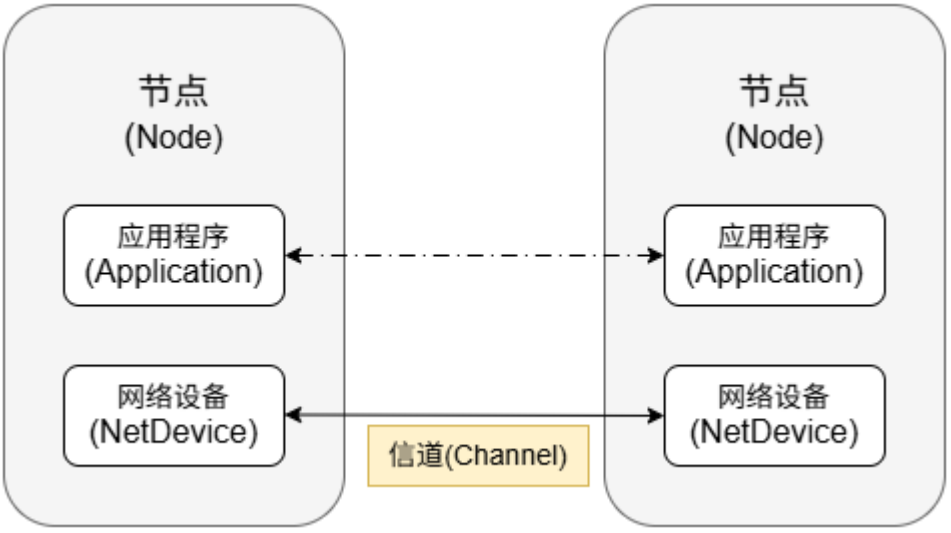
第5章 核心概念

在开始ns-3编写代码之前，需要了解ns-3的核心概念。

5.1 ns3对网络组件的抽象

ns3是一款网络模拟器，它需要对现实世界中的网络硬件和软件进行抽象，以便在ns3系统中模拟现实世界的网络通信。使用面向对象编程的思想来理解ns3：

核心概念：节点（Node），应用程序（Application），网络设备（NetDevice），信道（Channel）的关系：



- **节点:Node**
  - ns-3将基本的计算设备抽象为Node（具体表现为C++的**Node类**）
  - Node类提供了很多方法来模拟计算设备在计算机网络中的行为。
  - Node的概念对应于Internet中的host，end system...（例如：交换机、路由器、服务器、PC机、笔记本电脑、智能手机、具备联网功能传感器节点等）
  - 把Node看成一台计算机，需要向计算机内添加软硬件功能部件：应用程序，协议栈，计算机外设（例如有线网卡、无线网卡）等。
- **应用程序:Application**
  - ns-3 Application是**运行在网络节点（Node）上的软件**，以驱动模拟世界中的模拟。（具体表现为C++的**Application类**）
  - 在操作系统领域中将软件分为：系统软件和应用软件，ns-3只关注软件的网络功能，不区分是系统软件还是应用软件。
  - ns-3中有很多Application类的子类：
    - UdpEchoClientApplication
    - UdpEchoServerApplication
- **信道:Channel**
  - 信道Channel：数据传输的通道（即通信子网），有线、无线....（具体表现为C++的**Channel类**）
    - 现实世界两台主机间的信道：主机 <------> 双绞线/光纤 <------> 主机
    - 模拟世界中的信道：Node <------> Channel <------> Node

        - the basic communication subnetwork abstraction is called the channel and is represented in C++ by the class Channel
    - Channel类：提供相应的方法来管理通信子网对象，将节点连接到通信子网中。
        - 开发者可以用面向对象编程对信道进行定制化开发。
        - 一个信道的特化版本可以模拟像导线这样简单的事物。还可以模拟像大型以太网交换机这样复杂的事物，或者模拟无线网络中充满障碍物的三维空间。
            - A Channel specialization may model something as simple as a wire. 一条线缆。即使是最基础的通信媒介（比如一根普通的导线），也可以通过定义一个专门的 `Channel` 子类来建模。
            - The specialized Channel can also model things as complicated as a large Ethernet switch. 以太网交换机
            - or three-dimensional space full of obstructions in the case of wireless networks. 无线网络中充满障碍物的三维空间。
    - 常见的Channel类：
        - CsmaChannel：可以建模实现 `csma(carrier sense multiple access)` 的通信子网，类似以太网的功能。
        - PointToPointChannel：点对点信道
        - WifiChannel：无线网信道
- **网络设备:Net Device**
    - ns-3中net device抽象包括两个部分： （具体表现为C++的**NetDevice类**）
        - software driver：设备驱动程序
        - simulated hardware：仿真硬件
    - net device 被安装到 Node中，使当前Node能够和其他Node构建网络（借助Channel）。
    - 和现实世界的主机类似，一个Node可以安装多个网络设备，进而被接入到多个Channel。（笔记本电脑：有线网卡，无线网卡）
    - ns-3中网络设备的概念和现实世界中的网络设备概念的不同：
        - 现实世界中的网络设备通常指的是交换机、路由器等。
        - ns-3中的网络设备是要安装进节点中的组件，为节点拓展相应的网络功能。
    - ns-3将基本的网络设备抽象为NetDevice类：
        - NetDevice类提供了管理到Node和Channel对象的连接的方法；
        - 并且可以由开发人员在面向对象编程的意义上进行专门化（即：继承基类，进行定制化开发）。
    - 常见NetDevice类：
        - CsmaNetDevice
        - PointToPointNetDevice
        - WifiNetDevice
    - **注意：**
        - NetDevice要与Channel匹配
        - CsmaNetDevice 对应 csmchannel
        - PointToPointNetDevice被设计为与PointToPointChannel和WifiNetDevice一起工作。

