网络建模仿真工具

目录

> 基于C++的网络建模仿真 <

1.1、C/C++仿真简介

1.2、C/C++建模仿真例子

1.3、C/C++建模仿真缺点

C/C++仿真简介

- 在信息与通信领域，C/C++是使用最为广泛的**高级语言**
- 它们使用灵活，执行效率较高，有着良好的**仿真速度**

- 高级语言编译器一般只提供少量通用函数，针对性不强。
- 仿真模型中的大部分模块需要从最底层开始编写，要求仿真人员对**每一步实现的原理**都了如指掌

> 基于C++的网络建模仿真 <

1.1、C/C++仿真简介

1.2、C/C++建模仿真例子

1.3、C/C++建模仿真缺点

C/C++建模仿真例子

- 均匀分布
- 瑞利分布
- 高斯分布

用C语言采用**Box-Muller**法构造AWGN信道模型本质：产生相互独立的高斯随机变量组成的一个序列

1. 产生两个（0,1）区间内的均匀分布随机数
2. 使用其中一个产生瑞利分布随机数
3. 由瑞利分布随机数、均匀分布随机数通过**变换得到高斯分布的随机数**

均匀分布随机数的产生

- 均匀分布随机数的产生

  采用常见的线性同余法

  - 目前应用广泛的伪随机数生成算法
  - 基本思想：通过对前一个数进行线性运算并取模，从而得到下一个数

$$x_{n+1} = (ax_n + c) \quad mod(m)$$

$$y_{n+1} = x_{n+1}/m$$

由uniform01函数实现(0,1)区间上均匀分布随机数的产生：

```
double uniform01(int *seed)
{   /*Generate a random number between 0 and 1*/
        double t;
        *seed = 2045*(*seed)+1;
        *seed = *seed - (*seed/1048576)*1048576;
        t = (double)((*seed)/1048576.0);
        return t;
}
```

瑞利分布随机数的产生

瑞利分布随机数在(0,1)区间上均匀分布随机数的基础上由反变换法得到，由以下函数实现：

```
double Rayleigh(double sigma, int *seed)
{
        return (double)sqrt(-2*sigma*sigma*log(1-uniform01(seed)));
}
```

高斯分布随机数的产生

高斯分布随机变量的产生由以下函数实现。该函数可以得到长度为Ns、均值为0，标准差为sigma的高斯分布随机数。

```
void gauss(double mean,double sigma,int Ns, double *gauss)
{           int i;
            double xita;
            double z;
            int seed = rand();
            for(i=0;i<Ns;i++)
            {
              z=Rayleigh(sigma, &seed);
              xita=uniform01(&seed);
             gauss[i]=(double)(mean+z*cos(2*3.14159265*xita));
            }
}
```

> 基于C++的网络建模仿真 <

1.1、C/C++仿真简介

1.2、C/C++建模仿真例子

1.3、C/C++建模仿真缺点

C/C++仿真建模缺点

➢ **开发工作量较大：**

- C/C++作为仿真工具往往需要**从底层开始逐步搭建仿真模型**

➢ 但是学术界已经针对信息与通信领域的仿真应用，合作开发了一套基于C++的**开源程序包IT++**。这套程序包包含了数学、信号处理和通信有关的类和函数，方便二次开发。

IT++：http://itpp.sourceforge.net/4.3.1/index.html

目录

> 基于OMNET++的网络建模仿真 <

2.1、OMNET++简介

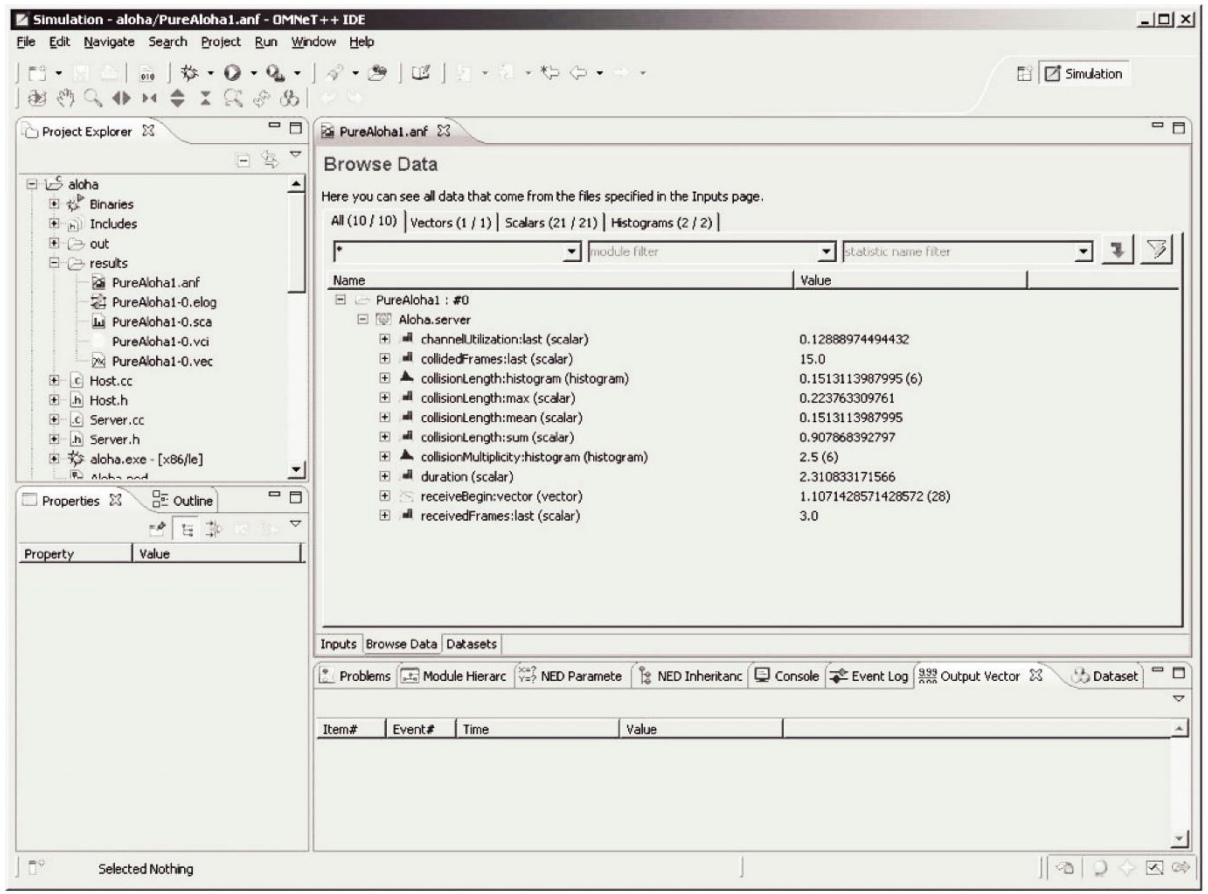
2.2、OMNET++发展

2.3、OMNET++优缺点

2.4、OMNET++实现

OMNET++:

- 一个可扩展、模块化、基于组件的C++仿真库和框架，主要用于构建网络模拟器
- 可模拟有线和无线通信网络、片上网络、排队网络等多种情况
- 提供基于Eclipse的IDE，图形运行时环境和许多其他工具
- 有实时模拟、网络仿真、数据库集成、SystemC集成和其他一些功能的扩展



基于OMNET++的网络建模仿真

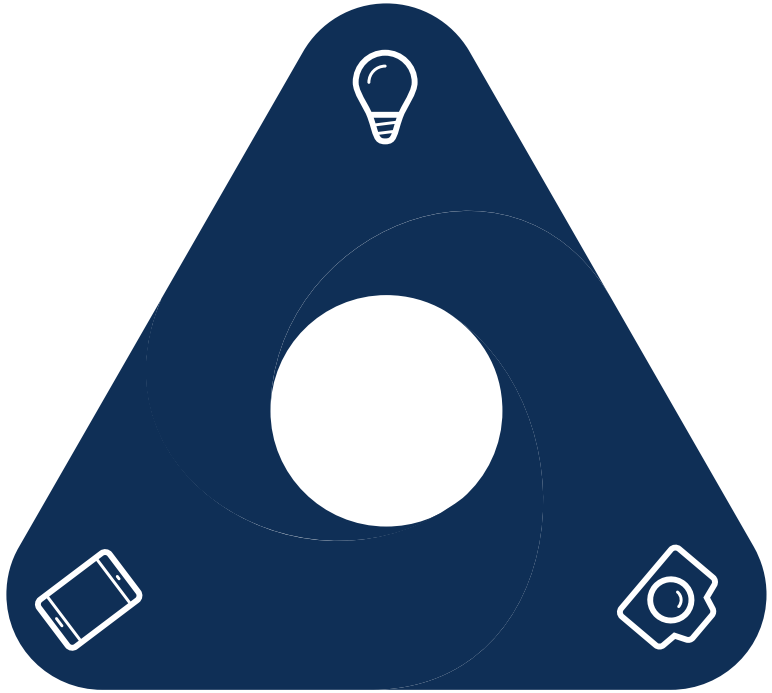
2.1、OMNET++简介

2.2、OMNET++发展

2.3、OMNET++优缺点

2.4、OMNET++实现

OMNET++发展



起源

- 由布达佩斯技术大学的Andras Varga等人设计的一种内核源代码完全开放的仿真软件
- 可支持Linux、Windows及DOS等多个平台

近几年来发展

- 添加多种**仿真模型和模型框架**：排队队列、资源建模、互联网协议、无线网络、局域网、应用程序等
- 大多数模型框架都是**开源的**，作为独立项目开发，并遵循自己的发布周期

> 基于OMNET++的网络建模仿真 <

2.1、OMNET++简介

2.2、OMNET++发展

2.3、OMNET++优缺点

2.4、OMNET++实现

OMNET++优点

- 使用C++语言进行仿真
- 提供图形用户界面，可以动态观察仿真程序的运行
- 使用参量的方式, 可以不修改源代码和重新编译便对不同网络模型仿真
- 可以使用非常简易的NED语言来代替C++完成对网络拓扑的描述
- 可以使用变量观察函数及绘图功能将考察的变量实时地绘成曲线
- 对硬件系统要求不高，运行速度较快，相当于用C编写的1.3倍

OMNET++缺点

- 入门门槛较高，需要扎实的C++语言理解和网络基础知识
- OMNET++软件的使用教程较少,实际问题只能去google group等地方查阅（开源软件的通病）
- 软件模块的学习周期长，入门项目学习平均时间为三个月

➢基于OMNET++的网络建模仿真≺

2.1、OMNET++简介

2.2、OMNET++发展

2.3、OMNET++优缺点

2.4、OMNET++实现

OMNET++仿真设计实现

01

网络拓扑设计

NED描述的文件可包含部分：

- 输入文件（外部定义好的拓扑文件）
- 信道
- 简单和复合模块
- 网络

02

网络运行机制

仿真网络的运行通过发送和接收消息的方式实现

网络拓扑设计—信道

OMNET++中的物理层连接通过信道定义描述，仿真中信道主要抽象为三个特征：

- 传播延时（delay）
- 误码率（error）
- 数据传输速率（datarate）

channel channelname
delay 0.005
error 0.0001
datarate 14400
endchannel

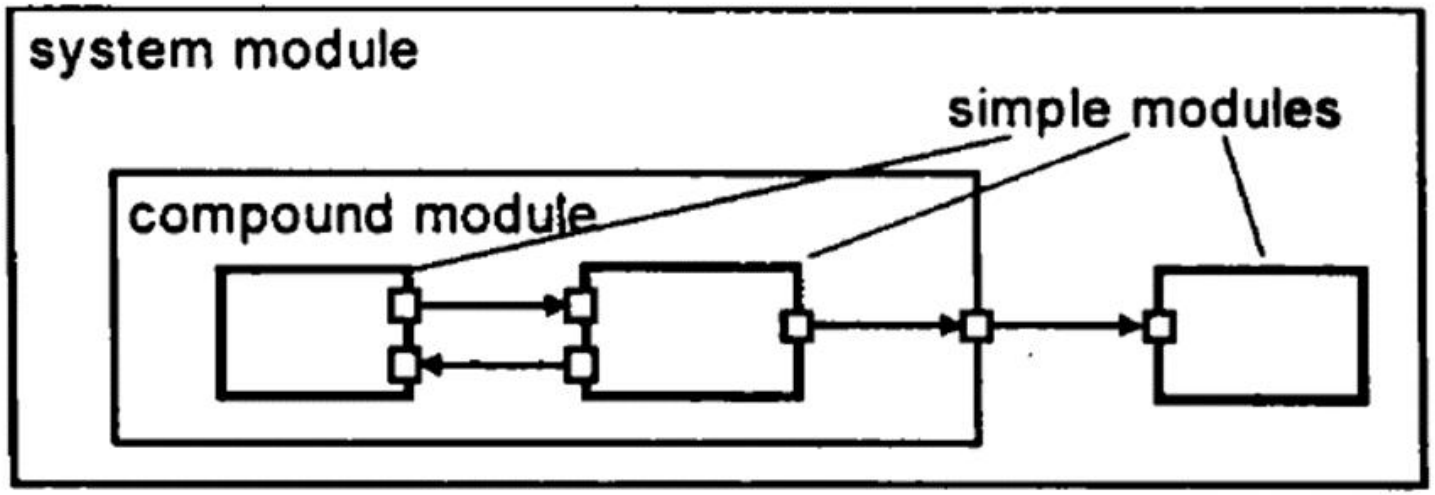
网络拓扑设计—简单模块

简单模块是网络中的基本组成部分，它通过参量表达真实网络的特征，可以被复合模块嵌套。

simple layer0 //简单模块名
gates://参数，在.cc中使用
  in:lowergate_in[];
  in:uppergate_in
  in:blackboard_in
  out:lowergate_out[];
  out:uppergate_out;
end simple

