

计 算 机 网 络 实 验

实 验 指 导 书

|  |  |
| --- | --- |
| 实验名称 | NS-3的WiFi模块的MAC层模型的仿真验证 |

电子信息与通信学院 2018年制

# 实验目的

1. 了解IEEE 802.11 DCF网络的相关协议
2. 安装并熟悉使用NS-3
3. 验证NS-3的WiFi模块的可靠性
4. 验证Dai的IEEE 802.11 DCF网络的统一分析理论模型

# 实验背景

1. **网络仿真技术**

近年来，随着计算机和网络通信技术的不断发展，网络技术的研究也进入到了一个飞速发展的时期。研究人员不断开发出新的网络协议、算法和应用，以适应日益增长的网络通信需要。然而由于网络的不可控、易变和不可预测等特性的存在，给新的网络方案的验证、分析和比较带来了极大的困难。目前网络通信的研究一般分为以下3种方法。

1）分析方法。在理论和协议层面上对网络通信技术或系统进行研究分析，抽象出数学分析模型，利用数学分析模型对问题进行求解。如采用数学建模、协议分析、状态机、集合论以及概率统计等对多种理论分析手段和方法对通信网络及其算法、协议、网络性能等各个方面进行研究。

2）网络模拟。即计算机模拟仿真算法。网络模拟日益成为分析、研究、设计和改善网络性能的强大工具，它通过在计算机上建立一个虚拟的网络平台，来实现真实网络环境的模拟，网络技术研究人员在这个平台上不仅能对网络通信、网络设备、协议以及网络应用进行设计研究，还能对网络的性能进行分析和评价。

3）实验网方法。对网络协议、网络行为和网络性能采用建立实验室测试网络、网络测试平台（network testbed）和小规模商用实验网络的方式对网络进行实战检验。就是设计出研究所需要的合理硬件和软件配置环境，建立测试床和实验室，在现实的网络上进行研究。

以上3种方法有利有弊，相辅相成并各有侧重点。

理论研究适用于早期研究与设计阶段，对新算法和新技术进行理论准备和验证，除了人力和知识，几乎不需要什么额外成本。

实验网方法是网络和系统在投入实际应用前的一次系统的演练，能够发现网络设计与用户需求之间的相合度以及检验网络实际使用的效用和性能。该阶段建设成本很高，要求技术和设备开发相对成熟，网络系统基本成型，主要是对业务、系统稳定性能和服务性能的检验。

而网络模拟阶段可以说是理论方法和实验网方法的中间阶段，它可以对新协议进行初步实现和验证，并有助于新协议的及时调整和改进。网络模拟阶段由于采用计算机软件进行模拟，使得很多研究工作人员能够研究大规模网络和学习新协议新算法的设计和实现，并且能够在网络实用钱对其进行检验和改进。此外，它还可以在各种新老系统和算法之间进行比较而不必花费巨资去建立多个实际系统。因此，网络模拟是网络通信研究中一种非常重要的方法。

1. **NS-3简介**

俗话说“工欲善其事必先利其器”。为了方便网络研究人员的研究工作，一款好的网络模拟软件是必不可少的。它必须具备以下特点：

1）能够提供简单易用的使用界面，能方便快捷地建立和修改模拟环境和模拟配置；

2）能够提供常用的绝大部分网络协议、算法和应用模块；

3）能够方便地进行配置和扩展，有利于添加新的协议和算法。

当前有许多优秀的网络模拟软件，如NS-2（network simulator version 2）、OPNET（optimized performance network engineering tool）、OMNet++（objective modular network testbed in C++）、GloMoSim（global mobile information system simulator等，这为网络研究和开发人员提供了很好的网络模拟平台。

相比于以上提到的网络仿真软件，NS-3是比较年轻的网络仿真工具。NS-3是尽量吸取现有网络模拟工具的优点并避开其缺点，应用现代软件工程思想和网络仿真技术而设计开发的新一代网络模拟工具。

NS-3不是NS-2的扩展，而是一个全新的网络模拟器，是由美国华盛顿大学的Thomas R.Henderson教授及其研究小组在美国自然科学基金（NFS）的支持下，于2006年开始应用现代网络模拟技术和软件开发技术设计并开发的一个全新网络模拟工具。NS-3广泛汲取了现有优秀开源网络模拟器如NS-2、GTNetS、Yans等的成功技术和经验，专门用于教育和研究用途的离散事件模拟器，它基于GNU GPLv2许可，可以免费地获取、使用和修改。