- **拓扑辅助类：Topology Helpers**
  - **为什么需要Helpers类?**
    - 简化网络拓扑配置，将机械的重复性的配置操作用Helper实现!
    - > In a large simulated network you will need to arrange many connections between Nodes, NetDevices and Channels. Since connecting NetDevices to Nodes, NetDevices to Channels, assigning IP addresses, etc., are such common tasks in ns-3, we provide what we call topology helpers to make this as easy as possible.
      >
      > For example, it may take many distinct ns-3 core operations to create a NetDevice, add a MAC address, install that net device on a Node,
      > configure the node's protocol stack, and then connect the NetDevice to a Channel. Even more operations would be required to connect multiple devices onto multipoint channels and then to connect individual networks together into internetworks.
      >
      > We provide topology helper objects that combine those many distinct operations into an easy to use model for your convenience.
  - Helpers示例(在官方文档的类索引中搜索`Helper`):
    - PointToPointHelper
    - InternetStackHelper
    - Ipv4AddressHelper
    - UdpEchoServerHelper
    - UdpEchoClientHelper

5.2 第一个模拟脚本

`examples/tutorial`目录下有一些示例程序。

first.cc:

```
/*
 * SPDX-License-Identifier: GPL-2.0-only
 */

#include "ns3/applications-module.h"
#include "ns3/core-module.h"
#include "ns3/internet-module.h"
#include "ns3/network-module.h"
#include "ns3/point-to-point-module.h"

// Default Network Topology
//
//       10.1.1.0
// n0 -------------- n1
//    point-to-point
//

using namespace ns3;

NS_LOG_COMPONENT_DEFINE("FirstScriptExample");
```

```
int
main(int argc, char* argv[])
{
    CommandLine cmd(__FILE__);
    cmd.Parse(argc, argv);

    Time::SetResolution(Time::NS);
    LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
    LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);

    NodeContainer nodes;
    nodes.Create(2);

    PointToPointHelper pointToPoint;
    pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
    pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));

    NetDeviceContainer devices;
    devices = pointToPoint.Install(nodes);

    InternetStackHelper stack;
    stack.Install(nodes);

    Ipv4AddressHelper address;

    address.SetBase("10.1.1.0", "255.255.255.0");

    Ipv4InterfaceContainer interfaces = address.Assign(devices);

    UdpEchoServerHelper echoServer(9);

    ApplicationContainer serverApps = echoServer.Install(nodes.Get(1));
    serverApps.Start(Seconds(1));
    serverApps.Stop(Seconds(10));

    UdpEchoClientHelper echoClient(interfaces.GetAddress(1), 9);
    echoClient.SetAttribute("MaxPackets", UintegerValue(1));
    echoClient.SetAttribute("Interval", TimeValue(Seconds(1)));
    echoClient.SetAttribute("PacketSize", UintegerValue(1024));

    ApplicationContainer clientApps = echoClient.Install(nodes.Get(0));
    clientApps.Start(Seconds(2));
    clientApps.Stop(Seconds(10));

    Simulator::Run();
    Simulator::Destroy();
    return 0;
}
```