网络拓扑设计—复合模块

复合模块可以嵌套简单模块，无嵌套层数限制。可以将外部的任何网络实体抽象为复合模块。



网络拓扑设计—复合模块

```
module CompoundModule
  parameters: //子模块中使用
    CNNCTVTY: numeric,
    KIND:numeric,
    COLOR:numeric,
    PX:numeric
    PY:numeric;
  gates: //外部接口
    in:  in[]
    out: out[]
submodules: //子模块列表
  sm_layer0: layer0 //子模块sm_lay0
    gatessizes:
      lowergate_in[CNNCTVTY],
      lowergate_out[CNNCTVTY];
    display:"p=158,86;i=layer0"
sm_application:application; //子模块sm_application
    display:"p=158,86;i=application";
  connections: nocheck//子模块gates间连接关系
  for i=0,CNNCTVTY-1 do
 //该模块输入、输出端分别与sm_layer0模块输入、输出端相连
      in[i]→sm_layer0.lowergate_in[i]; //该模块输入端
      out[i]←sm_layer0.lowergate_out[i];
  endfor;
 //am_application子模块与sm_layer0连接关系
sm_application.lowergate_in←sm_layer0.uppergate_out;

sm_application.lowergate_out→sm_layer0.uppergate_in;
  display:"b=290,485,rect;o=white";
endmodule
```

网络拓扑设计—网络定义

网络包括一个或多个模块，各模块之间通过端口(gate)来连接。网络定义方式与复合模块类似。网络中端口有两种类型：

- 输入端口：负责接收消息
- 输出端口：负责发送消息

Module test

Parameters:

P1:numeric

P2:numeric

Submodule:

Snode1: node //子模块1

…//

Snode2:node//子模块2

…//

Connections nocheck:

Display:“p=0,0;b=860,560,rect;o=white”;

endmudule

Network thetest: test

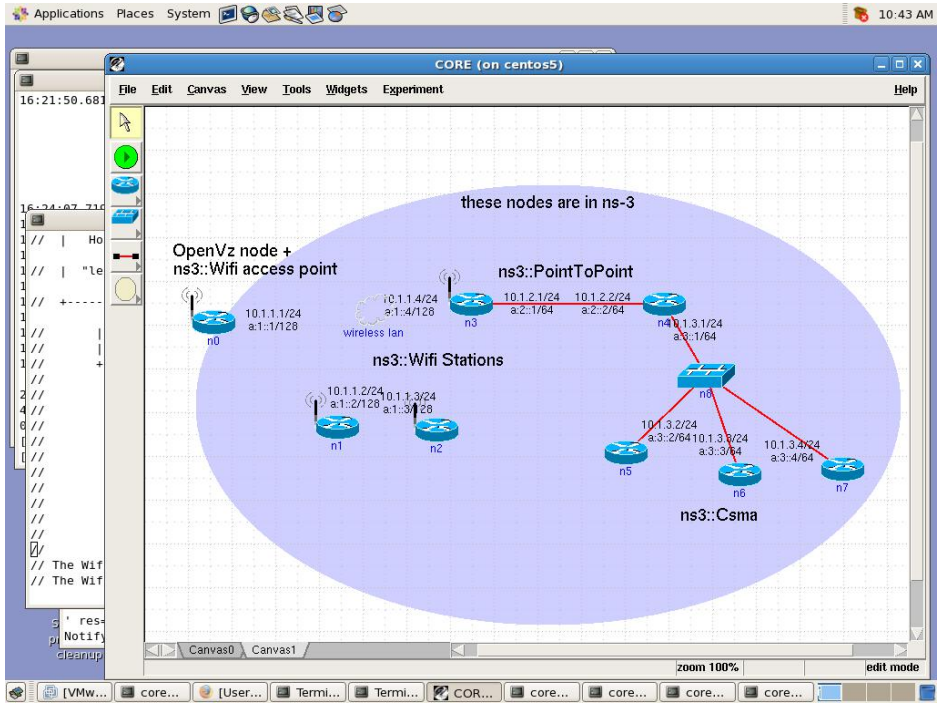
endnetwork

目录

目录

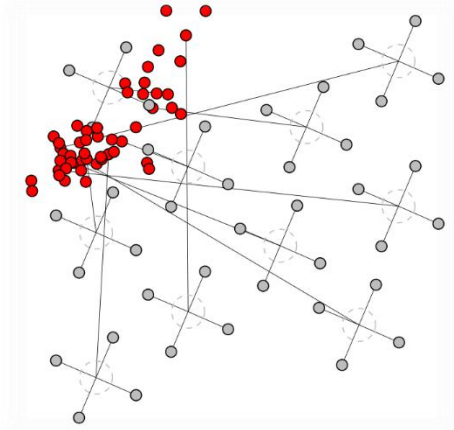
NS-3是一个离散事件网络模拟器

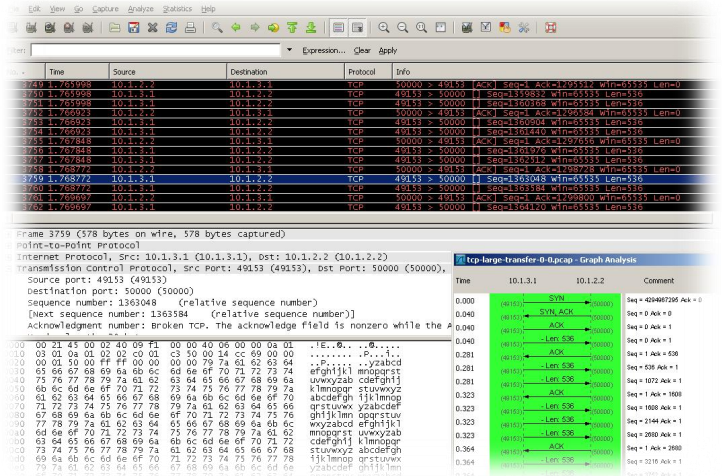
- NS-3是一个全新的模拟器，是由美国华盛顿大学的Thomas R. Henderson教授及其研究小组于2006年开发的一个全新网络模拟工具。