NS-3相对简单易学、可扩展性好、节省资源，并且能提供高性能的，与真实网络相近的网络仿真，尤其能够集成到实验床和虚拟机环境，逐渐得到学术界的认可，在和仿真相关的国际会议如SIMUTools、SpringSim、ICST-NSTools等，以及网络方面的顶级会议如Sigcomm、Infocom等中研究和使用NS-3的文献逐渐增多，详见<http://www.nsnam.org/overview/publications>。

1. **仿真实验意义**

NS-3具有用于WiFi模块的MAC层的详细模型，但是，由于其复杂性，很少有研究来验证NS-3 MAC层模型。已经进行了许多研究来验证NS-3中的物理层和信道模型[4-8]。 Patidar等人报道了通过改变节点数量对NS-3 MAC层的初步验证研究[9]。 Baldo等人[10]使用测试床验证了NS-3 的MAC模型。在本实验中，我们提供[1]中的统一IEEE 802.11 DCF分析模型来研究NS-3的WiFi模块的准确性。

# 理论模型分析

在本节中，我们介绍IEEE 802.11 DCF网络的统一分析框架[1]，并概述不饱和网络和饱和网络、基本接入机制和RTS / CTS机制下的网络总速率的表达式，。我们的目标是基于该模型获得的分析结果来评估NS-3 的MAC层模型的准确性。

1. **IEEE 802.11 DCF网络的投统一分析框架**

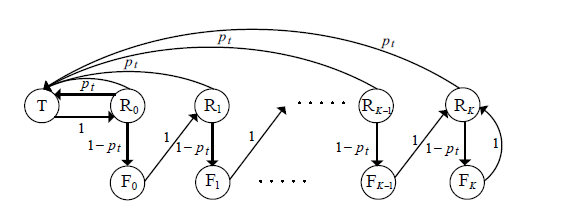


图 3-1 嵌入马尔科夫链模型状态转移图

考虑n节点IEEE 802.11 DCF网络，其中每个节点配备有无限大小的缓冲区，并且每个队列头包（HOL）的最大重传次数是无限的。在[1]中，针对同构IEEE 802.11 DCF网络提出了统一的分析框架，其中每个节点具有相同的传输和退避参数，包括初始退避窗口大小*W*，最大退避等级*K*，数据包长度（单位为bytes）和数据速率（单位为Mbps）。上述模型描述网络节点所处状态，T表示发送状态，指数据传输成功；F表示冲突状态，指尝试发送数据但是失败了；R表示等待传输状态，也就是时间计数器在不断减小的状态。

对于[1]中的不饱和网络，标准化吞吐量输出，定义为数据帧成功传输的时间百分比，由下式给出

（1）

在饱和网络中，即每个节点的队列缓存区始终有数据包等待发送。 由[1]可知，标准化吞吐量为

（2）

其中和分别表示数据帧成功发送和发生冲突的平均时间，而是给定信道空闲的前提下，队列头包成功发送的稳态概率的定点方程的非零根：

表明数据帧成功传输的时间效率。然而，在实际网络中，数据速率性能通常更受关注，它反映在每秒钟内成功传输的信息。我们考虑数据速率，定义网络总速率为每秒成功传输的信息位数。因此，在不饱和网络中，从公式（1）可得，网络总速率为

(4)

对于饱和网络，其网络总速率由1）数据帧的成功传输效率，2）每次成功传输中用于数据帧有效载荷传输的时间部分，以及3）传输速率确定，由公式（2）可得

其中，表示时隙长度（以为单位）。

在NS-3的参数设置中，当节点遇到冲突时，节点将经历一段ACKTimeout或CTSTimeout超时时间段。 因此，基本接入机制的成功传输和发生冲突的平均时间可写为