一个网络模拟脚本的组成部分：

1. 开源许可声明
2. 头文件
    - 需要引入相关模块对应的头文件，头文件的命名规则是：`模块名-module.h`
    - 编译后的ns3，头文件保存位置为`build/include/ns3`
    - core模块和network模块的头文件是所有的网络模拟脚本都要包含的
    - 如果使用了非ns3的第三方库，需要引入对应的头文件
    - 由于同文件众多，为了给开发者提供便利，ns-3将头文件进行了分组，将分组后的头文件放到一个`.h`文件中，开发者仅需要按照分组引入头文件即可。例如：`core-module.h`
    - ```
      #include "ns3/core-module.h"
      #include "ns3/network-module.h"
      #include "ns3/internet-module.h"
      #include "ns3/point-to-point-module.h"
      #include "ns3/applications-module.h"
      ```
3. 名字空间
    - `using namespace ns3;`
    - 在此声明之后，不必在所有ns-3代码之前键入`ns3::`范围解析操作符即可使用它。
4. 定义日志组件：`NS_LOG_COMPONET_DEFINE`
    - 这一行声明了一个名为FirstScriptExample的日志组件，它允许通过引用名称来启用和禁用控制台消息日志记录。
    - 查看日志组件API文档：Document--->Latest Release--->Doxygen API Documentation--->Topics-->Core--->Debugging tools--->Logging
    - Take a look：**Detailed Description**, **NS_LOG_COMPONENT_DEFINE**
5. main函数
    1. 准备工作（非必须）
        - 读取命令行参数
        - 设置模拟最小时间单元
        - 设置开启哪些日志组件
    2. 创建网络拓扑
        1. 创建节点
        2. 配置信道属性
        3. 创建信道并将节点连接到信道上
        4. 使用助手类来提交编程效率：NodeContainer，PointToPointHelper，NetDeviceContainer
    3. 为节点安装网络协议
        1. 通常为TCP/IP协议栈
        2. 为节点上的网络设备分配IP地址
        3. 使用助手类来提交编程效率：InternetStackHelper ，Ipv4AddressHelper，Ipv4InterfaceContainer
    4. 为节点安装应用程序以产生网络流量
      ```

1. 安装应用程序
2. 设置流量发送和接收相关参数
3. 设置应用程序启动和停止时间
4. 使用助手类来提交编程效率：UdpEchoServerHelper，UdpEchoClientHelper，ApplicationContainer

5. 网络模拟的启动和停止

模拟器什么时候停止运行?

ns-3是一个离散事件（DE）模拟器。在这样的模拟器中，每个事件与其执行时间相关联，并且通过按照模拟时间的时间顺序执行事件来进行模拟。事件可能会导致未来的事件被调度（例如，计时器可能会重新调度自身以在下一个时间间隔过期）。

初始事件通常由每个对象触发，例如，IPv6将调度路由器发布，邻居请求等，应用程序调度第一个数据包发送事件等。

当一个事件被处理时，它可以生成零个、一个或多个事件。随着模拟的执行，事件被消耗，但是可能会（也可能不会）生成更多的事件。当事件队列中没有其他事件时，或者发现特殊的stop事件时，模拟将自动停止。停止事件是通过：`Simulator::Stop(stopTime);` 函数。

有一种典型的情况，当存在一个自我维持事件时，`Simulator::Stop` 对于停止模拟是绝对必要的。自我维持（或循环）事件（`Self-sustaining or recurring`）是总是重新安排自己的事件。因此，它们总是保持事件队列非空。

有许多包含重复事件的协议和模块，例如：

- FlowMonitor：定期检查丢包情况
- RIPng：定期广播路由表更新

在这些情况下，必须使用 `Simulator::Stop` 来优雅地停止模拟。此外，当ns-3处于仿真模式（`emulation mode`）时，`RealtimeSimulator` 用于使仿真时钟与机器时钟保持一致，并且需要 `Simulator::Stop` 来停止该进程。

Tutorial教程中的许多模拟程序没有显式地调用 `Simulator::Stop`，因为事件队列将自动用完事件。然而，这些程序也将接受对 `Simulator::Stop` 的调用。例如，第一个示例程序中的以下附加语句将在11秒时显式地调度停止：

```
    Simulator::Stop(Seconds(11));
    Simulator::Run();
    Simulator::Destroy();
    return 0;
}
```

以上内容实际上不会改变这个程序的行为，因为这个特定的模拟会在10秒后自然结束。但是，如果将上述语句中的停止时间从11秒更改为1秒，则会注意到模拟在任何输出打印到屏幕之前就停止了（因为输出发生在模拟时间的2秒左右）。

在调用 `Simulator::Run` 之前调用 `Simulator::Stop` 是很重要的。否则，`Simulator::Run` 可能永远不会将控制返回给主程序来执行停止!

运行第一个网络模拟程序:

```
$ cd workspace/ns-allinone-3.45/ns-3.45
$ ./ns3 run first
# 下面的指令创建自己的第一个网络模拟程序，并运行
```

```
$ cp examples/tutorial/first.cc scratch/myfirst.cc
$ ./ns3 build # 编译myfirst.cc 同：./ns3
......省略部分输出内容......
-- Configuring done (1.2s)
-- Generating done (1.5s)
-- Build files have been written to: /home/ns3/workspace/ns-allinone-3.45/ns-3.45/cmake-
cache
[  0%] Building CXX object scratch/CMakeFiles/scratch_myfirst.dir/myfirst.cc.o
[  0%] Linking CXX executable /home/ns3/workspace/ns-allinone-3.45/ns-
3.45/build/scratch/ns3.45-myfirst-default
Finished executing the following commands:
/usr/bin/cmake --build /home/ns3/workspace/ns-allinone-3.45/ns-3.45/cmake-cache -j 1
$ ./ns3 run myfirst    # 同: ./ns3 run scratch/myfirst
At time +2s client sent 1024 bytes to 10.1.1.2 port 9
At time +2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time +2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time +2.00737s client received 1024 bytes from 10.1.1.2 port 9
```