NS-3是一个开源的、免费的软件

- NS-3主要用于研究和教育方面，致力于维护一个开放的环境，供研究人员贡献和分享他们的软件。
- NS-3不仅提供了分组数据网络工作执行的模型，为用户提供仿真实验的模拟引擎，而且可完成一些难以用真实系统执行的研究，其仿真环境可控且可重复，有助于了解网络如何工作。





相关学习资源

01 http://www.nsnam.org：NS-3的主要网站，可以访问有关NS-3系统的基本信息。

02 http://www.nsnam.org/documentation/：NS-3的详细文档，其中包括与系统体系结构相关的文档。

03 http://www.nsnam.org/wiki/：NS-3的维基百科，这里有一些补充的重要信息，这里还能找到用户和开发人员常见问题解答，故障排除指南，第三方贡献代码、论文等。

04 http://code.nsnam.org/：NS-3的源代码。

目录

NS-3的诞生

- ✓ **NS-3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的支持下，于2006年开发的一个全新网络模拟工具**
  - NS-3是**开源的**，致力于构建为能够协同工作的软件库系统，努力为研究者提供一个开放的环境来共享他们自己的软件。
- ✓ **NS-3并不是新一代NS-2**
  - 脚本语言选择不同：

    NS-2使用OTcl脚本语言，仿真的结果可以通过网络动画器nam来演示。
    而在NS-3中，仿真器全都由C++编写，仅仅带有可选择性的Python语言绑定。
  - NS-3并不支持NS-2的API。NS-2中的一些模块已经被移植到了NS-3。
  - NS-3增加新的功能，如：更详细的802.11模块、IP寻址策略的使用。
  - 在NS-3开发的过程中，NS-3项目组会继续维护NS-2，同时也会研究过渡和整合机制。

目录

Mercurial与NS-3

- Mercurial系统基于python实现，是 NS-3 系统选用的源码管理系统。
- Mercurial是一种轻量级分布式版本控制系统，易于学习和使用，扩展性强

Mercurial系统的优点

- 管理轻松，避免复杂流程
- 分布式系统更加可靠
- 较低的网络依赖性，实现离线管理

http://www.selenic.com/mercurial/，获取Mercurial入门指南、最新消息、软件配置等信息。

waf与NS-3

- NS-3的编译系统采用了**waf**：下载NS-3的源码到本地系统之后，需要对源码进行编译来生成可执行程序。正如源码管理方式多种多样，编译源码也有多种工具。

- waf是用Python开发的新一代编译管理系统，读者不必掌握python，即可编译现有的NS-3项目。

- 如果读者想要扩展现有的NS-3系统，大多数情况只需了解Python知识的很少且非常直观的一个子集。

如需了解Waf的细节，可以访问http://code.google.com/p/waf/

目录

相比其他的网络模拟器，NS-3有一些特点和优势：

1. 核心软件的可拓展性
2. 贴合真实系统
3. 软件集成
4. 支持虚拟化和测试平台
5. 灵活的追踪和统计
6. 详细的属性系统

核心软件的可拓展性

- 使用C++编写，提供可选择的python接口
  - 可拓展的文件 API (doxygen):

Doxygen是一种开源跨平台的，以类似JavaDoc风格描述的文档系统，完全支持C、C++、Java、Objective-C和IDL语言，部分支持PHP、C#。注释的语法与Qt-Doc、KDoc和JavaDoc兼容。Doxygen可以从一套归档源文件开始，生成HTML格式的在线类浏览器，或离线的LATEX、RTF参考手册