其中，表示控制帧的发送速率（以Mbps为单位）。MAC报头（MH）和ACK帧以字节为单位。PHY头部（PH）、DIFS和SIFS以为单位。

在RTS/CTS机制下，数据帧成功传输和发生冲突的平均时间可写为

其中，RTS和CTS以字节为单位。

# 实验设置

1. **NS-3的WiFi模块的架构**

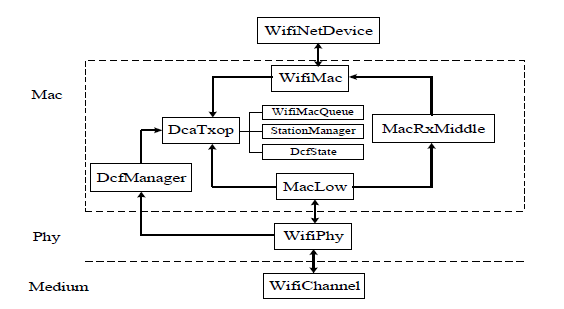


图 4-1 NS-3的WiFi模块的网络架构

NS-3 WiFi模块的网络架构如上图所示。在NS-3 WiFi网络中，节点包含一个WifiNetDevice对象，用于将WifiChannel，WifiPhy，WifiMac和WifiRemoteStationManager连接在一起。 当应用程序启动传输时，WifiNetDevice接口将数据包发送到WifiMac类，该类处理高层MAC级别功能，例如不同的MAC类型，信标，关联等。DcaTxop处理来自DcfManager的对于信道的请求访问。 授予访问权限后，DcaTxop会将数据包推送到MacLow以启动数据传输。 WifiPhy类主要用于接收数据包和能耗跟踪。 WifiChannel旨在与WifiPhy互连，以便可以通过信道接收数据包。

1. **仿真设置**

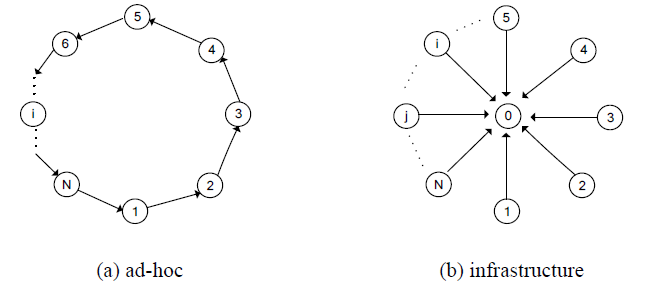


图 4-2 网络拓扑

我们考虑具有不同系统参数的自组织网络和基础设施网络，如图4-2所示，通过比较两种网络拓扑在NS-3仿真中的结果与IEEE 802.11 DCF网络的统一数学模型之间的差异，分析自组织网络与基础设施网络的区别。在基础设施网络中，接入点（AP）需要连续发送信标帧将网络中的基本信息通知给各个节点，并且节点和AP之间的关联也是必要的。另一方面，在ad-hoc网络中避免了由于关联（信标传输，主动扫描等）引起的附加信道活动。请注意，我们只关注数据包有效负载传输，并在数学模型中不考虑关联效应。 因此，我们可以预期在ad-hoc网络中节点仿真结果更接近我们的数学分析。

在ad-hoc网络中，节点数目被设置以5为间隔依次在5到50的范围内增加。每个节点将数据帧以54 Mbps的速率发送到相邻节点。每个节点都是发送者也是接收者。因此，网络总速率是每个节点的网络速率的总和。在仿真实验中，我们利用默认的WiFi信道和YANS模型中的物理层[16]，并选择AdhocWifiMac作为MAC层的类型。目前，NS-3支持多种IEEE 802.11标准，我们选择802.11a标准作为WiFi标准，数据帧的发送速率和控制帧的发送速率分别为6、9、12、18、24、36、48和54 Mbps。系统参数的值总结在表1中。

表I 802.11标准中的系统参数设置

|  |  |  |  |
| --- | --- | --- | --- |
| PHY 头部（PH） | 20 | ACKTimeout | 69 |
| MAC 头部（MH） | 36 bytes | CTSTimeout | 69 |
| ACK | 14 bytes | DIFS | 34 |
| RTS | 20 bytes | SIFS | 16 |
| CTS | 14 bytes | Slot time | 9 |

