利用 NS-3 平台分析 802.11 无线网络协议性能

李珊娜,安志强
(北京交通大学 信息中心,北京 100044)

摘 要:该文介绍了 802.11 无线网络协议的类型,详细分析了 NS-3 的主要组件和仿真过程。利用 NS-3 仿真平台编写仿真脚本,模拟实现 Sta-AP 和 ad hoc 网络在不同的网络拓扑结构中的性能状况,通过调整 AP 数量和节点移动速度设置不同的网络环境,并在网络节点上加载不同的路由协议,调整不同的网络性能参数,对 NS-3 平台特点有了更深的认识。利用 Wireshark 和 NS-3 平台的 FlowMonitor 方法获取网络性能数据,通过对比分析 FlowMonitor 和 Wireshark 监测流量的差距,发现 FlowMonitor 方法本身的局限性,从而有助于现实环境中更科学地选择网络协议部署网络环境。借助于高效的仿真分析平台,最终得到一系列有价值的实验结果,为下一步研究奠定了良好的实践基础。

关键词:NS-3 仿真;802.11 协议;ad hoc;基础性网络;FlowMonitor;Wireshark
中图分类号:TP393 **文献标识码**:A **文章编号**:1673-629X(2022)04-0080-06
doi:10.3969/j.issn.1673-629X.2022.04.014

Analysis of 802.11 Wireless Network Protocol Performance by NS-3 Platform Simulation

LI Shan-na, AN Zhi-qiang
(Network Center, Beijing Jiaotong University, Beijing 100044, China)

Abstract: We introduce the types of 802.11 wireless network protocols and analyze the main components and simulation process of NS-3 in detail. The simulation script is written by NS-3 simulation platform to simulate the performance of Sta-AP and ad hoc networks in different network topologies. By adjusting the number of AP and the speed of nodes to set different network environments, loading different routing protocols on network nodes, and adjusting different network performance parameters, we have a deeper understanding of the characteristics of NS-3 platform. Using Wireshark and FlowMonitor method provided by NS-3 platform to obtain network performance data, the limitations of FlowMonitor method are found after the comparative analysis of the gap between FlowMonitor and Wireshark. It is helpful to choose network protocol for deployment network environment more scientifically in real environment. With the help of efficient simulation and analysis platform, a series of valuable experimental results are finally obtained, which lays a good foundation for further research.

Key words: NS-3 simulation; 802.11 protocol; ad hoc; infrastructure network; FlowMonitor; Wireshark

0 引 言

IEEE 802.11 协议是一种无线局域网协议,主要包括两种类型,即独立型网络(以下简称 ad hoc 网络)和基础型网络(以下简称 Sta-AP 网络)。ad hoc 网至少由两个工作站直接通信组成,其中一个工作站负责接入点 AP 的工作,工作站间彼此可以直接通信,两者间的距离必须在可以直接通信的范围内[1];Sta-AP 网至少包含一个接入点 AP 和一个工作站,接入点负责基础型网络所有的传输[2]。

该文主要介绍利用 NS-3 平台仿真不同的网络拓扑结构,通过调整 AP 数量、节点之间的距离、节点移动速度等参数设置不同的网络环境。实验主要设计模拟三个节点拓扑结构、五个节点拓扑结构、五个节点远距离拓扑结构、六个节点两个 AP 拓扑结构的网络环境,在每一种拓扑结构下对比 Sta-AP 网络和 ad hoc 网络的性能参数,得出一系列结论为现实网络环境搭建提供辅助决策。

实验中利用 Wireshark 和 NS3 的 FlowMonitor 功能作为网络性能分析手段,在网络发送端节点上加载 NS-3 的 OnOff 应用,产生发送数据包流量,在网络接

收稿日期:2021-03-01 **修回日期**:2021-07-06
基金项目:国家留学基金委项目资助(201707095088)
作者简介:李珊娜(1980-),女,高级工程师,硕士,通讯作者,从事计算机网络研究。