- 详情可见: https://www.nsnam.org/documentation/

核心软件的可拓展性

- 研究者可提供模型拓展
- 开源软件，方便软件和模型更新
- 实现模型之间的联合使用
- 进行内存管理

贴合真实系统

➢ 一般网络仿真系统的问题：

实验中常常面临同时要模拟和测试的情况，如果模拟器无法近似真实情况将导致：

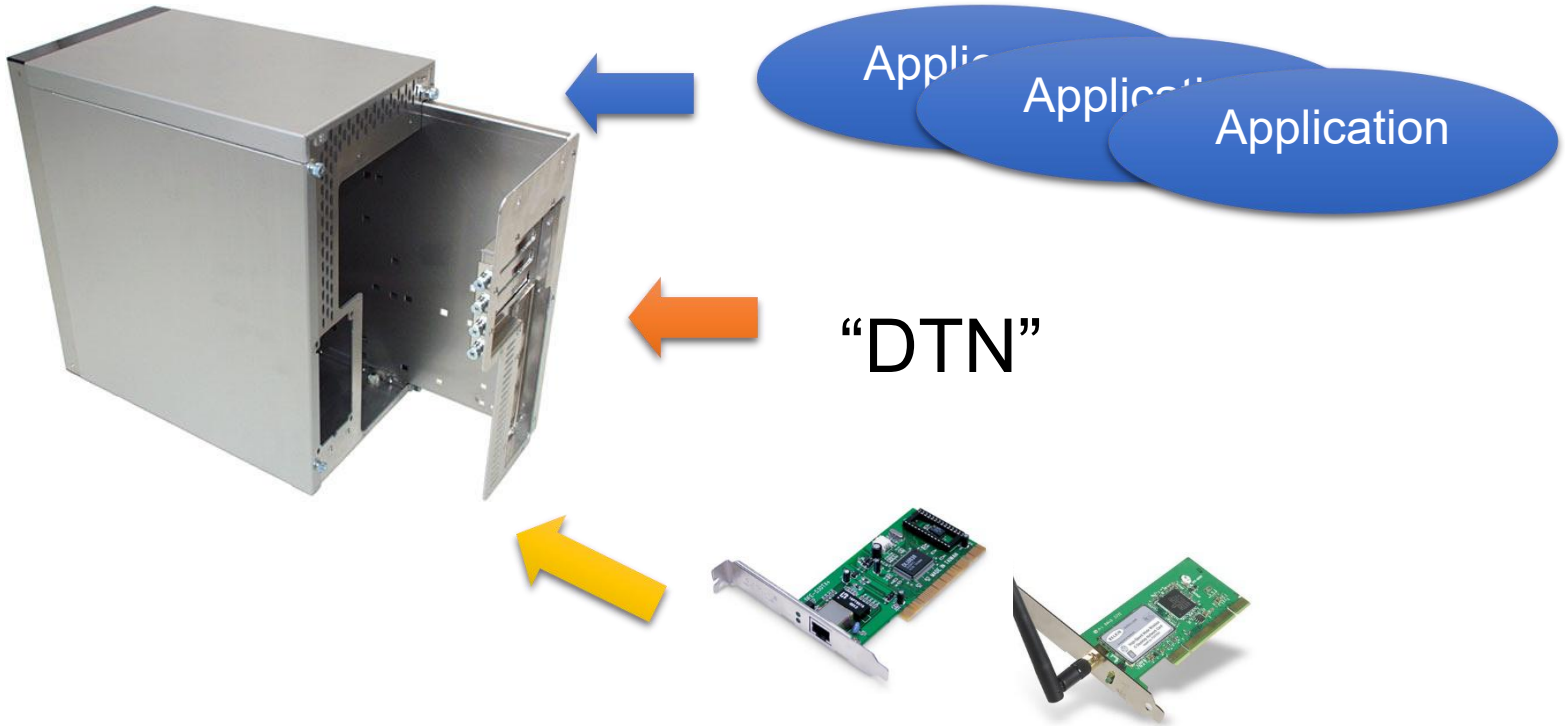
- 难以实现结果对比和模型验证
- 难以在模拟和实际模型中复用软件

➢ ns-3 系统提供的解决办法：

- 提供更像真实计算机的模拟节点
- 支持关键接口，如套接字API和IP/设备驱动程序接口（在Linux中）
- 重用内核和应用程序代码

贴合真实系统

一个NS-3节点是一个添加了应用程序，堆栈和网卡的计算机系统



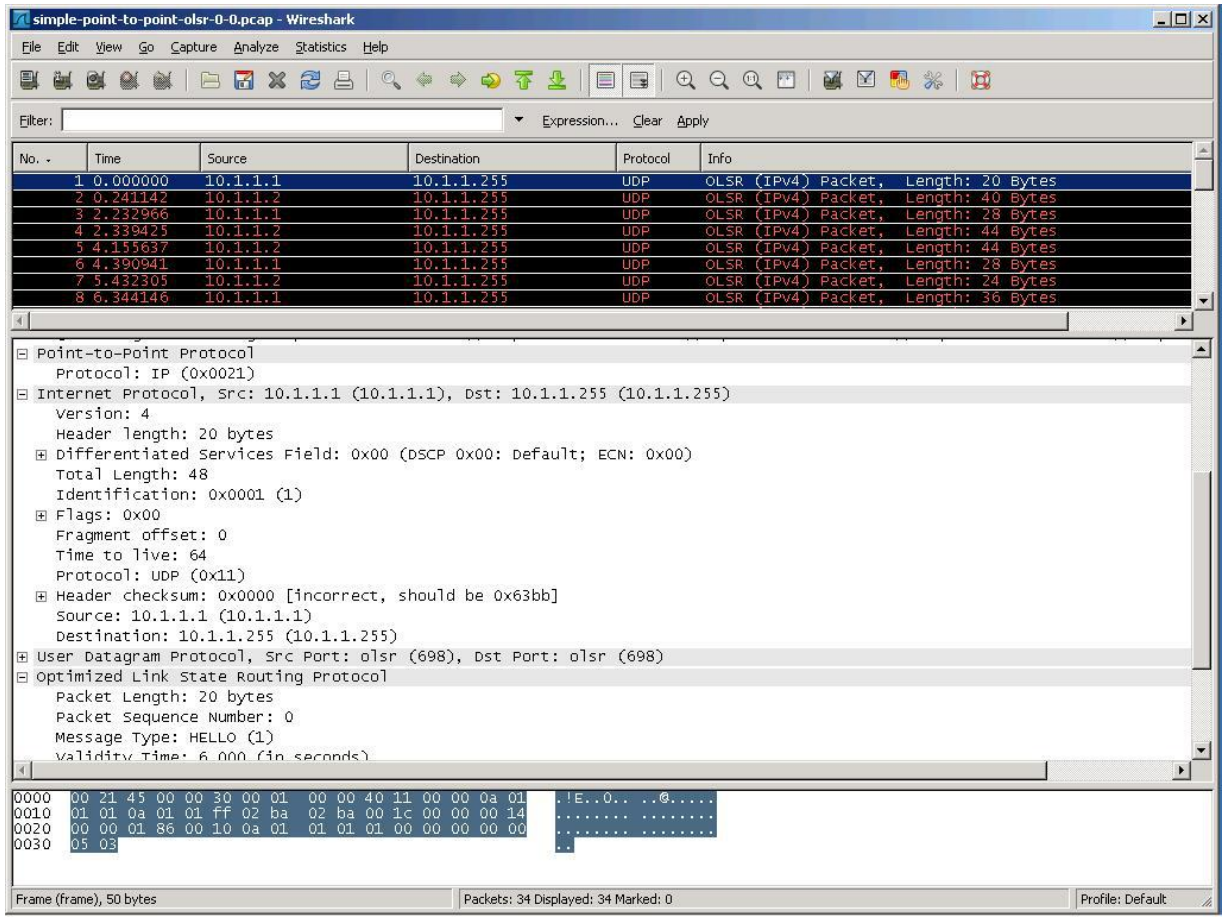
➢ 一般网络仿真系统的问题：

- 帮助实现开源的模型和工具繁杂

➢ ns-3 系统提供的解决办法：

- ns-3符合标准输入/输出格式，因此可以重复使用其他工具。
  - ✓ 例如：pcap跟踪输出
- ns-3正在添加对运行实现代码的支持
- ns-3“进程”API