5.3 ns3源码概览

src目录下是ns3的源码：

```
ns3@ns3-vm:~/ns-allinone-3.45/ns-3.45$ cd src
ns3@ns3-vm:~/ns-allinone-3.45/ns-3.45/src$ ls core
CMakeLists.txt  doc  examples  helper  model  test
ns3@ns3-vm:~/ns-allinone-3.45/ns-3.45/src$ ls wifi
CMakeLists.txt  doc  examples  helper  model  test
ns3@ns3-vm:~/ns-allinone-3.45/ns-3.45/src$ ls csma
CMakeLists.txt  doc  examples  helper  model  test
```

example下有很多示例代码，通过这些示例代码可以快速上手开发：

```
ns3@ns3-vm:~/ns-allinone-3.45/ns-3.45$ ls -l examples/
total 68
drwxrwxr-x 2 ns3 ns3 4096 Jul  6 12:38 channel-models
-rw-rw-r-- 1 ns3 ns3  514 Mar 10 07:55 CMakeLists.txt
drwxrwxr-x 2 ns3 ns3 4096 Jul  6 12:38 energy
drwxrwxr-x 2 ns3 ns3 4096 Jul  6 12:38 error-model
drwxrwxr-x 2 ns3 ns3 4096 Jul  6 12:38 ipv6
drwxrwxr-x 2 ns3 ns3 4096 Jul  6 12:38 matrix-topology
drwxrwxr-x 2 ns3 ns3 4096 Jul  6 12:38 naming
drwxrwxr-x 2 ns3 ns3 4096 Jul  6 12:38 realtime
drwxrwxr-x 2 ns3 ns3 4096 Jul  6 12:38 routing
drwxrwxr-x 2 ns3 ns3 4096 Jul  6 12:38 socket
drwxrwxr-x 2 ns3 ns3 4096 Jul  6 12:38 stats
drwxrwxr-x 2 ns3 ns3 4096 Jul  6 12:38 tcp
drwxrwxr-x 2 ns3 ns3 4096 Jul  6 12:38 traffic-control
drwxrwxr-x 2 ns3 ns3 4096 Aug 10 19:37 tutorial
drwxrwxr-x 2 ns3 ns3 4096 Jul  6 12:38 udp
drwxrwxr-x 2 ns3 ns3 4096 Jul  6 12:38 udp-client-server
drwxrwxr-x 2 ns3 ns3 4096 Jul  6 12:38 wireles
```

5.4 补充：相关类总结

节点相关的类

Node

https://www.nsnam.org/docs/release/3.45/doxygen/d0/d3d/classns3_1_1_node.html#details

> This class holds together:
>
> - a list of NetDevice objects which represent the network interfaces of this node which are connected to other Node instances through Channel instances.
> - a list of Application objects which represent the userspace traffic generation applications which interact with the Node through the Socket API.
> - a node Id: a unique per-node identifier.
> - a system Id: a unique Id used for parallel simulations.
>
> Every Node created is added to the NodeList automatically.

一个Node节点内包含：

- 一个Netdevice列表
- 一个Application列表
- 节点ID
- 系统ID：并行仿真中使用

NodeList

https://www.nsnam.org/docs/release/3.45/doxygen/db/d1a/classns3_1_1_node_list.html

> the list of simulation nodes.
>
> Every Node created is automatically added to this list.

`NodeList`是一个**全局单例类**，负责维护网络中所有节点的完整列表。每当创建新的节点时，该节点会自动被添加到`NodeList`中，提供静态方法来访问所有节点。主要用于**查询和遍历节点**。

NodeContainer

https://www.nsnam.org/docs/release/3.45/doxygen/d7/db2/classns3_1_1_node_container.html

> keep track of a set of node pointers.
>
> Typically ns-3 helpers operate on more than one node at a time. For example a device helper may want to install devices on a large number of similar nodes. The helper Install methods usually take a NodeContainer as a parameter. NodeContainers hold the multiple Ptr<Node> which are used to refer to the nodes.