1. **NS-3仿真基本流程**

使用NS-3进行网络仿真时，一般经过以下4个步骤。

1）选择或开发相应模块

根据实际仿真对象和仿真场景选择相应的模块：如是有线局域网络（CSMA）还是无线局域网络（Wi-Fi）；节点是否需要移动（mobility）；使用何种应用程序（application）；是否需要能量（energy）管理；使用何种路由协议（internet、aodv等）；是否需要动画演示等可视化界面（visualizer、netanim）等。

2）编写网络仿真脚本

有了相应的模块，就可以搭建网络仿真环境，NS-3仿真脚本支持2种语言：C++和Python，但是2种语言的API接口是一样的，部分API可能还没有提供Python接口。编写NS-3仿真脚本的大体过程如下。

* 生成节点：NS-3中节点相当于一个空的计算机外壳，接下来要给这个计算机安装网络所需要的软硬件，如网卡、应用程序、协议栈等。
* 安装网络设备：不同的网络类型有不同的网络设备。从而提供不同的信道、物理层和MAC层，如CSMA、Wi-Fi、WiMAX和point-to-point等。
* 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，如是UDP还是TCP，选择何种不同的路由协议（OLSR、AODV和Global等）并为其配置相应的IP地址，NS-3既支持IPv4也支持IPv6。
* 安装应用层协议：依据选择的传输层协议选择相应的应用层协议，但有时需要自己编写应用层产生网络数据流量的代码。
* 其他配置：如节点是否移动，是否需要能量管理等。
* 启动仿真：整个网络场景配置完毕，启动仿真。

NS-3仿真的基本模型如图3-4所示，搭建NS-3网络仿真场景和搭建实际网络类似，首先需要网络节点，NS-3中使用节点的概念；节点需要有网络设备，类似于网络接口卡，NS-3中有相应网络设备的概念；网络设备通过传输媒体连接，NS-3中使用信道的概念来代表传输媒体，设置信道延迟等属性，并且和实际网络相似：信道和网络设备是对应的，CSMA网络设备对应CSMA的信道，Wi-Fi网络设备对应Wi-Fi的信道。

以上概念使网络节点实现了物理连接，但要实现通信，还需要软件支持，也就是协议，应用层产生数据，利用类socket编程（和真实的BSD socket很像）实现数据分组的向下传递，数据分组通过协议栈—TCP/IP向下传递给网络设备（可以简单理解为网卡），该网络设备包括MAC层、物理层协议，于是数据分组就像在真实网络中流动一样，由数据帧转换成二进制流，最终变成信号通过媒体信道传输到目的节点。