- 例子：使用Wireshark查看NS-3追踪：



支持虚拟化和测试平台

- **一般网络仿真系统的问题：**

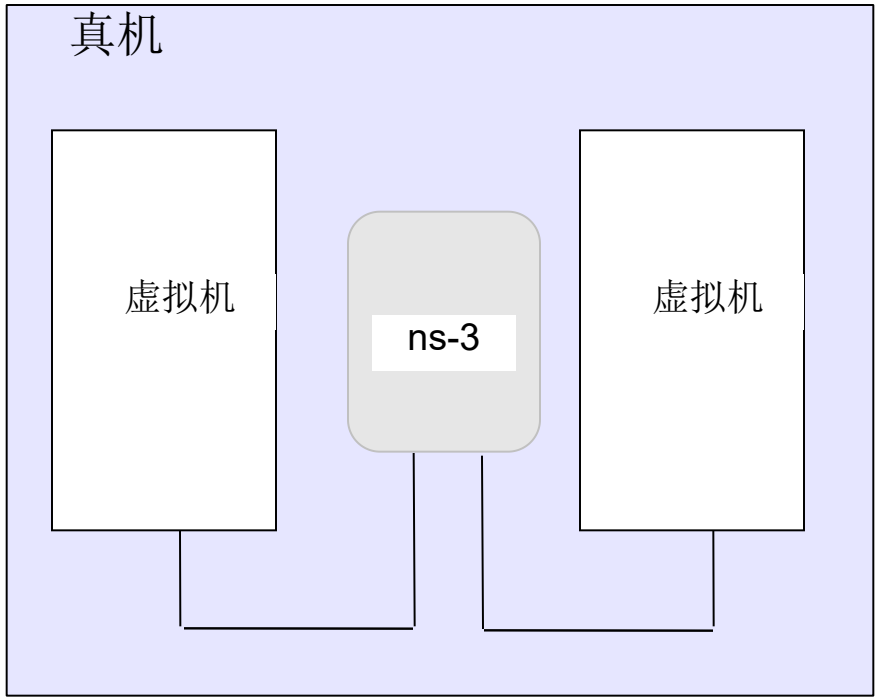
  无法很好地支持研究人员在模拟和测试平台或实时系统之间操作

- **ns-3 系统提供的解决办法：**

  开发了两种与真实系统集成的模式：

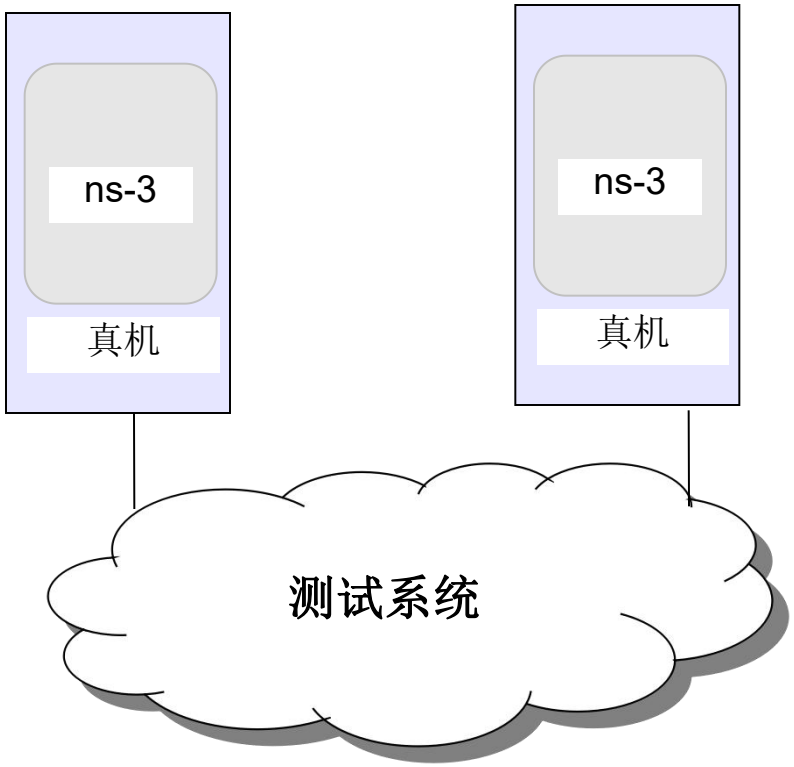
  - 1）虚拟机运行在ns-3设备和通道之上
  - 2）ns-3堆栈在仿真模式下运行，并通过实际设备发送/使用数据包

支持虚拟化和测试平台



1) NS-3连接虚拟机

ns-3.5 加入功能



2) 测试系统连接NS-3堆栈

ns-3.3加入功能

跟踪和统计：Tracing

Tracing系统：NS-3中将采集到的数据直接存放在一个文件中，以便后期处理与分析的系统。

NS-3的Tracing系统大体分为3个部分：

Tracing Sources，Tracing Sinks，以及将Tracing Sources和Tracing Sinks关联起来的方法。

跟踪是获得结构化的模拟输出结果，它要求：

- 可实现多输出
- 输出结果形式改变不影响模拟过程

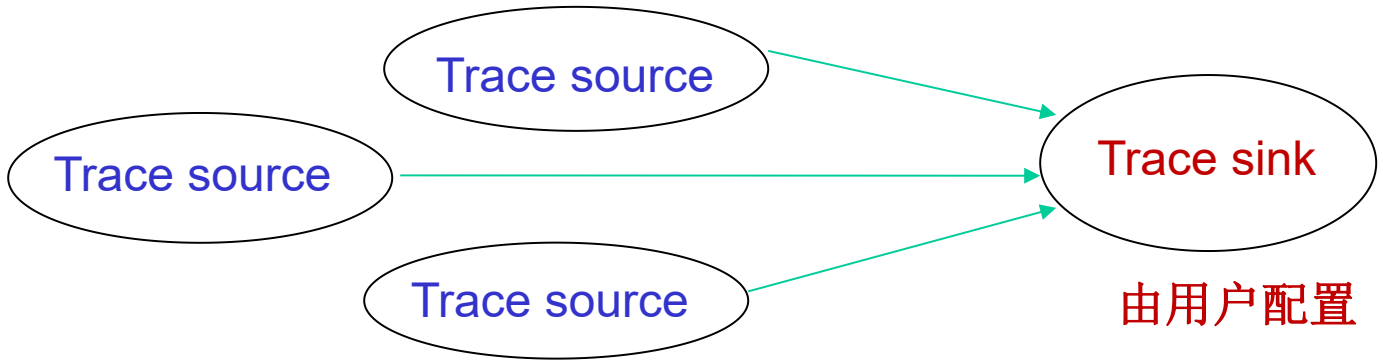
跟踪和统计：Tracing

三个部分的工作结构：

- Tracing Sources提供信息，Tracing Sinks消费信息
- Tracing Sources实体：
  - 可以用来标记仿真中发生的时间，可以提供一个访问底层数据的方法
  - 例如：当一个网络设备收到一个网络分组时，Tracing Sources可以指示，并提供一个途径将分组内容传递给对该分组感兴趣的Tracing Sinks
  - 可以在感兴趣的状态发生变化时，给出相应的指示
  - 例如：TCP网络协议中的拥塞窗口发生改变时，Tracing Sources会给出指示