收端节点上加载 NS-3 的 PacketSinkHelper 应用模拟接收数据包应用。当节点安装 ad hoc 协议时,通过加载不同的路由协议和设置节点移动参数比较其网络性能差异情况,也验证了路由协议的部分特性,发现 Wireshark 和 NS-3 的 FlowMonitor 在特定情况下获取网络流量的区别,最终得出结论为 FlowMonitor 方法本身的局限性导致这样的现象。

1　NS-3 仿真平台

NS-3 是为网络研究和教育而开发的网络仿真平台,它提供了一个分组数据网络如何工作和运行的模型,也为用户提供了一个用于仿真实验的仿真引擎。NS-3 使用经典的 Unix 语言作为环境工具,在 GNU/Linux 平台下开发,用 C++语言实现,兼容时下流行的 Python[3]。

NS-3 仿真流程主要通过编写网络仿真脚本实现,根据实际仿真对象和仿真场景选择相应的仿真模块,主要包括以下内容:

(1)生成网络节点[4]:NS-3 中网络节点是基本计算设备的抽象,节点由 C++类中的 Node 类来描述,Node 类提供了管理计算设备所需的各种方法,如网卡、应用程序、协议栈等[5]。

(2)安装网络设备:不同的网络类型有不同的网络设备,网络设备由 C++中的 NetDevice 类来描述,NetDevice 类提供管理连接节点和信道的各种方法,如 CsmaNetDevice、WiFiNetDevice 和 PointToPointNetDevice 等。

(3)安装协议栈:NS-3 网络中一般是 TCP/IP 协议栈,依据网络选择具体协议栈,如是 UDP 还是 TCP,选择具体的路由协议(OLSR、AODV 和 Global 等),并为其配置相应的 IP 地址。

(4)安装应用层协议:C++中用 Application 类来描述被仿真的应用程序,这个类提供了管理仿真时用户层应用的各种方法。依据选择的传输层协议选择相应的应用层协议[6]。

通过以上网络仿真脚本的编写,完成网络场景配置。由此可见,搭建 NS-3 网络仿真场景和搭建实际网络类似。实验中各种网络拓扑结构的模拟都将采用上述过程和方法,熟练使用 NS-3 仿真流程尤为重要,可以参考官网的方法和实例。

2　性能分析工具

2.1　Wireshark

Wireshark 是一个网络封包分析软件。网络封包分析软件的功能是撷取网络封包,并显示出最为详细的网络封包资料。Wireshark 使用 WinPCAP 作为接口,直接与网卡进行数据报文交换。使用 Wireshark 将得到大量的冗余数据包列表,使用捕捉过滤器和显示过滤器会有助于在大量的数据中迅速定位所需信息。实验中着重分析 Wireshark 产生的 pcap 文件,它反映了网络真实的交互过程及性能数据[7]。

2.2　FlowMonitor

FlowMonitor 是为 NS-3 设计的网络监控功能,它自动检测网络中所有的流,可以方便地收集存储 NS-3 生成的网络性能数据。NS-3 提供了 FlowMonitorHelp 类,可以方便地创建 FlowMonitor。它使用安装在网络节点中的探针来跟踪节点交换的数据包,并进行测量一些参数。数据包根据它们所属的流进行划分,其中每个流根据探测器的特征定义,即包含协议、源地址、源端口、目的地址、目的端口的五元组。收集每个流的统计信息可以以 XML 格式导出,也可以直接访问探针以请求关于每个流的特定统计数据。

3　仿真内容与结果分析

在客户端安装相同的应用产生网络流量[8],通过 Wireshark 和 FlowMonitor 来监控服务器端每秒接收网络包的数量。下面将分别介绍不同网络拓扑结构下的实验结论。

3.1　三个节点网络拓扑

三个节点组成等边三角形的拓扑结构,如图 1 所示,每个网络节点都将同时安装 Sta-AP 和 ad hoc 协议。其中 n0 作为发送端节点,n2 为接收端节点,n0 节点上安装由 OnOffHelper 类生成的 OnOff 应用,通过监测 n0 至 n2 的网络流量得出实验结论。