`NodeContainer`是一个**辅助容器类**，用于方便地管理和操作节点的子集：可以包含任意节点的子集，支持节点的添加、合并等操作，简化了节点组的创建和管理。

一个设备辅助程序（Helper）通常需要在大量类似的节点上安装网络设备。辅助程序（Helper）的安装方法通常会将一个 NodeContainer 作为参数传入。

信道相关的类

Channel

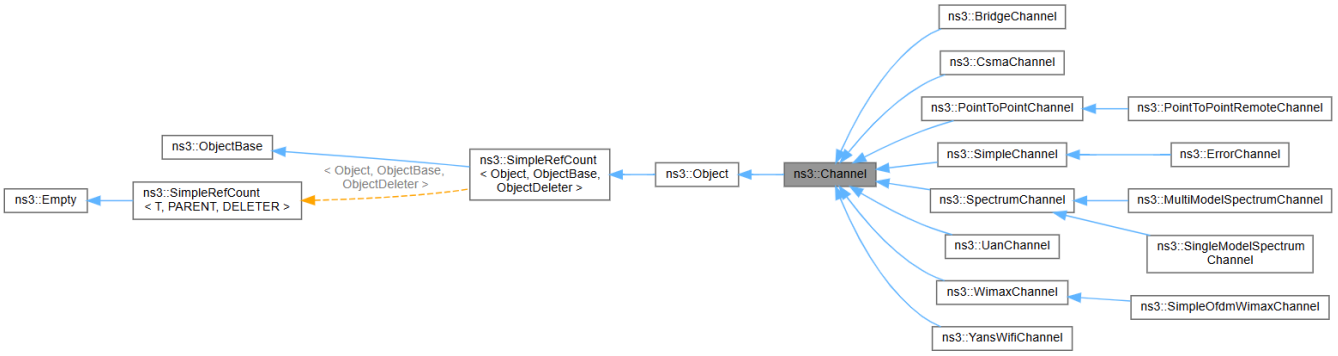
https://www.nsnam.org/docs/release/3.45/doxygen/d5/d61/classns3_1_1_channel.html

> Abstract Channel Base Class.
>
> A channel is a logical path over which information flows. The path can be as simple as a short piece of wire, or as complicated as space-time.
>
> Subclasses must use Simulator::ScheduleWithContext to correctly update event contexts when scheduling an event from one node to another one.

强烈建议查阅`Inheritance diagram`。

常见的信道子类：

- ns3::BridgeChannel
- ns3::CsmaChannel
- ns3::PointToPointChannel
- ns3::SimpleChannel
- ns3::SpectrumChannel
- ns3::UanChannel
- ns::WimaxChannel
- ns::YansWifiChannel



ChannelList

https://www.nsnam.org/docs/release/3.45/doxygen/d5/df2/classns3_1_1_channel_list.html#details

> the list of simulation channels.
>
> Every Channel created is automatically added to this list.

`ChannelList`是一个**全局单例类**，负责维护网络中所有信道的完整列表。每当创建新的信道时，该信道会自动被添加到`ChannelList`中，提供静态方法来访问所有信道。主要用于**查询和遍历信道**。

网络设备相关的类

NetDevice

https://www.nsnam.org/docs/release/3.45/doxygen/d0/da3/classns3_1_1_net_device.html

