

计 算 机 网 络 实 验

实 验 指 导 书

|  |  |
| --- | --- |
| 实验名称 | 基于NS-3的IEEE 802.11 DCF协议仿真实验 |

电子信息与通信学院 2018年制

# 实验目的

1．了解网络仿真的意义

2．掌握CSMA、DCF协议的基本原理

3．安装并熟悉使用NS-3

4．学会利用NS-3对DCF协议进行仿真

# 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。

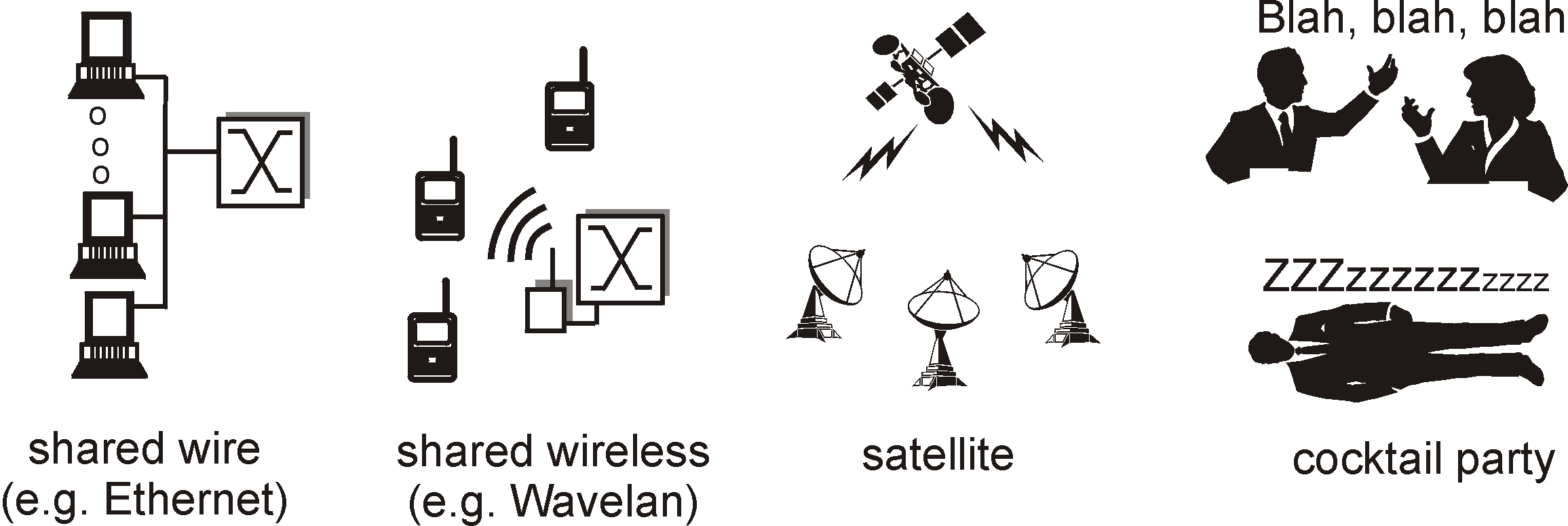
NS-3不是NS-2的扩展，而是一个全新的网络模拟器，是由美国华盛顿大学的Thomas R.Henderson教授及其研究小组在美国自然科学基金（NFS）的支持下，于2006年开始应用现代网络模拟技术和软件开发技术设计并开发的一个全新网络模拟工具。NS-3广泛汲取了现有优秀开源网络模拟器如NS-2、GTNetS、Yans等的成功技术和经验，专门用于教育和研究用途的离散事件模拟器，它基于GNU GPLv2许可，可以免费地获取、使用和修改。

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>。

# 实验原理

本实验则是通过利用NS-3仿真模拟器，完成IEEE 802.11 DCF协议的仿真，在了解了NS-3背景之后，接下来对本实验所仿真的无线局域网协议进行介绍。802.11协议是1997年IEEE发布的第一个在国际上被认可的无线局域网领域的协议，该标准在最开始的时候主要用于办公室和校园，随着技术的发展和无线网络的普及，IEEE对该协议进行了补充，推出了一系列的协议。