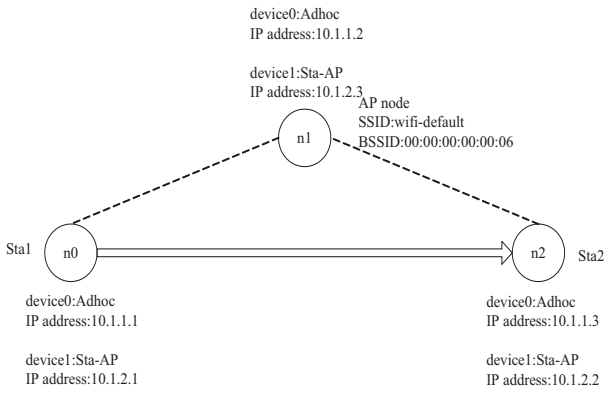

图 1　三个节点拓扑结构

当 n0 和 n2 之间的距离小于 294 米时,从图 2 中得出结论 ad hoc 性能更好一些;当 n0 和 n2 之间的距离等于 294 米时,三角形高度等于 255 米时,模拟 ad hoc 和 Sta-AP 协议,n2 每秒接收的数据包都为 0;当距离不变,三角形高度调整为 50 米时,Sta-AP 协议下 n2 每秒接收的数据包为 107 个,而 ad hoc 协议下 n2 每秒接收的数据包仍为 0 个。由此可见,当超过一定

距离范围时,Sta-AP 表现的更好。

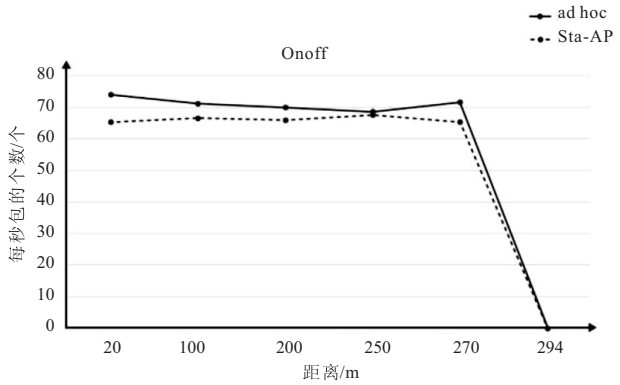

图 2 不同距离时每秒接收包的数量

既然超过一定距离时 Sta-AP 表现更好,尝试在当 n0 和 n2 之间的距离等于 294 米时,三角形高度等于 50 米时加载路由协议,可以发现加载 AODV 协议[9]时 ad hoc 表现的更好,加载 OLSR 协议时[10],Sta-AP 表现更好。

3.2 五个节点网络拓扑

五个节点可以互相通信,每个网络节点都将安装 Sta-AP 或 ad hoc 协议,当节点安装 ad hoc 协议时,同时也加载路由协议。结构如图 3 所示。实验中两条链路将产生网络传输流量,第一条为 n0 作为发送端节点,n2 为接收端节点;第二条为 n3 作为发送端节点,n4 为接收端节点。n0、n3 节点上安装由 OnOffHelper 类生成的 OnOff 应用,通过监测两条链路的网络流量得出实验结论。