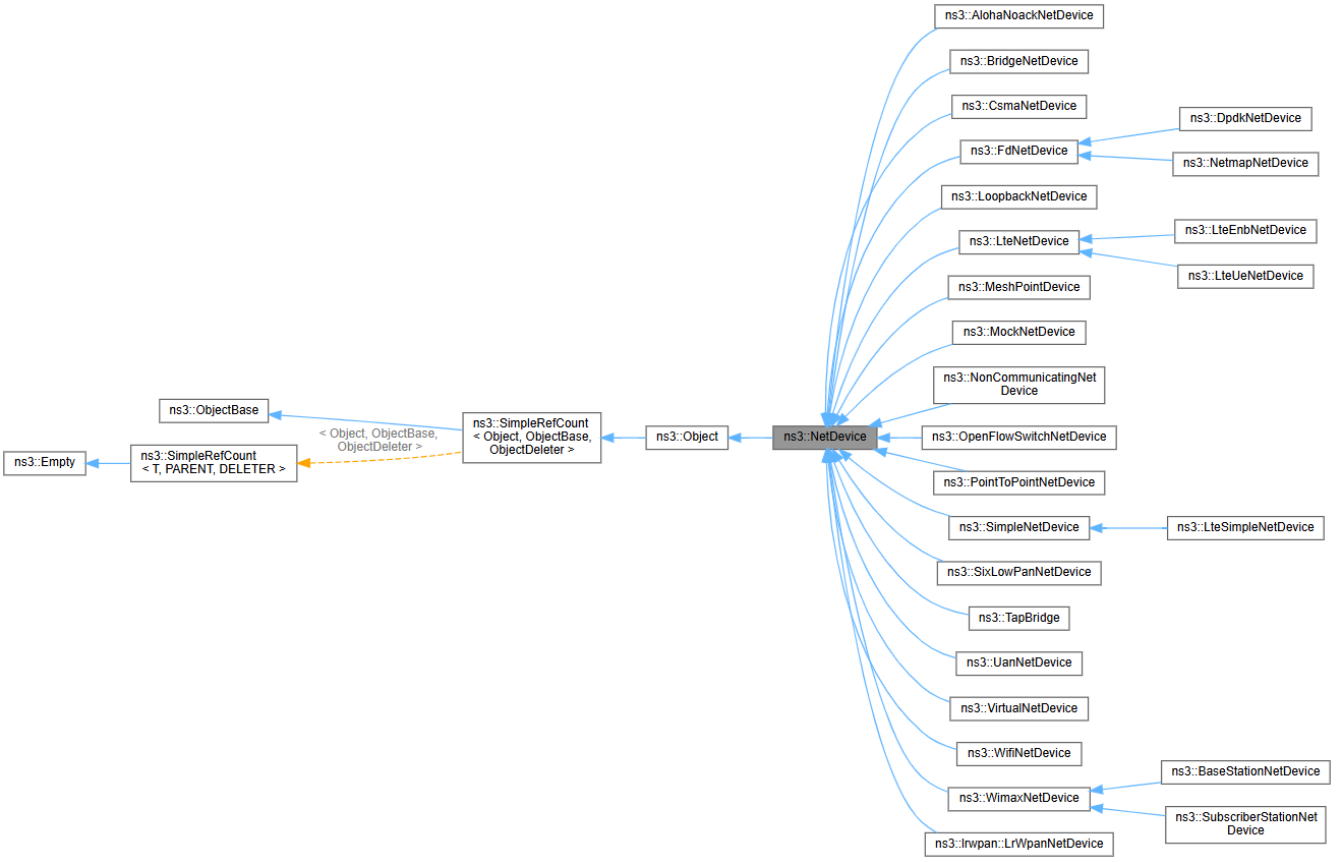
> Network layer to device interface.
>
> This interface defines the API which the IP and ARP layers need to access to manage an instance of a network device layer. It currently does not support MAC-level multicast but this should not be too hard to add by adding extra methods to register MAC multicast addresses to filter out unwanted packets before handing them to the higher layers.
>
> In Linux, this interface is analogous to the interface just above dev_queue_xmit() (i.e., IP packet is fully constructed with destination MAC address already selected).
>
> If you want to write a new MAC layer, you need to subclass this base class and implement your own version of the pure virtual methods in this class.
>
> This class was designed to hide as many MAC-level details as possible from the perspective of layer 3 to allow a single layer 3 to work with any kind of MAC layer. Specifically, this class encapsulates the specific format of MAC addresses used by a device such that the layer 3 does not need any modification to handle new address formats. This means obviously that the NetDevice class must know about the address format of all potential layer 3 protocols through its GetMulticast methods: the current API has been optimized to make it easy to add new MAC protocols, not to add new layer 3 protocols.
>
> Devices aiming to support flow control and dynamic queue limits must perform the following operations:（旨在支持流量控制和动态队列限制的设备必须执行以下操作：）
>
> - in the NotifyNewAggregate method
>   - cache the pointer to the netdevice queue interface aggregated to the device
>   - set the select queue callback through the netdevice queue interface, if the device is multi-queue
> - anytime before initialization
>   - set the number of device transmission queues (and optionally create them) through the netdevice queue interface, if the device is multi-queue
> - when the device queues have been created, invoke NetDeviceQueueInterface::ConnectQueueTraces, which
>   - connects the Enqueue traced callback of the device queues to the PacketEnqueued static method of the NetDeviceQueue class
>   - connects the Dequeue and DropAfterDequeue traced callback of the device queues to the PacketDequeued static method of the NetDeviceQueue class
>   - connects the DropBeforeEnqueue traced callback of the device queues to the PacketDiscarded static method of the NetDeviceQueue class

强烈建议查阅`Inheritance diagram`.

子类(仅列出部分子类):

- ns3::BridgeNetDevice

- ns3::CsmaNetdevice
- ns3::LoopbackNetDevice
- ns3::LwWpanNetDevice
- ns3::LteNetDevice
- ns3::SimpleNetDevice
- ns3::MeshPointDevice
- ns3::PointToPointNetDevice



NetDevice相关的Helper

上面图中的大部分NetDevice都有一个对应的Helper类，用于创建和管理对应的网络设备。例如：PointToPointHelper，CsmaHelper。

NetDeviceContainer

https://www.nsnam.org/docs/release/3.45/doxygen/d6/ddf/classns3_1_1_net_device_container.html

> holds a vector of ns3::NetDevice pointers
>
> Typically ns-3 NetDevices are installed on nodes using a net device helper. The helper Install method takes a NodeContainer which holds some number of Ptr. For each of the Nodes in the NodeContainer the helper will instantiate a net device, add a MAC address and a queue to the device and install it to the node. For each of the devices, the helper also adds the device into a Container for later use by the caller. This is that container used to hold the Ptr which are instantiated by the device helper.