1. **多址接入问题**



**图1. 各种多址访问信道**

在我们生活中常见的各种通信网络如以太网、无线局域网、蜂窝网等中，通常有多个用户均接入到同一个共享的广播信道中进行数据传输。在这类网络场景中，设计者面临的一个重要问题是：如何在多个用户之间协调它们对信道的接入? ——这就是所谓的“多址接入”问题（Multiple Access）。

在网络中解决多址接入问题而制定相关协议的为介质访问控制层（MAC）。MAC层位于OSI七层协议中的数据链路层，数据链路层分为上层LLC（逻辑链路控制）和下层的MAC（介质访问控制），MAC主要负责控制与连接物理层的物理介质。在发送数据的时候，MAC协议可以事先判断是否可以发送数据，如果可以发送将给数据加上一些控制信息，最终将数据以及控制信息以规定的格式发送到物理层；在接收数据的时候，MAC协议首先判断输入的信息是否发送传输错误，如果没有错误，则去掉控制信息发送至LLC（逻辑链路控制）层。

MAC协议按照协议部署方式分可以分为集中式MAC协议和分布式MAC协议两类。集中式协议将繁重的计算工作交由中心节点执行，效率高，因为中心节点位于WLAN的中心，所有的无线节点都能够与之传输信息，因此不存在隐藏节点和暴露节点的问题。但由于所有的无线节点都能够与之传输信息，在网络规模较大的情况下，对于中心节点的处理能力和处理时延都有较高的要求。比如空闲监听多重接入（Idle Sense Multiple Access, ISMA）、资源竞买多重接入（Resource Auction Multiple Access, RAMA）和可丢弃令牌MAC协议（Disposable Token MAC Protocol, DTMP）都属于集中式MAC协议。

分布式MAC协议中，节点相互竞争对媒介的使用权。当只有一个节点试图发送信息时，数据包能够成功传送出去。当同时有几个节点试图发送数据，就会产生冲突。除了ALOHA协议之外，所有的分布式MAC协议都是基于载波侦听和冲突避免机制的。由于载波侦听的位置相关性，隐藏节点和暴露节点这两种现象在CSMA协议中是频繁出现的。发送节点有可能发现不到在接收节点处发生的冲突，因此接收节点需要将冲突信息传送给发送节点。但是由于无线系统是工作在半双工模式下的，发送节点在进行传输的同时是接收不到接收节点所发出的冲突信息的，因此当冲突过多时会产生很大的开销。典型的分布式MAC协议就有CSMA、DCF协议等。

**（二）CSMA协议**

载波侦听多路访问（Carrier Sense Multiple Access, CSMA）起源于美国夏威夷大学开发的ALOHA网所采用的争用型协议，并进行了改进，使之具有比ALOHA协议更高的介质利用率。在CSMA协议中，每个用户节点通过侦听信道来避免冲突。载波侦听指的是节点在发送数据包之前必须先监听信道状态，若信道空闲则发送数据，若信道忙则推迟发送。用户节点在传输数据包的过程中仍然对信道进行监听，若检测到信道中发生了冲突，则立即停止传输，然后重新对数据包的传输进行规划，其传输时信道的状态如图3-1。