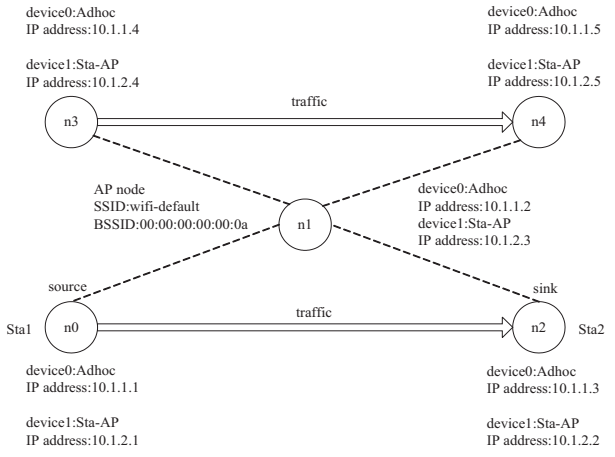

图 3 五个节点拓扑结构

从图 4、图 5 的实验结果可以发现,无论 AP 节点是静止还是移动,ad hoc 模式下接收包的数量总是 Sta-AP 模式的两倍。

通过分析 Wireshark 产生的 pcap 抓包文件可以解释上述结论的原因。通过比较 Sta-AP 和 ad hoc 模式下一个包的传输时间,发现 ad hoc 模式下 n0 到 n2 用时 0.004 449 s,Sta-AP 模式下 n0 到 n2 用时0.008 898 s,所以 ad hoc 模式下接收包的数量总是 Sta-AP 模式的两倍。Sta-AP 模式下 n0 节点要先连接 n1(AP 节点)传输用时 0.004 449 s,然后 n1 再连接到 n2 传输用时 0.004 449 s,所以累计就是 0.008 898 s。因此,这也是这种网络拓扑结构下 ad hoc 模式比 Sta-AP 模式性能更胜一筹的原因。

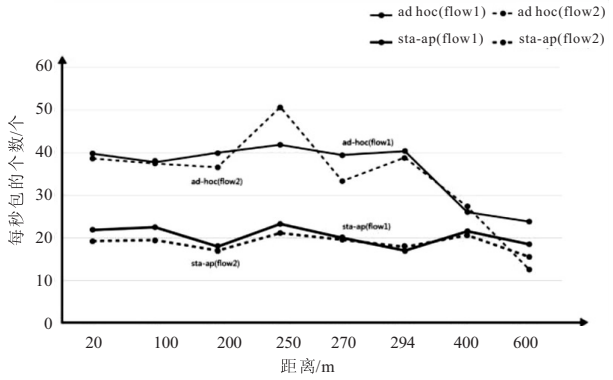

图 4 AP 静止时每秒接收包的数量

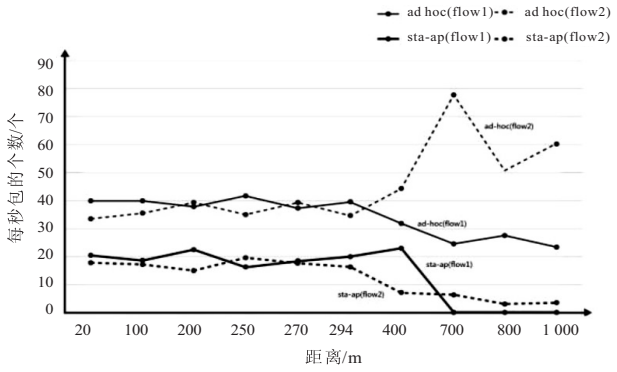

图 5 AP 移动时每秒接收包的数量

3.3 五个节点远距离的网络拓扑

通过设置 n0 到 n2 之间、n3 到 n4 之间不同的远距离长度如 600 米、1 500 米、3 000 米,比较 ad hoc 模式和 Sta-AP 模式的性能,如图 6 所示。每个网络节点都将安装 Sta-AP 或 ad hoc 协议,当节点安装 ad hoc 协议时,同时也加载路由协议。

