::Destroy();
return 0;
```

5. 打印所有节点的路由表

```
//开启全局路由
Ipv4GlobalRoutingHelper::PopulateRoutingTables();

// 创建一个 OutputStreamWrapper，将 std::cout 包装为 Ptr<OutputStreamWrapper>
Ptr<OutputStreamWrapper> stream = Create<OutputStreamWrapper>(&std::cout);
//在仿真时间1.0秒时打印所有节点的路由表
Ipv4GlobalRoutingHelper::PrintRoutingTableAllAt(Seconds(1.0), stream);
```

6. pcap抓包

```
//---------------------开启pcap跟踪-----------------------------------------
    //开启P2PHelper类对象的pcap; "second"为保存文件的前缀名
    pointToPoint.EnablePcapAll("second");

    //开启csmaHelper类对象的pcap
    //使用CSMA网段索引为1的设备（第二个设备）进行sniff，True开启Promiscuous mode
```

```
    csma.EnablePcap("second", csmaDevices.Get(1), true);

    pointToPoint.EnablePcapAll("third");
    phy.EnablePcap("third", apDevices.Get(0));
    csma.EnablePcap("third", csmaDevices.Get(0), true);
```

6. 验证路由表

代码块

```
// 在 PopulateRoutingTables() 后添加
Ptr<OutputStreamWrapper> routingStream = Create<OutputStreamWrapper>
("routes.txt", std::ios::out);
Ipv4GlobalRoutingHelper::PrintRoutingTableAllAt(Seconds(0.0), routingStream);
```

7. 开启多路径ECMP随机路由/配置静态路由

代码块

```
Config::SetDefault("ns3::Ipv4GlobalRouting::RandomEcmpRouting",
BooleanValue(true));
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

Ptr<Ipv4StaticRouting> staticRoutingNode0 =
Ipv4RoutingHelper::GetRouting<Ipv4StaticRouting>(
    nodes.Get(0)->GetObject<Ipv4>()->GetRoutingProtocol()
);

// 添加路由规则：目标网络为 "10.1.2.0/24"，通过下一跳 "10.1.1.2"（节点1）发送
staticRoutingNode0->AddNetworkRouteTo(
    Ipv4Address("10.1.2.0"),  // 目标网络
    Ipv4Mask("255.255.255.0"), // 子网掩码
    Ipv4Address("10.1.1.2"),   // 下一跳地址（节点1的接口IP）
    1                          // 出口接口索引（节点0连接到节点1的接口）
);
```

8. 获取节点的相邻节点与对应链路编号

代码块

```
#include <ns3/node.h>
#include <ns3/net-device.h>
```

```cpp
#include <ns3/channel.h>
#include <vector>
#include <map>

using namespace ns3;

// 获取指定节点的所有相邻节点及连接链路
std::map<Ptr<Node>, Ptr<Channel>> GetNeighborInfo(Ptr<Node> node) {
    std::map<Ptr<Node>, Ptr<Channel>> neighbors;

    // 遍历节点的所有网络设备
    for (uint32_t devId = 0; devId < node->GetNDevices(); ++devId) {
        Ptr<NetDevice> localDev = node->GetDevice(devId);
        Ptr<Channel> channel = localDev->GetChannel();

        if (!channel) continue;  // 跳过无物理连接的设备

        // 获取信道上的所有设备
        for (uint32_t chDevId = 0; chDevId < channel->GetNDevices();
++chDevId) {
            Ptr<NetDevice> remoteDev = channel->GetDevice(chDevId);

            // 排除自身设备
            if (remoteDev == localDev) continue;

            Ptr<Node> remoteNode = remoteDev->GetNode();
            neighbors[remoteNode] = channel;
        }
    }

    return neighbors;
}

// 使用示例
void ExampleUsage() {
    Ptr<Node> currentNode = ...; // 获取目标节点

    auto neighborMap = GetNeighborInfo(currentNode);

    for (auto& pair : neighborMap) {
        Ptr<Node> neighbor = pair.first;
        Ptr<Channel> link = pair.second;

        std::cout << "Neighbor Node ID: " << neighbor->GetId()
                  << " via Channel: " << link->GetId()
                  << std::endl;
    }
```

```
49    }
```

```
user@user-virtual-machine:~/Downloads/ns-allinone-3.38/ns-3.38$ ./ns3 run scratc
h/auto_route/main.cc
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable ../build/scratch/auto_route/ns3.38-main-default
Neighbor Node ID: 2 via Channel: 4
Neighbor Node ID: 3 via Channel: 5
Neighbor Node ID: 5 via Channel: 6
At time +2s client sent 1024 bytes to 10.0.0.1 port 9999
user@user-virtual-machine:~/Downloads/ns-allinone-3.38/ns-3.38$
```

9. 通过链路编号获取相应链路信息

```
代码块
1
```