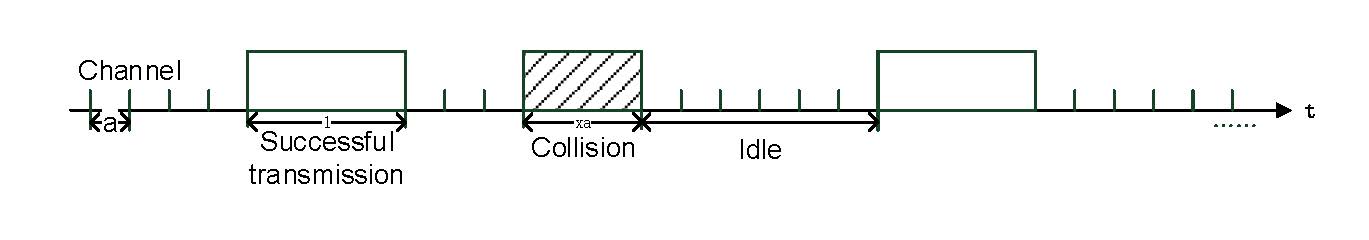


图 3-1 CSMA网络信道状态

CSMA协议可以具体分为non-persistent、1-persistent、p-persistent三种类型。其中，non-persistent指的是节点在监听到数据包空闲时则立即发送数据，若监听到信道忙，则随机退避一段时间再对信道进行监听。1-persistent指的是节点在监听到信道空闲时立即发送数据，若监听到信道忙则持续监听，直到信道变为空闲时则立即发送数据。P-persistent指的是节点监听到信道空闲时，以p概率发送信道，以1-p概率随机退避一段时间再对信道进行监听，若监听到信道忙，则节点仍持续监听信道至空闲。

尽管CSMA可以提高信道的利用率，但是如果节点要发送的数据包很长时，若有多个节点同时传输仍可能造成信道资源的浪费。并且由于无线局域网半双工的特点，无法做到在传输数据包的同时侦听信道，所以在无线局域网中所采用的是CSMA/CA协议。该协议在节点发送数据包之前，先向目的节点发送一个较小的RTS帧预约信道，目的节点回复一个CTS帧后节点开始传输数据包，这样的方式在数据包较长使可以在一定程度上提高信道的利用率。

**（三）DCF协议**

分布式协调功能（Distributed Coordination Function, DCF）是IEEE 802.11 MAC层的协议，定义了无线局域网中最基本的信道共享的机制。DCF协议是基于CSMA 协议的，拥有两种接入机制，包括基本接入机制（basic access mechanism）和RTS/CTS接入机制（RTS/CTS mechanism）。