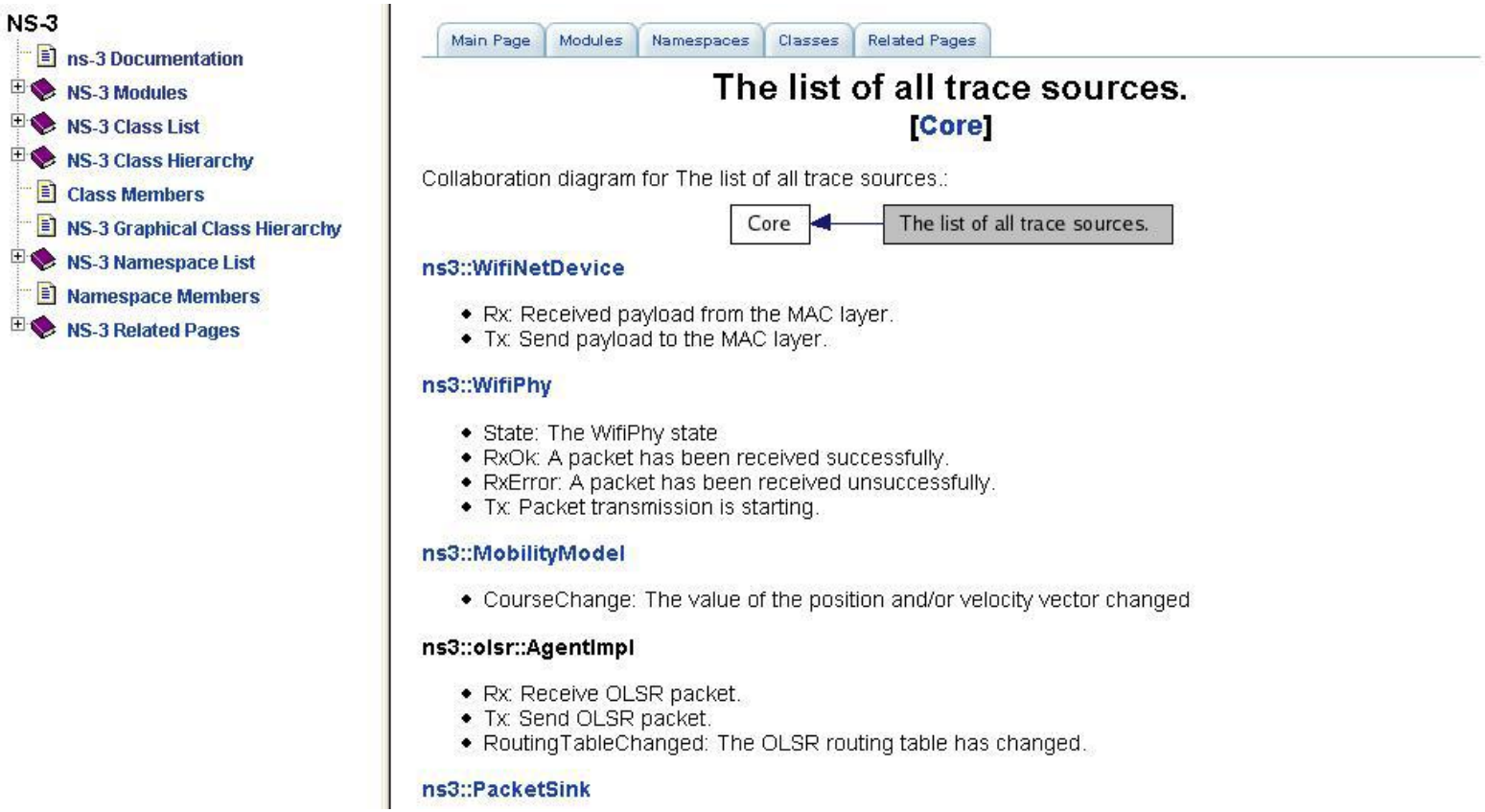
跟踪和统计：Tracing

- Tracing Sink实体：
  - Tracing Sources提供信息，Tracing Sinks输出信息。
  - Tracing Sources本身不起任何作用，只有当它和一段有实际功能的代码相关联时才有意义，这段代码使用Tracing Sources提供的信息来做相关事物。
  - Tracing Sinks：使用Tracing Sources提供信息的实体。
  - 一对多关系：一个Tracing Sources产生的信息可以没有Tracing Sink，也可以有一或多个Tracing Sink



跟踪和统计：Tracing

- 各种跟踪源（例如，分组接收，状态机转换）通过系统进行检测



跟踪和统计：统计

跟踪系统支持统计和数据管理框架

数据管理统计系统的特点：

- 可管理场景的多个独立运行
- 将数据编组为多种输出格式
- 包括数据库，每个运行元数据
- 关联到ns-3跟踪源
- 统计对象可以在运行时与模拟器交互
  - 例如：当计数器达到一个值时停止模拟

属性系统

一般网络仿真系统的问题：

无法支持研究人员快速了解模拟系统的可用属性，实现快速配置

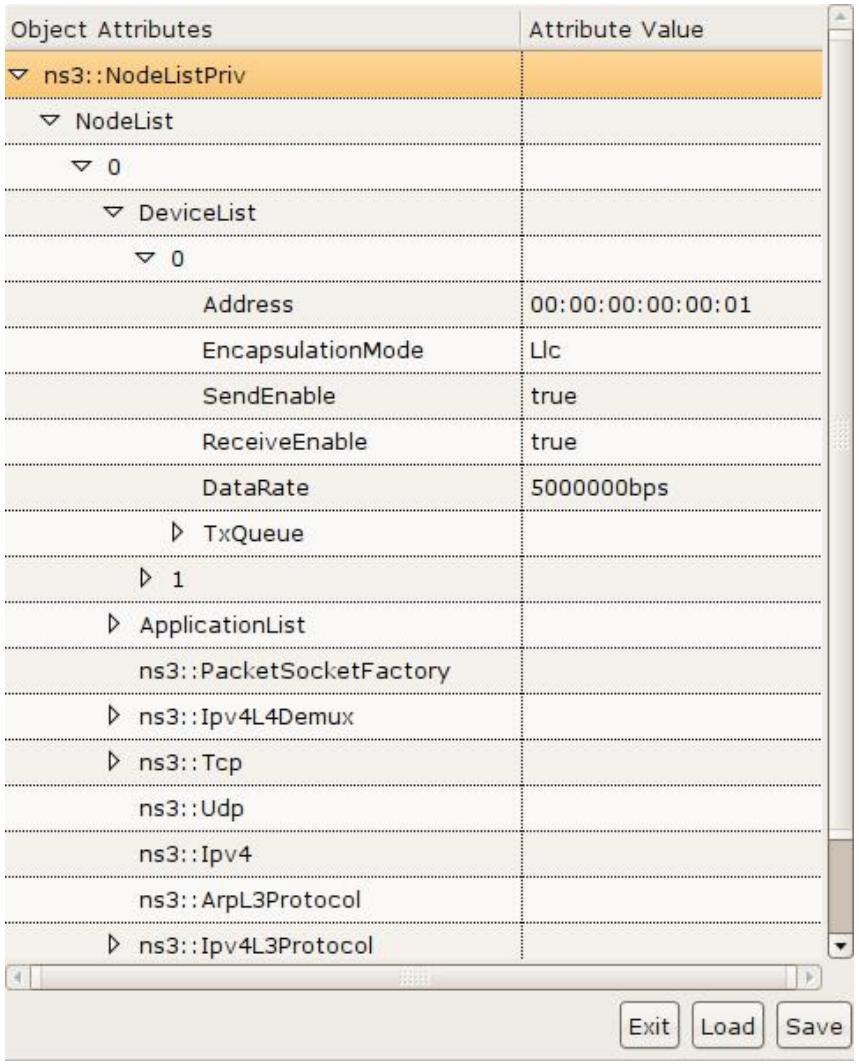
ns-3 系统提供的解决办法:

- 每个ns-3对象都有一组属性：A name, help text；A type；An initial value
- 控制静态对象的所有模拟参数
- 转储并在配置文件中读取它们
- 在GUI中可视化它们
- 可以轻松验证模拟的参数

属性系统

对象的属性在Doxygen中记录

可用构建图形配置工具



目录

结点Node

- 在NS-3中，基本计算设备抽象称为节点Node。
  - 在因特网术语中，连接到网络的计算设备称为**主机**或**终端系统**。
  - 在NS-3中是网络模拟器，我们把这样的系统称为节点，以区别特定的Internet模拟器host。
  - 这个抽象在C++中表示由类Node表示（在NS-3中也这样表示）。
  - Node类提供了用于在模拟中管理计算设备的表示方法。

- NS-3中：Node为要添加功能的计算机，添加应用程序、协议栈、外围卡及其相关驱动程序，使计算机能够做有用的工作。

应用Application

- 通常，计算机软件分为两大类：**系统软件、应用程序。**
- **系统软件：**根据某种计算模型组织各种计算机资源。
  - 如内存，处理器周期，磁盘，网络等。
  - 用户通常运行应用程序获取并使用由系统软件控制的资源，来实现某个目标。
- **应用程序：**如软件应用程序在计算机上运行以在“现实世界”中执行任务一样，NS-3应用程序在NS-3节点上运行，以在模拟世界中驱动模拟。
  - 生成一些要模拟的活动的用户程序的基本抽象
  - 在C ++中由类Application表示
  - Application类提供了在模拟中管理用户级应用程序版本的表示的方法
  - 开发人员在面向对象的编程意义上，专门化Application类来创建新的应用程序。

信道Channel

- **信道：**在现实世界中人们可以将计算机连接到网络，数据在这些网络中流动的媒体。
  - 例如：将以太网电缆连接到墙上的插头时，将计算机连接到以太网信道。
- NS-3中：信道是将节点连接到表示通信信道的对象。
  - 由C ++类Channel表示
  - Channel类提供了管理通信子网对象和将节点连接到它们的方法。
  - 可以模拟像电线一样简单的东西，还可以模拟像大型以太网交换机那样复杂的东西，或者在无线网络的情况下充满障碍物的三维空间。

网络设备Net Device

- 网络设备：
  - 如果想将计算机连接到网络，则必须有网线和需要在计算机中安装的外围设备卡的硬件设备如网络接口卡或网卡。
  - 硬件网络设备需要软件驱动程序来控制。
- 在NS-3中，网络设备抽象涵盖软件驱动程序和模拟硬件。
  - 网络设备“安装”在节点中，以使节点能够通过信道与模拟中的其他节点通信
  - 网络设备抽象在C ++中由NetDevice类表示
  - NetDevice类：提供了管理 Node和Channel对象连接的方法；并且可能由面向对象编程意义上的开发人员专门化。
  - 节点可以通过多个NetDevices连接到多个通道。

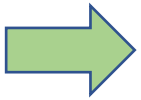
拓扑助手Topology Helpers

- 需求：在大型模拟网络中，需要在Node、NetDevices 和Channels之间安排许多连接。

- NS-3中的任务：将NetDevices连接到节点、将NetDevices连接到通道、分配IP地址等

- 拓扑辅助工具：

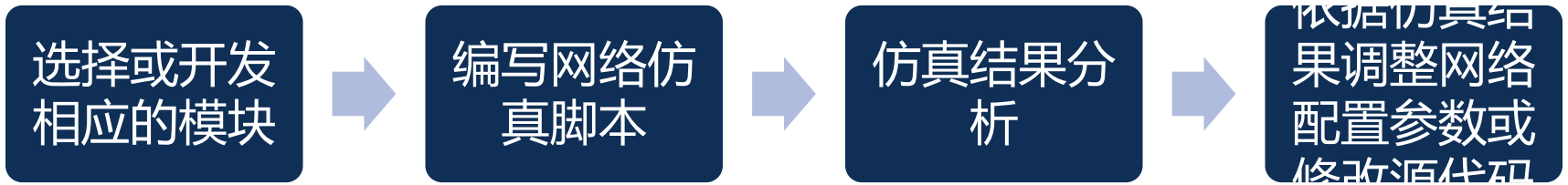
NS-3提供拓扑辅助对象，将这些许多不同的操作组合成一个易于使用的模型，以方便使用



- 创建NetDevice
- 添加MAC地址
- 在节点上安装该网络设备
- 配置节点的协议栈
- 将NetDevice连接到Channel
- 将多个设备连接到多点通道
- 将各个网络连接到互联网络中

目录

基本操作流程



首先，选择合适的模块进行开发，根据实际仿真对象和仿真场景选择相应的仿真模块：

- 有线局域网络（CSMA）/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。

基本操作流程

然后，编写网络仿真脚本， NS-3仿真脚本支持在2种语言：C++和python ， 大致包括：

- 生成节点：NS-3中的节点
- 安装网络设备：
  - 不同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
  - 如何选择不同的路由协议（OLSR、AODV、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
  - 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。
- 启动仿真：整个网络场景配置完毕，启动仿真。

➢ 获得仿真结果后，进行仿真结果分析：

- 仿真结果一般有两种
- 网络场景：如节点拓扑结构，移动模型等
  - ✓ 一般可以通过可视化界面（pyviz或NetAnim）可以直接观测到。
- 网络数据：
  - ✓ 可以在可视化界面下进行简单统计，
  - ✓ 也可以通过专门的统计框架（stats）或者通过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、分组丢失率等。

➢ 最后，依据仿真结果调整网络配置参数或修改源代码：

- 基于仿真结果的调整是网络模拟流程的关键步骤。
- 如果实际结果与预期相差较大，分析原因（是网络参数有问题还是协议本身有问题），然后重新设计，仿真，直到达到满意的结果。

多仿真工具对比

模拟器	基于C++仿真	OMNET++	NS-3
语言	C++	NED	C++/python
是否开源	是	是	是
运行速度	较慢	较快	较快
应用场景	信道仿真、通信网等多场景	有线、无线通信网络建模仿真	可对多种网络协议仿真，更加真实
图形化接口	无	NED语言与图形化接口结合	模块化图形用户界面
使用便利程度	入门门槛高，使用复杂，需要根据知识原理编写	入门门槛较高，需要前期进行多项目学习	入门门槛低，易于掌握和分析
资源调度	无	无	无
特点	数据处理能力强大，能处理的场景完善，更贴近于底层设计	自定义程度高，可以根据需求定义网络	仿真预测性好，应用成本低，可靠性高

The end