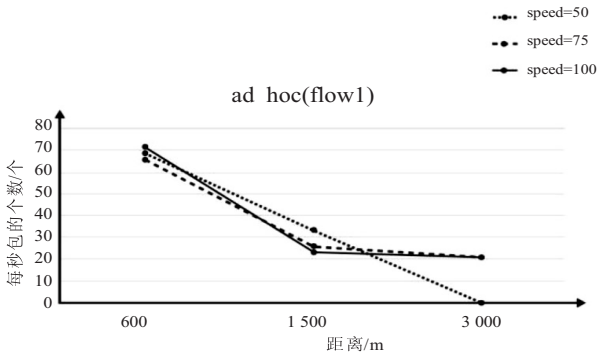

图 6 节点速度变化后的性能

实验中设置除了 n1 节点外的节点一直都在移动,比较 n1 节点静止和移动状态下,两个协议的性能。

实验结果发现不管中间节点 n1 静止还是移动,ad hoc 模式都比 Sta-AP 模式性能更好。随着距离的增

加,增加节点的移动速度时,无论是 ad hoc 模式还是 Sta-AP 模式,服务端收到的包都在增加。

3.4　六个节点两个 AP 的网络拓扑

这个场景选取 n0、n1 为两个 AP 节点,n2、n3 为客户端节点,负责发送数据包,n4 和 n5 为服务端节点,用于接收数据包。左侧的 n2、n3 连接 n0 即 AP1(SSID =wifi1),右侧 n4、n5 连接 n1 即 AP2(SSID = wifi2),AP1 和 AP2 之间用有线连接起来,每个网络节点都将安装 Sta-AP 或 ad hoc 协议。实验中两条链路产生网络传输流量,第一条 flow1 为 n3 作为发送端节点,n5 为接收端节点,第二条 flow2 为 n2 作为发送端节点,n4 为接收端节点。拓扑结构如图 7 所示。

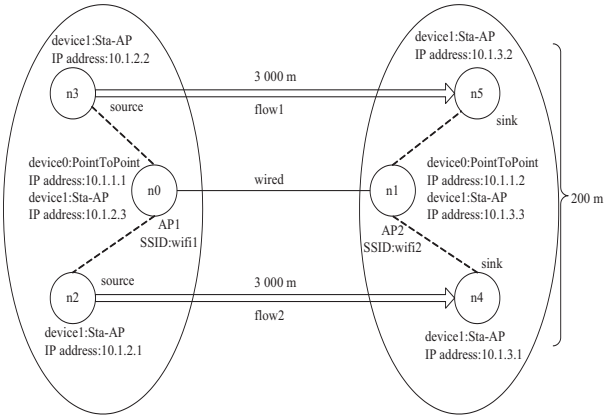

图 7　六个节点两个 AP 的网络拓扑

//创建使用 P2P 链路的两个节点 n0 和 n1,每一个节点安装点到点的网络设备,在它们之间有一个点到点的信道。两个设备被配置在一个有 2 ms 传输延时的信道上,以 5 Mbps 的速率传输数据[11]。

```
NodeContainer p2pNodes;
p2pNodes.Create(2);
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute("DataRate",StringValue("5Mbps"));
pointToPoint.SetChannelAttribute("Delay",StringValue("2ms"));
NetDeviceContainer p2pDevices;
p2pDevices = pointToPoint.Install(p2pNodes);
```

//创建左侧无线自组网节点 n2 和 n3,并将 n0 作为 AP1 节点。AP1 是个双模节点,安装有 Sta-AP 和 PointToPoint 两个网络设备。

```
NodeContainer leftAdhocNodes;
leftAdhocNodes.Create(2);
leftAdhocNodes.Add(p2pNodes.Get(0));
```

//创建右侧无线自组网节点 n4 和 n5,并将 n1 作为 AP2 节点。AP2 是个双模节点,安装有 Sta-AP 和 PointToPoint 两个网络设备。

```
NodeContainer rightAdhocNodes;
rightAdhocNodes.Create(2);
rightAdhocNodes.Add(p2pNodes.Get(1));
```

//NS3 缺省的设置是 802.11a 采用的频段,实验中设置为 802.11b 协议。

```
WifiHelper wifi;
wifi.SetStandard(WIFI_PHY_STANDARD_80211b);
```

//初始化 WIFI 信道和物理层

```
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default();
```

//设置传输损耗模型[12]

```
wifiPhy.Set("EnergyDetectionThreshold",DoubleValue(-80));
wifiPhy.Set("CcaMode1Threshold",DoubleValue(-81));
YansWifiChannelHelper wifiChannel;[13]
wifiChannel.SetPropagationDelay("ns3::ConstantSpeedPropagationDelayModel");
wifiChannel.AddPropagationLoss("ns3::FriisPropagationLossModel");
wifiPhy.SetChannel(wifiChannel.Create());
```

//分别为左侧和右侧节点装载 Wifi 协议

```
WifiMacHelper wifiMac;
wifiMac.SetType("ns3::AdhocWifiMac");
NetDeviceContainer leftdevices = wifi.Install(wifiPhy,wifiMac,leftAdhocNodes);
NetDeviceContainer rightdevices = wifi.Install(wifiPhy,wifiMac,rightAdhocNodes);
```