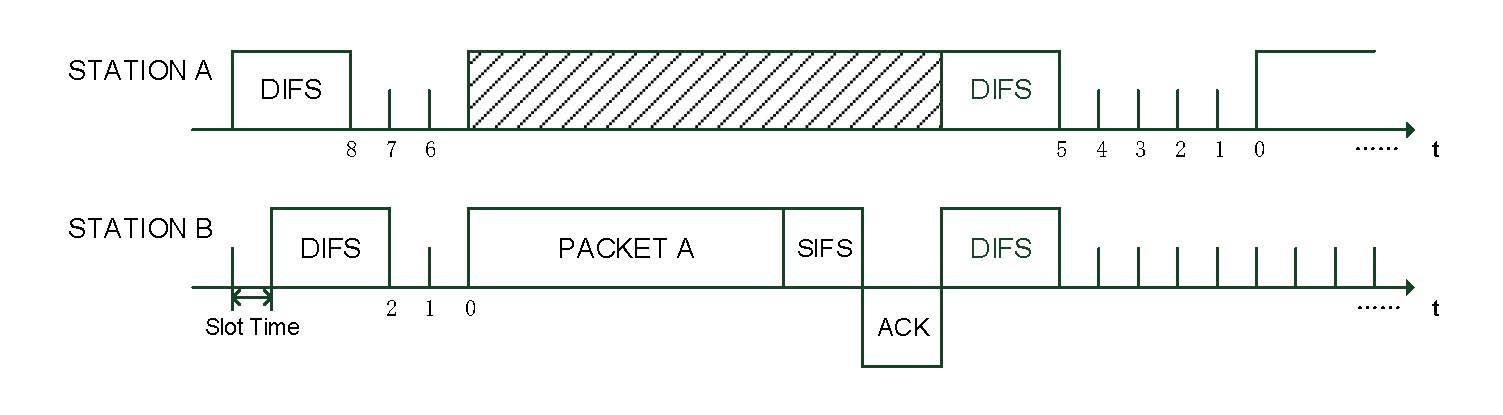


图 3-2 基本接入机制

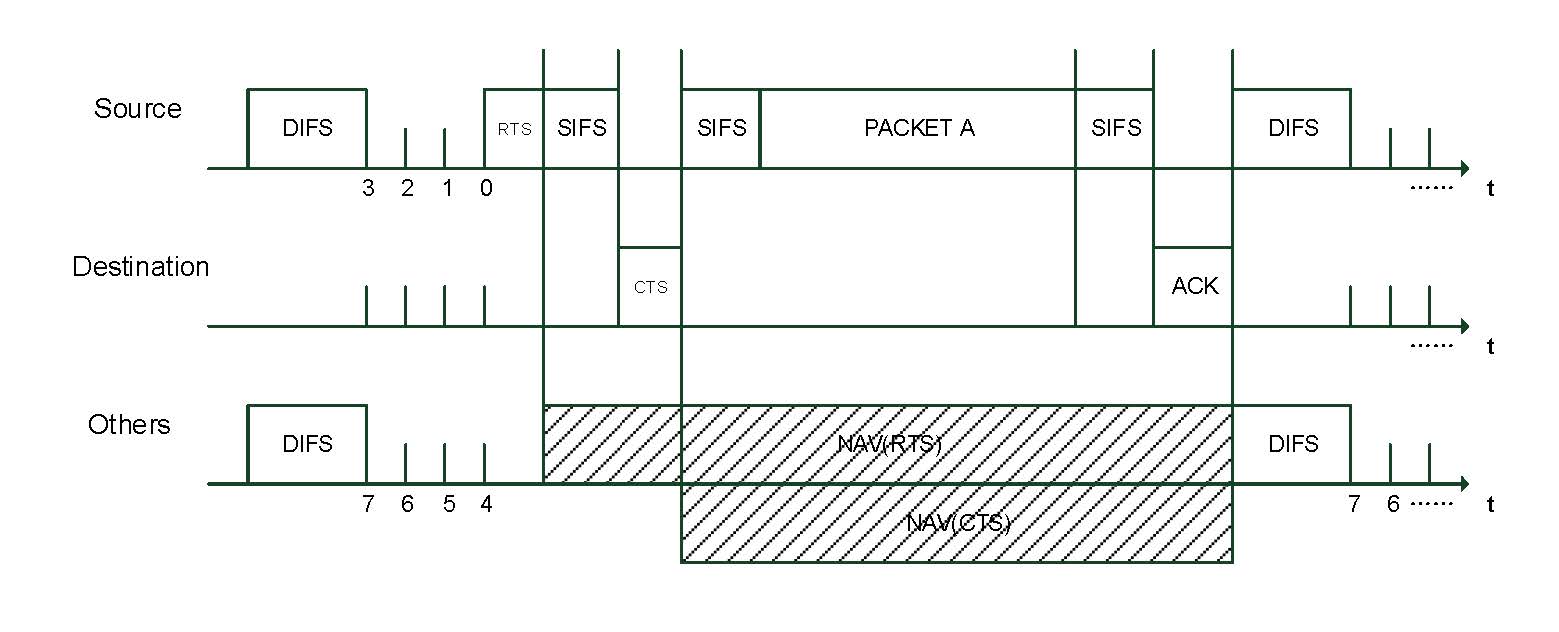


图 3-3 RTS/CTS接入机制

基本接入机制与CSMA 协议原理相同，是一种二次握手的技术。如图3-2，DCF中存在两种帧间间隔，一种是DCF帧间间隔（DIFS）和最短帧间间隔（SIFS）。当节点侦听到信道在一个DIFS内为空闲时，则认为信道空闲，开始传输数据，若发生冲突产生退避，也是侦听到信道一个DIFS空闲后才开始进行退避。SIFS是目的节点收到传输的数据后，发送给传输节点一个确认帧（ACK）之间的时间间隔。RTS/CTS接入机制则是CSMA/CA协议中的原理，由于无线局域网可能存在隐藏终端（Hidden Node）的问题，节点无法侦听到所有与目的节点相连接的节点是否有数据要与目的节点传输。所以引入了RTS/CTS这两个短帧，节点在传输数据包之前，先等待信道空闲一个DIFS后，向目的节点发送一个RTS（Request To Send）帧，当目的节点确认没有其余节点在进行传输后回返回一个CTS（Clear To Send）帧，表示传输节点预约信道成功，接着开始进行数据包传输。帧与帧之间的最小间隔仍为SIFS，如图3-3。