通常使用网络设备助手将ns-3 netdevice安装在节点上。Helper的Install方法接受一个NodeContainer，该NodeContainer包含一定数量的`Ptr<Node>`。对于NodeContainer中的每个节点，助手将实例化一个网络设备，向设备添加MAC地址和队列，并将其安装到节点上。对于每个设备，Helper还将设备添加到NetDeviceContainer中，供调用者稍后使用。NetDeviceContainer用来保存`Ptr<NetDevice>`的容器，它由设备助手实例化的。

应用程序相关的类

Application

https://www.nsnam.org/docs/release/3.45/doxygen/de/d96/classns3_1_1_application.html

> The base class for all ns3 applications.

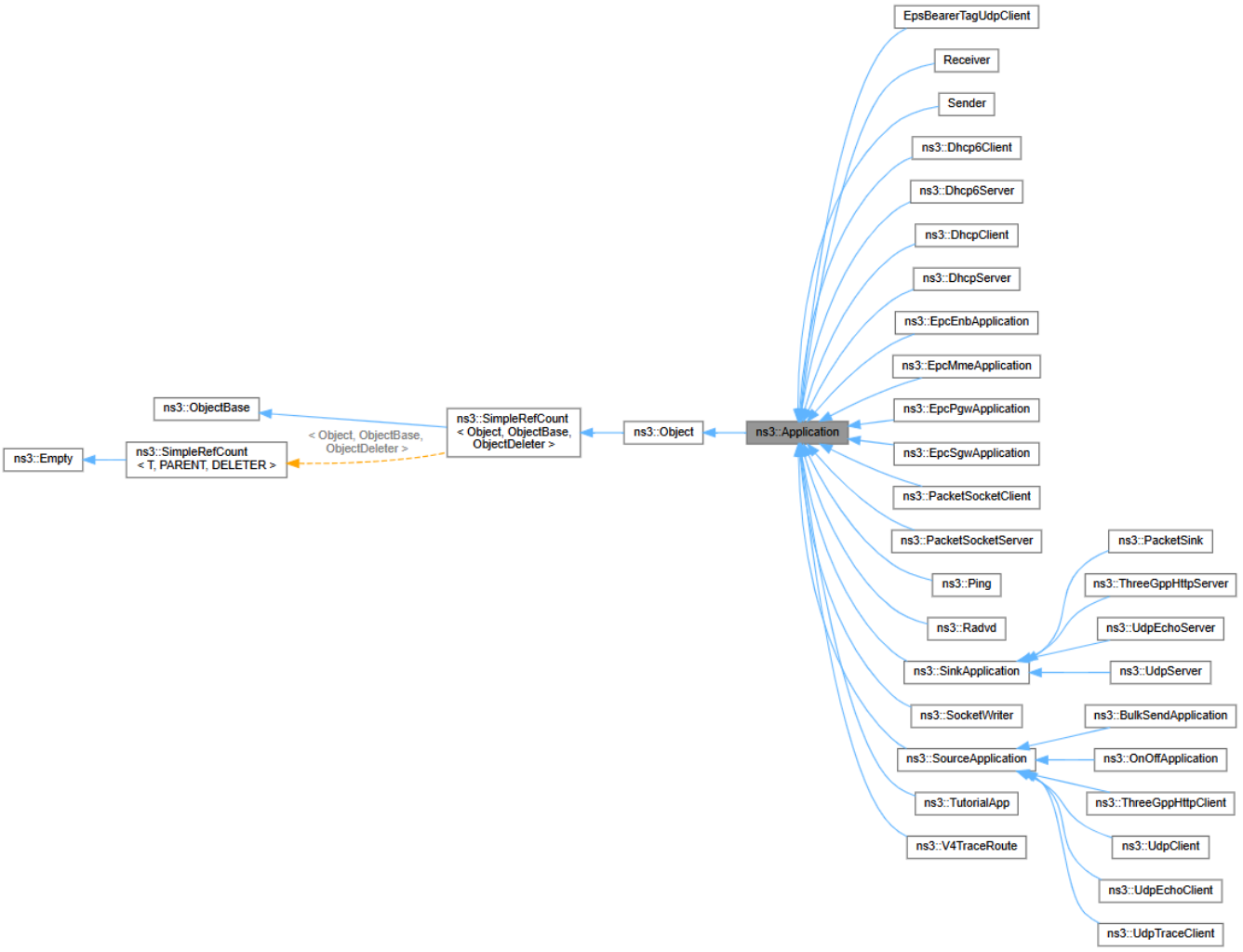
Application是所有应用程序的基类。

强烈建议查阅`Inheritance diagram`.

子类(仅列出部分子类):

- ns3::DhcpClient
- ns3::DhcpServer
- ns3::Ping6
- ns3::UdpEchoClient
- ns3::UdpEchoServer
- ns3::V4Ping
- ns3::V4TraceRoute

Application的继承关系:



ApplicationContainer

https://www.nsnam.org/docs/release/3.45/doxygen/df/dbe/classns3_1_1_application_container.html

> holds a vector of ns3::Application pointers.
>
> Typically ns-3 Applications are installed on nodes using an Application helper. The helper Install method takes a NodeContainer which holds some number of Ptr. For each of the Nodes in the NodeContainer the helper will instantiate an application, install it in a node and add a Ptr to that application into a Container for use by the caller. This is that container used to hold the Ptr which are instantiated by the Application helper.