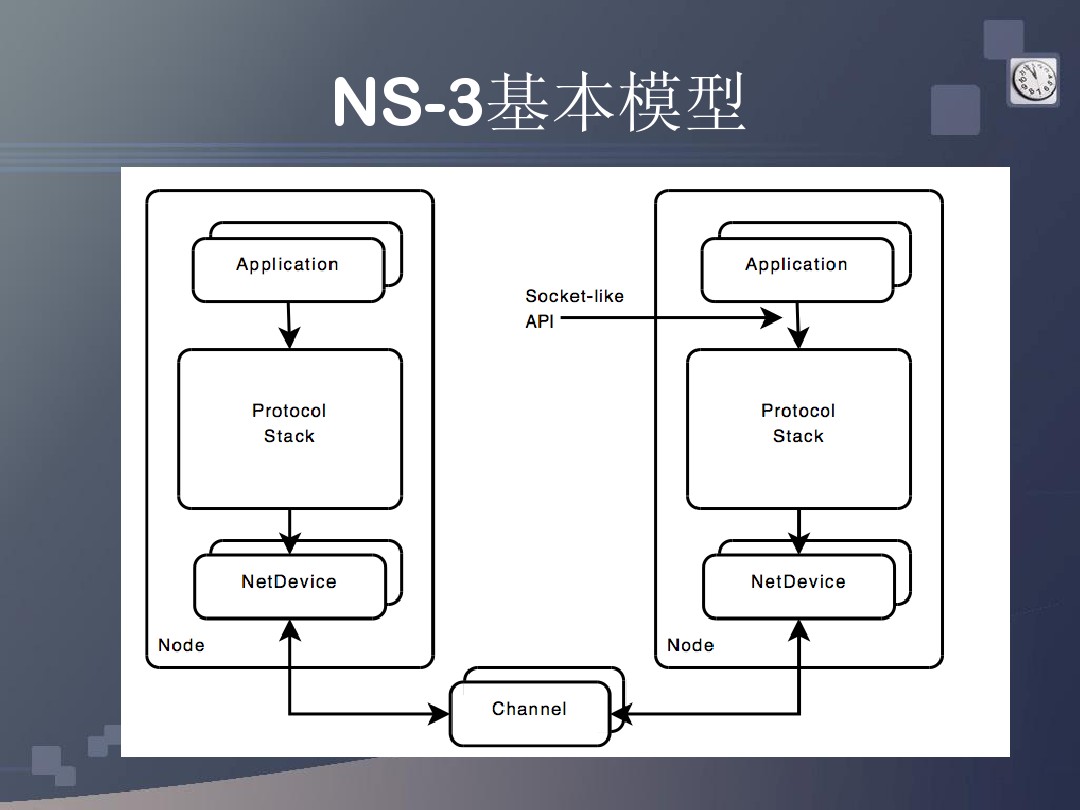


图 3-4 NS-3基本模型

目的节点收到数据分组后从下往上逐层转交，由媒体信号转换成二进制，由二进制转换成数据帧，再有数据帧转换成IP数据分组，然后经由传输层的端口号转交给相应的进程（应用程序application），至此在NS-3中完成了一次和真实网络极其相近的完整数据传输过程。

3）仿真结果分析

仿真结果一般有2种：一是网络场景，二是网络数据。网络场景如节点拓扑结构、移动模型等，一般通过可视化界面（PyViz或NetAnim）可直观观测到；网络数据也可在可视化界面下有简单的统计，此外，可以通过专门的统计框架（status）或者自行通过NS-3提供的追踪（tracing）系统收集、统计和分析相应的网络数据，如数据分组的延迟、网络流量、分组丢失率和节点消息缓存队列等。

4）依据仿真结果调整网络配置参数或修改源代码

有时实际结果和预期相差较远，这时我们要分析原因，是网络参数有问题，还是协议本身有出入等，然后再重新设计，重新仿真，如此反复，直到达到满意的结果。

# 实验内容

1. **NS-3代码示例**

在本节中，我们将在NS-3中展示一系列设计的DCF仿真，并演示分析戴的模型[1]和NS-3 WiFi模块的仿真结果之间的比较。特别是，为了评估戴模型在NS-3 WiFi模块中的性能，我们首先增加流量到达率，将网络从非饱和模式加载到饱和模式，然后设置不同数量的节点、初始退避窗口大小为和最大退避等级，以及不同的接入机制和不同的网络拓扑场景，以获得仿真结果。

本实验是用NS-3仿真802.11a无线局域网中的DCF协议，通过一系列不同的网络场景，如调整网络中的参数（如网络节点数目点、初始退避窗口大小为等）来输出网络总速率，比较NS-3仿真下的不同网络场景中的网络总速率和理论模型的数学分析之间的差异，来验证NS-3的WiFi模块的准确性。

下面先主要介绍一下在实验原理中主要提到的几个关键概念在示例代码中的表现形式：

1）节点

NS-3中基本计算设备被抽象为节点。节点由C++中的Node类来描述。Node类提供了用于管理仿真器中网络组件表示的各种方法。例如示例代码中的NodeContainer类：

NodeContainer c；//声明一个名为“c”的NodeContainer

c.Create(networkSize); //调用nodes对象的Create()方法创建networkSize个节点

NodeContainer的拓扑生成器提供了一种简便的方式来创建、管理和使用任何节点任何节点对象。以上两行声明的容器调用NS-3中的内部函数来产生networkSize个节点对象，并把指向这networkSize个对象的指针存储在系统中。在脚本中它们所代表的节点什么都没有做，构建拓扑的下一步是把这些节点彼此连接成网络。

2）应用

正如“现实世界”中在计算机上运行应用程序以执行各种任务一样，NS-3仿真环境中的应用程序在节点上运行来驱动模拟过程。在NS-3中，需要被仿真的用户被抽象为应用。应用在C++中用Application类来描述。这个类提供了管理仿真时用户层应用的各种方法。开发者应当用面向对象的方法自定义和创建新的应用。在示例代码中，我们分别定义了客户端和服务器的节点，使用Application类的实例：PacketSocketClient和PacketSocketServer。