//设置移动模型,让节点 n2、n3、n4、n5 移动起来[14]

```
mobility.SetMobilityModel("ns3::RandomWaypointMobilityModel","Speed",StringValue(speedConstantRandomVariableStream.str()),
"Pause",StringValue(pauseConstantRandomVariableStream.str()),"PositionAllocator",PointerValue(taPositionAlloc));
mobility.SetPositionAllocator(taPositionAlloc);
mobility.Install(leftAdhocNodes.Get(0));
mobility.Install(leftAdhocNodes.Get(1));
mobility.Install(rightAdhocNodes.Get(0));
mobility.Install(rightAdhocNodes.Get(1));
```

//设置移动模型,让节点 n0、n1 移动起来

```
Ptr<ListPositionAllocator> positionAlloc =
CreateObject<ListPositionAllocator>();
positionAlloc->Add(Vector(200,100,0));
positionAlloc->Add(Vector(2800,100,0));
mobility.SetPositionAllocator(positionAlloc);
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(p2pNodes);
```

//实验中使用 OnOff 模式模拟产生流量,设置 n2 向 n4 节点发送流量,设置 On 状态下的速率和发包大小

```
OnOffHelper onoff1("ns3::UdpSocketFactory",InetSocketAddress(rightAdhocInterfaces.GetAddress(0),45));
onoff1.SetAttribute("PacketSize",UintegerValue(packetSize));
onoff1.SetAttribute("DataRate",StringValue("
```

```
3000000bps"));
    onoff1.SetAttribute("OnTime",StringValue("ns3::ConstantRandomVariable[Constant=1]"));
    onoff1.SetAttribute("OffTime",StringValue("ns3::ConstantRandomVariable[Constant=0]"));
    //将 n2 设置为发送数据包节点
    ApplicationContainer app1 = onoff1.Install(leftAdhocNodes.Get(0));
    app1.Start(Seconds(var->GetValue(m_dataStart, m_dataStart + 1)));
    app1.Stop(Seconds(800.0))
    //实验中使用 PacketSinkHelper 模拟接收数据包,将 n4 设置为接收数据包节点。
    PacketSinkHelper sink2("ns3::UdpSocketFactory", InetSocketAddress(rightAdhocInterfaces.GetAddress(0),45));
    ApplicationContainer app2 = sink2.Install(rightAdhocNodes.Get(0));
    app2.Start(Seconds(var->GetValue(m_dataStart, m_dataStart + 1)));
    app2.Stop(Seconds(800.0));
    //实验中使用 FlowMonitorHelper 监控各个节点的流量
    FlowMonitorHelper flowmon;
    Ptr<FlowMonitor> monitor=flowmon.Install(allNodes);
    NS_LOG_INFO("Run Simulation.");
    Simulator::Stop(Seconds(800.0));
    Simulator::Run();
    monitor->CheckForLostPackets();
    Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier>(flowmon.GetClassifier());
    std::map<FlowId,FlowMonitor::FlowStats> stats=monitor->GetFlowStats();
    //监控各个节点的流量并输出
    for(std::map<FlowId,FlowMonitor::FlowStats>::const_iterator i=stats.begin();i!=stats.end();++i)
    {
    Ipv4FlowClassifier::FiveTuplet=classifier->FindFlow(i->first);
    //监控 flow1 的流量并输出发送包和接收包的数量
    if((t.sourceAddress=="10.1.2.2" && t.destinationAddress=="10.1.3.2" && t.destinationPort==45))
    {
    std::cout<<"Flow"<<i->first<<" protocol"<<unsigned(t.protocol)<<"(" <<t.sourceAddr
    ess<<"->"<<t.destinationAddress<<")\n";
    std::cout<<"Ap1-Sta1 transmitted packets:"<<i->second.txPackets<<"\n";
    std::cout<<"Ap2-Sta1 received packets:"<<i->second.rxPackets<<"\n";
    }
    //监控 flow2 的流量并输出发送包和接收包的数量[15]
    else if((t.sourceAddress=="10.1.2.1" && t.destinationAddress=="10.1.3.1" && t.destinationPort==45))
    {
    std::cout<<"Flow  "<<i->first<<" protocol"<<unsigned(t.protocol)<<"(" <<t.sourceAddress<<"->"<<t.destinationAddress<<")\n";
    std::cout<<"Ap1-Sta2 transmitted packets:"<<i->second.txPackets<<"\n";
    std::cout<<"Ap2-Sta2 received packets:"<<i->second.rxPackets<<"\n";
    }
```