这两种机制在不同的网络状态下有各自的优点，当网络中传输的数据包长度较长的时候，使用RTS/CTS机制，可以避免产生冲突时过长的数据包占用信道的时间太久。但当数据包长度较短时使用RTS/CTS帧则容易产生过多的冗余，此时则更适合使用基本接入机制。因此需要设置一个数据包的RTS/CTS阈值，当网络中传输的数据包长度低于该阈值时，采用基本接入机制，当长度高于阈值时采用RTS/CTS机制，通过这样灵活的调整网络中的传输机制可以实现网络性能的最优化。

尽管IEEE 802.11无线局域网能在全球范围内普及的主要原因归功于DCF协议，其继承了随机接入协议简单性和健壮性的优点。但是在DCF协议中所有的节点不仅采用相同的侦听机制，其基本退避参数也相同，所有的节点在网络处于饱和状态时都拥有相同的吞吐量，它能够提供的仅仅是best-effort服务。为了满足现在多媒体应用对网络服务质量需求的飙升，增强分布式信道访问（Enhanced Distributed Channel Access, EDCA）作为DCF协议的一种增强版本被提出了，用来提供服务质量的需求。本实验中主要还是通过NS-3完成对IEEE 802.11 DCF协议的仿真，EDCA协议的相关内容如何用NS-3进行仿真可以留给学生课下拓展思考。

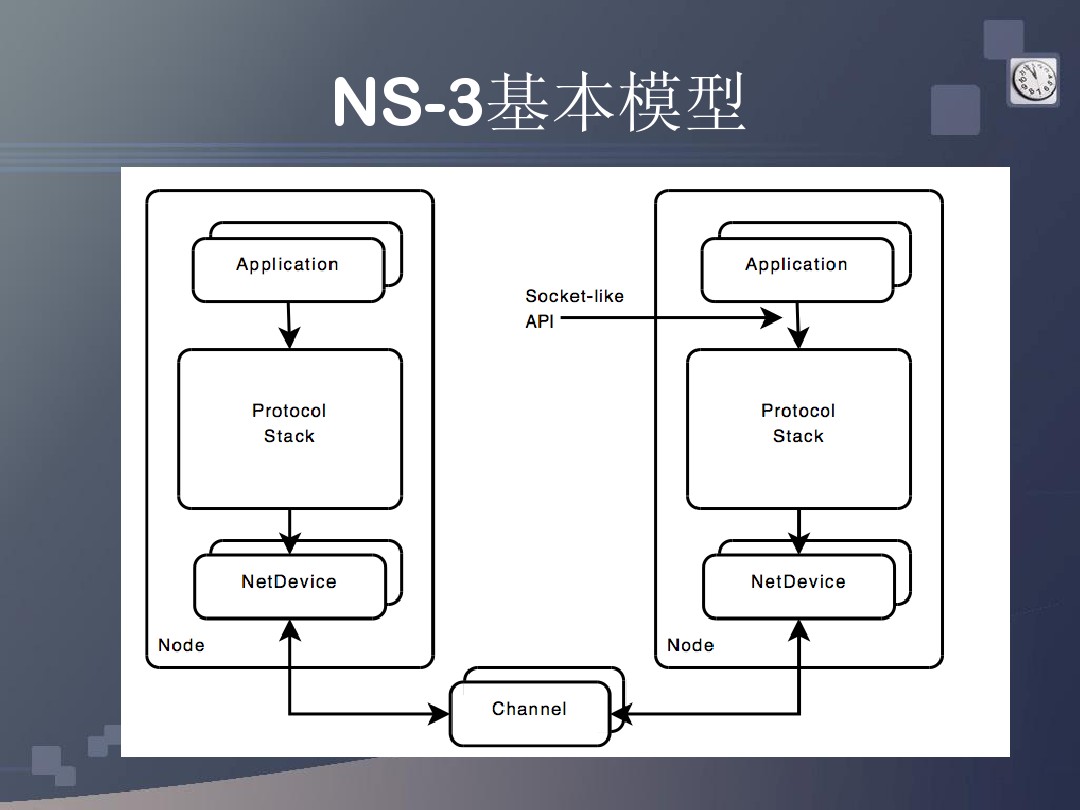


图 3-4 NS-3基本模型

**（四） 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。

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）代码示例**

本实验是用NS-3仿真802.11a无线局域网中的DCF协议，通过调整网络中的参数（如网络节点数目***n***、数据包发送速率***DR***等）来输出网络总吞吐量，比较无线网络中的网络参数和吞吐量的关系，以及DCF协议中基本接入机制和RTS/CTS机制随着不同的参数吞吐量的变化幅度趋势也各有不同，并分析其原因。