Ptr<PacketSocketClient> client = CreateObject<PacketSocketClient> (); //创建一个简单的客户端

client->SetRemote (socketAddr); //设置远程地址

c.Get (i)->AddApplication (client); //将应用程序与此节点关联

Ptr<PacketSocketServer> server = CreateObject<PacketSocketServer> ();//创建使用PacketSocket的服务端

server->SetLocal (socketAddr); //设置本地地址

c.Get (j)->AddApplication (server); //将应用程序与此节点关联

这些应用程序包含了一个client/server应用来发送和回应仿真网络中的数据分组。 PacketSocket用来连接应用程序和网络设备之间的通信。

3）信道

通常把网络中数据流流过的媒介称为信道。在NS-3的模拟环境中，可以把节点连接到代表数据交换信道的对象上。在这里，基本的通信子网这一抽象概念被称为信道，在C++中用channel类来描述。Channel类提供了管理通信子网对象和把节点连接至它们的各种方法。信道类同样可以由开发者以面向对象的方法自定义。一个信道实例可以模拟一条简单的线缆（wire），也可以模拟一个复杂的巨型以太网交换机，甚至可以是无线网络中充满障碍物的三维空间。示例代码中设置信道类型的部分如下：

// Set channel type

YansWifiChannelHelper channel = YansWifiChannelHelper::Default (); //设置信道类型

YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();

phy.SetChannel (channel.Create ());

4）网络设备

在NS-3中，网络设备这一抽象概念相当于硬件设备和软件驱动的总和。NS-3仿真环境中，网络设备相当于安装在节点上，使得节点通过信道和其他节点通信。像真实的计算机一样，一个节点可以通过多个网络设备同时连接到多条信道上。

网络设备由C++中的NetDevice类来描述。NetDevice类提供了管理连接其他节点和信道对象的各种方法，并且允许开发者以面向对象的方法来自定义。如果需要一个所有被创建的NetDevice对象列表，就需要使用一个NetDeviceContainer对象来存放它们，示例代码中有device对象被创建和保存在网络设备容器内。

NetDeviceContainer devices = wifi.Install (phy, mac, c); //将物理层、MAC层、节点置于网络设备中；

mac.SetType ("ns3::AdhocWifiMac","Ssid", SsidValue (ssid)); //将MAC层类型设置为Adhoc类型；

1. **研究无线局域网中网络从不饱和到饱和状态的网络总速率**

在NS-3中，我们增加网络总输入率 使网络状态从不饱和到饱和网络，是每时隙生成一个新数据包的概率。通过稳定地增加，网络从不饱和状态转变为饱和状态。

本实验主要研究DCF协议中，在基本接入机制下，随着网络总输入率的增加，网络总速率发生的变化，并分析其中的原因。

修改源码中的节点队列的数据包到达时间间隔T，即变量interval，单位为s/packets.

client->SetAttribute ("Interval", TimeValue (MilliSeconds (1)));//设置数据包的到达时间间隔

在不饱和网络下，由公式（4）可知，理论分析的网络总速率为，

NS-3仿真中可以得到网络总速率为

（10）

由此可得，

那么令公式（4）和公式（5）相等，即当网络总输入率增加，使得网络从不饱和状态到饱和状态。

得出，即等于0.5723时，网络转变为饱和状态。

由公式（11），可以得出T=0.0232;

将网络节点设为n=50, 网络总输入率依次设置为0.1、0.2、0.3、0.4、0.5、0.6、0.7、0.8、0.9、1；

那么，当分别设置为0.1、0.2、0.3、0.4、0.5、0.6、0.7、0.8、0.9、1时，对应的T分别为0.1328、0.0664、0.0442、0.0332、0.0266、0.0232、0.0232、0.0232、0.0232、0.0232。