实验结果表明(见图 8),如果客户端节点不安装任何路由协议,将不会发送任何数据包,安装路由协议后,将有数据流量产生。当所有节点都静止时,ad hoc 模式比 Sta-AP 模式性能更好,当除了两个 AP 节点静止其他所有节点都移动时,尤其是远距离时,Sta-AP 模式性能更好。当两个 AP 共用一个信道时,彼此有干扰发生。

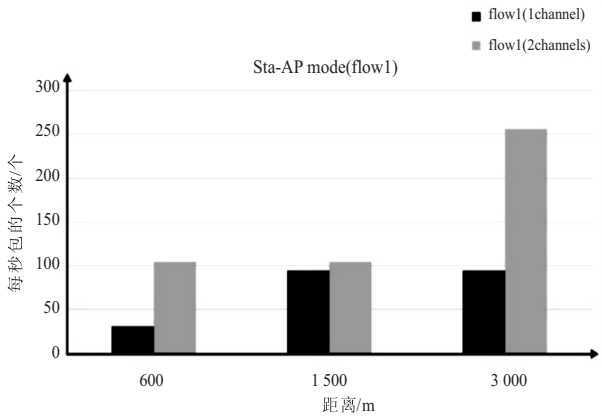

图 8 不同信道数量的传输性能

3.5 FlowMonitor 和 Wireshark 的区别

当选择六个节点两个 AP 的拓扑结构,选择 802.11b 协议模拟时,通过 FlowMonitor 和 Wireshark 监测流量的差距很大。起初分析是由于 FlowMonitor 的参数 MaxPerHopDelay 引起的,这个参数设置每一跳的最大延时,缺省值为 10 秒,如果超过最大延时将被认为丢包。但当分析 Wireshark 的 pcap 抓包文件,发现每一跳的延时都没有超过 10 秒。

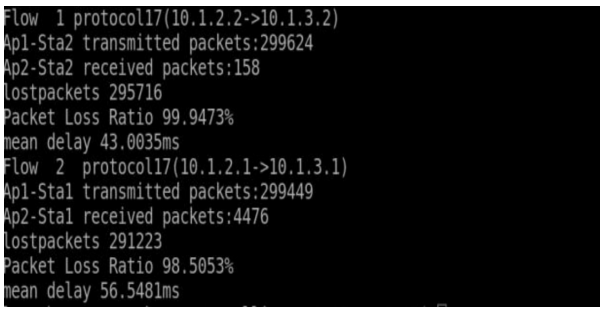

图 9 802.11a 协议下的 FlowMonitor 流量监测

将 802.11b 协议换作 802.11a 协议重复模拟上述现象时,发现通过 FlowMonitor 和 Wireshark 监测流量

的差距很小,如图 9 所示。使用 Wireshark 监测 flow1 接收包数为 217 个,flow2 接收包数为 4 479 个;使用 FlowMonitor 监测 flow1 接收包数为 158 个,flow2 接收包数为 4 476 个。由此推断 FlowMonitor 方法本身的局限性导致这样的现象。

4　结束语

该文主要研究了通过 NS-3 平台仿真不同网络拓扑结构,充分比较 ad hoc 模式和 Sta-AP 模式性能情况。利用 Wireshark 和 FlowMonitor 监测分析数据包,得出一系列实验结果,为搭建真实网络环境提供依据。Ad hoc 模式提供有限的无线传输范围,传输速度是 Sta-AP 模式的两倍。Sta-AP 模式必须要连接有线网,可以拓展无线网的范围,但是传输速度也减少一半,因为它需要花费时间传输信号到 AP 而不是直接发送到目的节点。FlowMonitor 方法本身的局限性导致。

参考文献:

[1] 徐　扬. AdHoc 网络性能分析及路由技术研究[D]. 西安:西安电子科技大学,2014.