通常使用Application helper将ns-3应用程序安装到节点上。helper Install方法接受一个NodeContainer，该NodeContainer包含一定数量的`Ptr<Node>`。对于NodeContainer中的每个节点，helper将实例化一个应用程序，将其安装在节点中，并将`Ptr< application >`添加到该ApplicationContainer容器中，以供调用者使用。ApplicationContainer是用来保存`Ptr<Application>`的容器，由Application helper实例化。

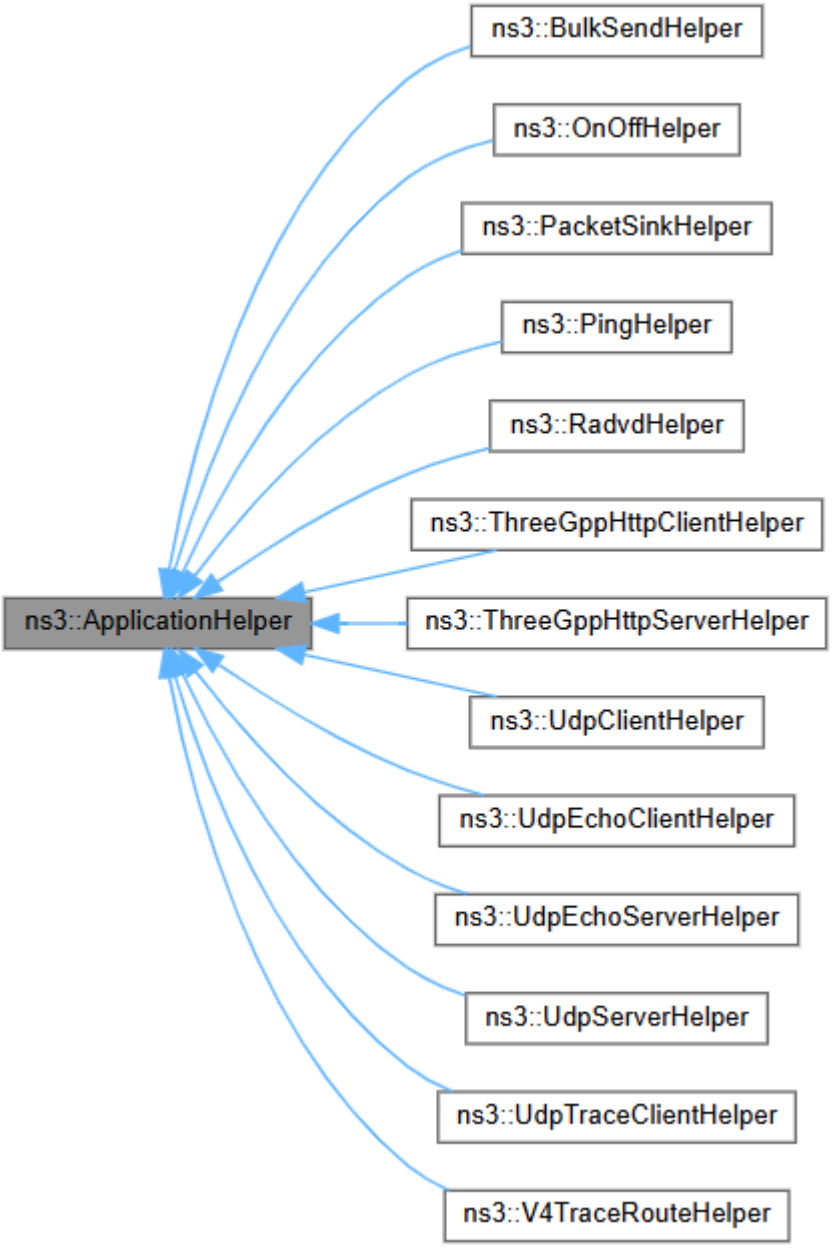
ApplicatioionHelper

https://www.nsnam.org/docs/release/3.45/doxygen/d6/d9c/classns3_1_1_application_helper.html

> A helper to make it easier to instantiate an application on a set of nodes.

简化应用程序的管理，在一组节点上实例化应用程序。

ApplicationHelper的继承关系：


ns3::BulkSendHelper
ns3::OnOffHelper
ns3::PacketSinkHelper
ns3::PingHelper
ns3::RadvdHelper
ns3::ThreeGppHttpClientHelper
ns3::ThreeGppHttpServerHelper
ns3::ApplicationHelper
ns3::UdpClientHelper
ns3::UdpEchoClientHelper
ns3::UdpEchoServerHelper
ns3::UdpServerHelper
ns3::UdpTraceClientHelper
ns3::V4TraceRouteHelper