具体设置如下：

将队列数据包到达间隔interval依次设置为0.1328、0.0664、0.0442、0.0332、0.0266、0.0232、0.0232、0.0232、0.0232、0.0232；输出对应仿真的吞吐量，

client->SetAttribute ("Interval", TimeValue (Time ("0.0232")));//设置数据包的到达时间间隔

利用matlab作图来分析吞吐量和网络总输入率之间的关系，实验结果如下：

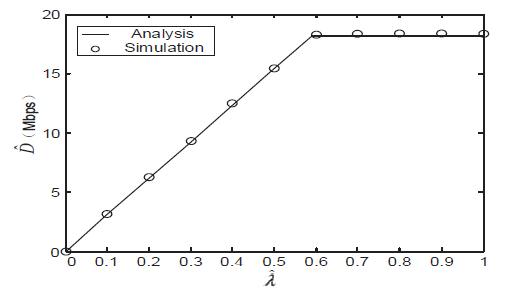


图 5-1 基本接入机制下吞吐量和网络总输入率的关系

1. **研究无线局域网中关键系统参数对网络吞吐量的影响**

本实验主要研究DCF协议中，在基本接入机制下，随着网络中关键系统参数的变化，吞吐量会发生的变化，并分析其中的原因。

1）网络节点数目对吞吐量的影响

修改源码中网络节点数目，设置数据包长PL=1023bytes、初始退避窗口大小CWmin=15、最大退避等级K=CWmax/CWmin=6、数据包发送速率DR=54Mbps和控制帧发送速率BR=6Mbps不变，具体实施如下：

将RTS阈值设为一个大于数据包长的值，这时，网络工作在基本接入机制；

Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", StringValue ("999999")); //设置RTS阈值大小

将网络节点数目依次设置为5、10、15、20、25、30、35、40、45、50；输出对应仿真的吞吐量，

uint32\_t netsize = 50;//节点数目改为设定值

或者在终端修改节点数目，命令如下：

sudo ./waf --run “scratch/adhoc --netsize=50”

利用matlab作图来分析吞吐量和网络节点数目之间的关系，实验结果如下：

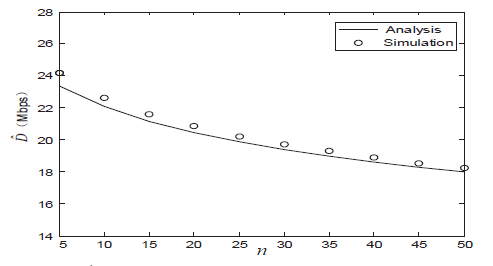


图 5-2 基本接入机制中吞吐量和节点数目*n*的关系

2）初始退避窗口大小对吞吐量的影响

将节点数目设置为50，依次改变初始退避窗口大小，并且保持最大退避等级K=6不变。

即设置CWmin依次为3、7、15、31、63、127、255、511、1023、2047，具体实施如下：

Config::SetDefault ("ns3::DcaTxop::MinCw", UintegerValue (3));//设置初始退避窗口大小

同样地，对应的CWmax分别为255、511、1023、2047、4095、8191、16383、32767、65535、131071；具体实施如下：

Config::SetDefault ("ns3::DcaTxop::MaxCw", UintegerValue (255)); //设置最大退避窗口大小，保持最大退避等级K=6

利用matlab作图来分析吞吐量和初始退避窗口大小之间的关系，实验结果如下：

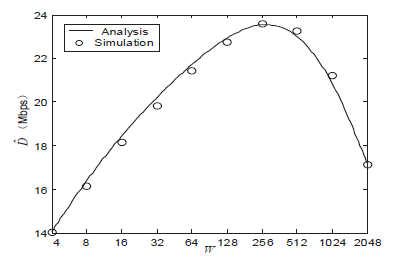


图 5-3 基本接入机制中吞吐量和初始退避窗口大小的关系

3）最大退避等级对吞吐量的影响

设置初始退避窗口大小始终为3，依次设置最大退避等级K设置为2、3、4、5、6、7、8、9、10；即对应设置最大退避窗口大小为15、31、63、127、255、511、1023、2047、4095；具体实施如下：