[2] 倪晓伟. 基于 NS-3 无线自组网多信道路由协议研究[D]. 北京:北京邮电大学,2012.

[3] 陈新锴,吕光宏. NS-3 下新队列管理模块的实现[J]. 软件导刊,2009(10):19-21.

[4] 沈德海. NS-3 在计算机网络课程教学中的运用[J]. 重庆科技学院学报:自然科学版,2011,13(2):161-163.

[5] 张登银,张保峰. 新型网络模拟器 NS-3 研究[J]. 计算机技术与发展,2009,19(11):80-84.

[6] 闵圣天,曾文序,李满荣,等. 基于 NS3 的网络协议分析与模拟[J]. 福建电脑,2014(2):99-100.

[7] 李　越,钱德沛,何　莹,等. 网络仿真器 NS 问题分析及改进方案[J]. 系统仿真学报,2005,17(11):2832-2836.

[8] 陈宇峰,董亚波,鲁东明. 网络流量模拟方法研究进展[J]. 计算机工程与设计,2009,30(6):1356-1359.

[9] 常秀丽. 基于 NS-3 的 Ad Hoc 网络路由协议研究与仿真[D]. 哈尔滨:哈尔滨工业大学,2010.

[10] 吴敬敬,李忠民,李建坤. Ad-Hoc 网络中 3 种典型路由协议的仿真分析与比较[J]. 南昌航空大学学报:自然科学版,2012,26(4):100-105.

[11] 蔡文郁,刘晓玲. 计算机网络启发式 NS-3 仿真案例教学模式[J]. 实验室研究与探索,2018,37(9):95-100.

[12] 顾　洁,朱宗卫,徐友庆,等. NS-3 仿真环境中 IEEE802.11 服务区分机制的研究[J]. 计算机工程,2019,45(8):135-140.

[13] 王　悦. NS-3 802.11 物理层源代码实现原理分析[J]. 计算机科学,2016,43(z1):282-283.

[14] 刘东亮. 基于 NS-3 的机会网络路由协议仿真技术研究[D]. 哈尔滨:哈尔滨工程大学,2014.

[15] 茹新宇,刘　渊. 网络仿真器 NS3 的剖析与探究[J]. 计算机技术与发展,2018,28(3):72-77.

(上接第 68 页)

ceedings of the IEEE conference on computer vision and pattern recognition. Salt Lake City, UT, USA: IEEE, 2018: 389-398.

[14] PARKIN A, GRINCHUK O. Recognizing multi-modal face spoofing with face recognition networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition workshops. Long Beach, CA, USA: IEEE, 2019.

[15] YU Z, QIN Y, LI X, et al. Multi-modal face anti-spoofing based on central difference networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition workshops (CVPRW). Seattle, WA, USA: IEEE, 2020: 650-651.

[16] SHEN T, HUANG Y, TONG Z. Facebagnet: bag-of-local-features model for multi-modal face anti-spoofing[C]//Proceedings of the IEEE conference on computer vision and pattern recognition workshops(CVPRW). Long Beach, CA, USA: IEEE, 2019.

[17] HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. Las Vegas, NV, USA: IEEE, 2016: 770-778.

[18] HU J, SHEN L, SUN G. Squeeze-and-excitation networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. Salt Lake City, UT, USA: IEEE, 2018: 7132-7141.

[19] ZHANG S, WANG X, LIU A, et al. A dataset and benchmark for large-scale multi-modal face anti-spoofing[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. Long Beach, CA, USA: IEEE, 2019: 919-928.

[20] ZHANG S, LIU A, WAN J, et al. Casia-surf: a large-scale multi-modal benchmark for face anti-spoofing[J]. IEEE Transactions on Biometrics, Behavior, and Identity Science, 2020, 2(2): 182-193.

[21] LIU A, TAN Z, LI X, et al. Static and dynamic fusion for multi-modal cross-ethnicity face anti-spoofing[J]. arXiv: 1912.02340, 2019.