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：

a.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 wifiStaNode；//声明一个名为“wifiStaNode”的NodeContainer

nodes.Create(nWifi); //调用nodes对象的Create()方法创建nWifi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nWifi个节点对象，并把指向这nWifi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与建立的作为AP的节点连接成网络。

b. 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：UdpEchoClientApplication 和UdpEchoServerApplication。

uint16\_t port = 8000;

UdpServerHelper server (port); //AP为服务端

ApplicationContainer serverApp;

serverApp = server.Install (wifiApNode.Get (0));

serverApp.Start (Seconds (0.0));

serverApp.Stop (Seconds (simulationTime + 1));

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。UdpEchoClientApplication用来回显服务器端所回复的分组。同其他生成器对象类似，UdpEchoServerApplication由UdpEchoServerHelper管理。

c. 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 ());

d. 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，就像示例代码中分别有staDevice和apDevice两个被创建和保存在设备容器内，一个是用于station的，一个是用于ap的。

NetDeviceContainer staDevice;//网络设备容器

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

mac.SetType ("ns3::ApWifiMac","Ssid", SsidValue (ssid));

NetDeviceContainer apDevice;

apDevice = wifi.Install (phy, mac, wifiApNode); //AP配置物理层、MAC层等置于网络设备中

# 实验内容

本实验是用NS-3仿真802.11a无线局域网中的DCF协议，通过仿真实验观察网络性能（吞吐量、时延）与网络关键参数（节点数量、初始退避窗口、最大退避次数等）之间的定量关系。

**（一）无线局域网的网络吞吐量与用户流量到达率的关系**

本实验主要研究的是用户流量到达率对网络状态的影响，通过网络吞吐量可以观察出网络饱和状态的变化。

根据网络中每个节点处数据包发送队列的状态，网络可以分为饱和和不饱和两种状态。若网络中每个数据包发送队列的长度是有限的，即数据包等待发送的时延是有限的情况下，网络此时的状态是非饱和的，该状态下网络中的吞吐量与网络中的总输入是持平的。若网络中每个数据包发送队列的长队很大，趋近于无限，此时数据包等待发送的时间也是无限的，那么此时网络的吞吐量将达到一定的值，网络状态达到饱和。

调整用户流量到达率，在源代码中相关的代码为：

client.SetAttribute ("Interval", TimeValue (Time ("0.001"))); //数据包到达间隔

通过改变每个节点数据包的到达间隔，可以调整用户流量到达率。可以从小到大改变其值，观察输出吞吐量的变化，图例如下所示。

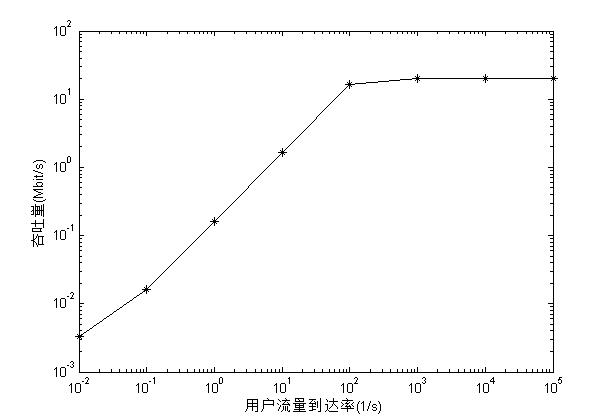


图4-1 用户流量到达率对网络吞吐量的影响

通过上图可以观察到，随着用户流量到达率的增加，网络的吞吐量也随之增加，当增加到一定值时开始不再变化。这是由于在用户流量到达率较小时，网络处于非饱和的状态，吞吐量会随着用户流量到达率的增加而增加，当用户流量到达率增加到一个临界值的时候，此时网络状态由不饱和变为饱和，网络吞吐量到达最大值，不会再随着用户流量到达率的增加而改变。值得说明的是在源代码中我们通过改变数据包的到达间隔来改变用户流量到达率，而到达间隔和到达率很明显是一个成反比的关系，因此上图中的横坐标可以用数据包到达间隔的倒数来代表用户流量的到达率。