Config::SetDefault ("ns3::DcaTxop::MinCw", UintegerValue (3));//设置初始退避窗口大小始终为3

将最大初始退避窗口依次设置为15、31、63、127、255、511、1023、2047、4095；输出对应仿真的吞吐量，具体实施如下：

Config::SetDefault ("ns3::DcaTxop::MaxCw", UintegerValue (15)); //设置最大退避窗口大小

利用matlab作图来分析吞吐量和最大退避等级之间的关系，实验结果如下

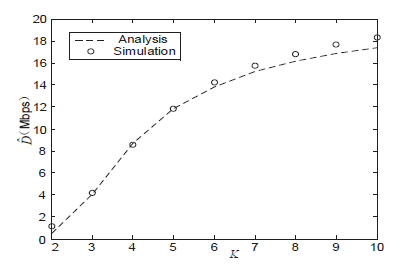


图 5-4 基本接入机制下网络总速率与最大退避等级的关系

1. **研究无线局域网中基本接入机制和RTS/CTS机制下的网络吞吐量的区别**

设置节点数目为50，初始退避窗口为15，最大退避等级为6，数据帧的发送速率为54Mbps，控制帧的发送速率为6Mbps，改变数据帧的长度，切换网络接入模式，谈论在两种接入机制下的网络吞吐量的差异。

1）基本接入机制

将RTS阈值设为一个大于数据包长的值，这时，网络工作在基本接入机制；

Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", StringValue ("999999")); //设置RTS阈值大小使网络运行在基本接入机制

将数据帧长度依次设置为250、500、750、1000、1250、1500、1750、2000、2250、2500；输出对应仿真的吞吐量，具体实施如下：

uint32\_t pktSize = 250;//设置数据帧的长度为250字节

或者在终端修改节点数目，命令如下：

sudo ./waf --run “scratch/adhoc --pktSize=250”

2）RTS/CTS机制

将RTS阈值设为一个小于数据包长的值，这时，网络工作在RTS/CTS机制；

Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", StringValue ("0")); //设置RTS阈值大小使网络运行RTS/CTS机制

将数据帧长度依次设置为250、500、750、1000、1250、1500、1750、2000、2250、2500；输出对应仿真的吞吐量，具体实施如下：

uint32\_t pktSize = 250;//设置数据帧的长度为250字节

或者在终端修改节点数目，命令如下：

sudo ./waf --run “scratch/adhoc --pktSize=250”

3）比较基本接入机制和RTS/CTS机制

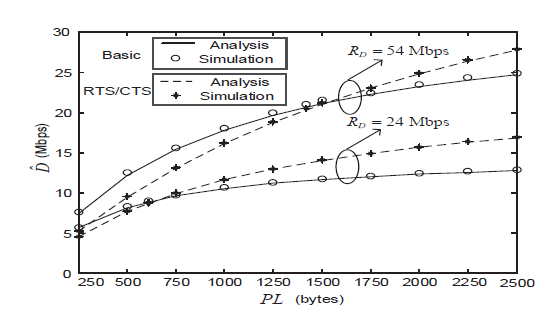


图 5-5 基本接入机制下吞吐量和网络总输入率的关系

1. **研究无线局域网中不同网络拓扑对网络总速率的影响**
2. 自组织网络与基础设施网络

无线自组织网络是分散式无线网络。该网络不依赖于预先存在的基础设施，例如有线网络中的路由器或基础设施无线网络中的接入点。相反，每个节点都是发送者和接收者。 它可以将数据包传输到其他节点，并从发送节点接收数据包。但是，基础设施网络通常基于预先设定的网络设施。因此，节点必须首先与接入点（AP）通信以访问网络。与自组织网络相比，由于添加了关联（信标传输，主动扫描等）引起的附加信道活动，这可能降低网络总速率。

1. 两种网络拓扑下的网络总速率

使用脚本1(adhoc.cc)和脚本2(infrastructure.cc)比较当节点数目不同时，自组织网络与基础设施网络的网络吞吐量。

设置节点数目依次为5、10、15、20、25、30、35、40、45、50；得到对应的吞吐量并比较：

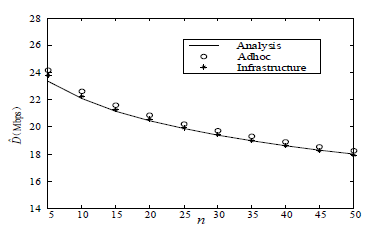


图 5-6 基本接入机制下吞吐量和网络总输入率的关系

# 思考问题

**1.NS-3的仿真结果与理论模型的差异的主要原因有哪些？**

**2.如何减小自组织网络与基础设置网络在网络总速率之间的差距？**