**（二）无线局域网的网络吞吐量与网络用户数量的关系**

设置饱和网络场景，数据包长=1900bytes、初始退避窗口大小CWmin=16、最大退避次数K= log(CWmax/CWmin)=6、数据包发送速率DR=54Mbps和控制帧发送速率BR=6Mbps。

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

修改网络用户数量（5、10、15、20、25、30、35、40、45、50），分别进行仿真实验，输出对应网络场景的吞吐量；

uint32\_t nWifi = 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/test --nWifi=5”

记录不同用户数量场景下仿真所得的网络吞吐量，并在matlab等工具中作图获得二者之间的定量关系图。

**思考：**

1. 根据所得仿真结果，说明所观察到的现象；

2. 根据你对DCF协议的理解，解释所观察到的现象。

**（三）无线局域网的网络吞吐量与初始退避窗口的关系**

设置饱和网络场景，数据包长=1900bytes、用户数量为20个、最大退避次数K=log(CWmax/CWmin)=6、数据包发送速率DR=54Mbps和控制帧发送速率BR=6Mbps。

将初始退避窗口CWmin依次设置为3、7、15、31、63、127、255、511、1023；对应的CWmax依次设置为255、511、1023、2047、4095、8191、16383、32767、65535；具体实施如下：

在ns3的src文件下wifi模块中找到regular-wifi-mac.cc；路径为

ns-3-dev/src/wifi/model/regular-wifi-mac.cc

在第1277行和1288行分别修改CWmin和CWmax，如：cwmin=15; cwmax=1023; 改为：

cwmin=3;

cwmax=255;

记录不同初始退避窗口场景下仿真所得的网络吞吐量，并在matlab等工具中作图获得二者之间的定量关系图。

**思考：**

1. 根据所得仿真结果，说明所观察到的现象；

2. 根据你对DCF协议的理解，解释所观察到的现象；

3. 记录网络吞吐量达到最大时对应的最优初始退避窗口值，试试看改变用户数量后该最优值的变化情况，解释原因。

**（四）无线局域网的网络吞吐量与最大退避次数的关系**

设置饱和网络场景，数据包长=1900bytes、用户数量为20个、初始退避窗口大小CWmin=16、数据包发送速率DR=54Mbps和控制帧发送速率BR=6Mbps。

将最大退避次数K从0依次增加，即将最大退避窗口CWmax=CWmin\*2^K依次增加；具体实施如下：

在ns3的src文件下wifi模块中找到regular-wifi-mac.cc；路径为

ns-3-dev/src/wifi/model/regular-wifi-mac.cc

在第1288行修改CWmax，保持CWmin =15，修改CWmax；改为：

cwmin=15;

cwmax=1023;

记录不同最大退避次数场景下仿真所得的网络吞吐量，并在matlab等工具中作图获得二者之间的定量关系图。

**思考：**

1. 根据所得仿真结果，说明所观察到的现象；

2. 根据你对DCF协议的理解，解释所观察到的现象；

3. 比较初始退避窗口与最大退避次数对网络吞吐量影响的差异，谈谈你对二者为何存在这样差异的理解。

# 思考问题

1. **除以上关键参数外，还有哪些参数可能会影响无线局域网的性能？**
2. **IEEE 802.11 DCF协议中的基本接入机制和RTS/CTS接入机制的主要区别？**
3. **能否分别获得DCF协议中使用不同接入机制（基本接入机制和RTS/CTS接入机制）时的网络性能，比较其差异，并分析在实际无线局域网中应当如何选择接入机制？**