# BUAANetworkExperimentTutorial说明手册

网络技术北京市重点实验室 钱程 2021/4/13

## 简介

本教程采用OMNeT++仿真环境和INET仿真模型库，在仿真平台上重现高等教育出版社《计算机网络实验教程-第二版》参考书中的网络实验，目的在于帮助读者学习计算机网络的相关知识，和OMNeT++环境INET框架下的网络构建与配置方法。

**目录**

[BUAANetworkExperimentTutorial说明手册 1](#_Toc69287102)

[简介 1](#_Toc69287103)

[0. 基础 3](#_Toc69287104)

[0.1. OMNeT++平台 3](#_Toc69287105)

[0.2. INET框架 3](#_Toc69287106)

[0.3. 运行环境 3](#_Toc69287107)

[0.4. 安装 3](#_Toc69287108)

[0.5. 运行仿真 7](#_Toc69287109)

[0.6. NED与INI文件 8](#_Toc69287110)

[1. 网络实验入门 9](#_Toc69287111)

[1.1. 简单局域网组建实验 9](#_Toc69287112)

[1.2. 基于地址转换的组网实验（NAT功能仍需调试） 10](#_Toc69287113)

[2. 数据链路层实验 11](#_Toc69287114)

[2.1. 以太网帧格式分析 11](#_Toc69287115)

[2.2. 以太网帧格式分析 12](#_Toc69287116)

[2.3. VLAN配置与分析（VLAN基本配置） 13](#_Toc69287117)

[2.4. VLAN配置与分析（Trunk和Hybrid配置） 13](#_Toc69287118)

[2.5. 广域网数据链路层实验 14](#_Toc69287119)

[3. 网络层实验 15](#_Toc69287120)

[3.1. ARP分析 15](#_Toc69287121)

[3.2. ICMP分析 15](#_Toc69287122)

[3.3. 网络层分片实验 16](#_Toc69287123)

[3.4. VLAN间通信 16](#_Toc69287124)

[4. 传输层实验 17](#_Toc69287125)

[4.1. TCP基本分析 17](#_Toc69287126)

[4.2. UDP分析 21](#_Toc69287127)

[5. 应用层实验 22](#_Toc69287128)

[5.1. HTTP分析 22](#_Toc69287129)

[5.2. DHCP分析 23](#_Toc69287130)

[6. RIP实验 25](#_Toc69287131)

[6.1. 静态路由与默认路由配置 25](#_Toc69287132)

[6.2. RIP配置和RIPv1报文结构分析 25](#_Toc69287133)

[6.3. 距离矢量算法分析 26](#_Toc69287134)

[6.4. 触发更新和水平分割 27](#_Toc69287135)

[6.5. 设计型实验2 28](#_Toc69287136)

[7. OSPF协议实验 28](#_Toc69287137)

[7.1. OSPF协议概述及基本配置 28](#_Toc69287138)

[7.2. OSPF报文交互过程 29](#_Toc69287139)

[7.3. OSPF协议的链路状态描述 30](#_Toc69287140)

[7.4. 区域划分以及LSA种类 32](#_Toc69287141)

[7.5. OSPF协议路由的计算 32](#_Toc69287142)

[7.6. 设计型实验1 32](#_Toc69287143)

[7.7. 设计型实验2 32](#_Toc69287144)

[8. BGP协议实验 33](#_Toc69287145)

[8.1. BGP的基本分析 33](#_Toc69287146)

[8.2. BGP的路由策略 33](#_Toc69287147)

[8.3. BGP的同步机制 33](#_Toc69287148)

[8.4. 设计型实验1 33](#_Toc69287149)

[8.5. 设计型实验2 33](#_Toc69287150)

[8.6. 设计型实验3 33](#_Toc69287151)

[9. 网络管理实验（未完成） 33](#_Toc69287152)

[10. 组播实验 33](#_Toc69287153)

[参考文献 34](#_Toc69287154)

## 基础

本章节将介绍OMNeT++平台、INET框架和本教程的相关基础。

### OMNeT++平台

OMNeT++（https://omnetpp.org/）是可扩展的，基于组件的模块化C ++仿真库和框架，主要用于构建网络仿真器。“网络”包括有线和无线通信网络，片上网络，排队网络等。作为独立项目开发的模型框架提供了特定的功能，例如对传感器网络，无线自组织网络，Internet协议，性能建模，光子网络等的支持。OMNeT ++提供了一个基于Eclipse的IDE，一个图形运行时环境以及许多其他工具。实时仿真，网络仿真，数据库集成，SystemC集成和其他一些功能都有扩展。

### INET框架

INET Framework（https://inet.omnetpp.org/）是OMNeT ++仿真环境的开源模型库。它为使用通信网络的研究人员和学生提供协议，代理和其他模型。在设计和验证新协议，或探索新的或特殊的情况时，INET尤其有用。INET包含以下模型：互联网堆栈（TCP，UDP，IPv4，IPv6，OSPF，BGP等），有线和无线链路层协议（以太网，PPP，IEEE 802.11等），对移动性的支持，MANET协议，DiffServ，具有LDP和RSVP-TE信令的MPLS，几种应用模型以及许多其他协议和组件。

### 运行环境

本教程的开发环境采用Win10操作系统，OMNeT++5.6.2版本，INET Framework 4.2.2版本。

### 安装

请按以下步骤安装：

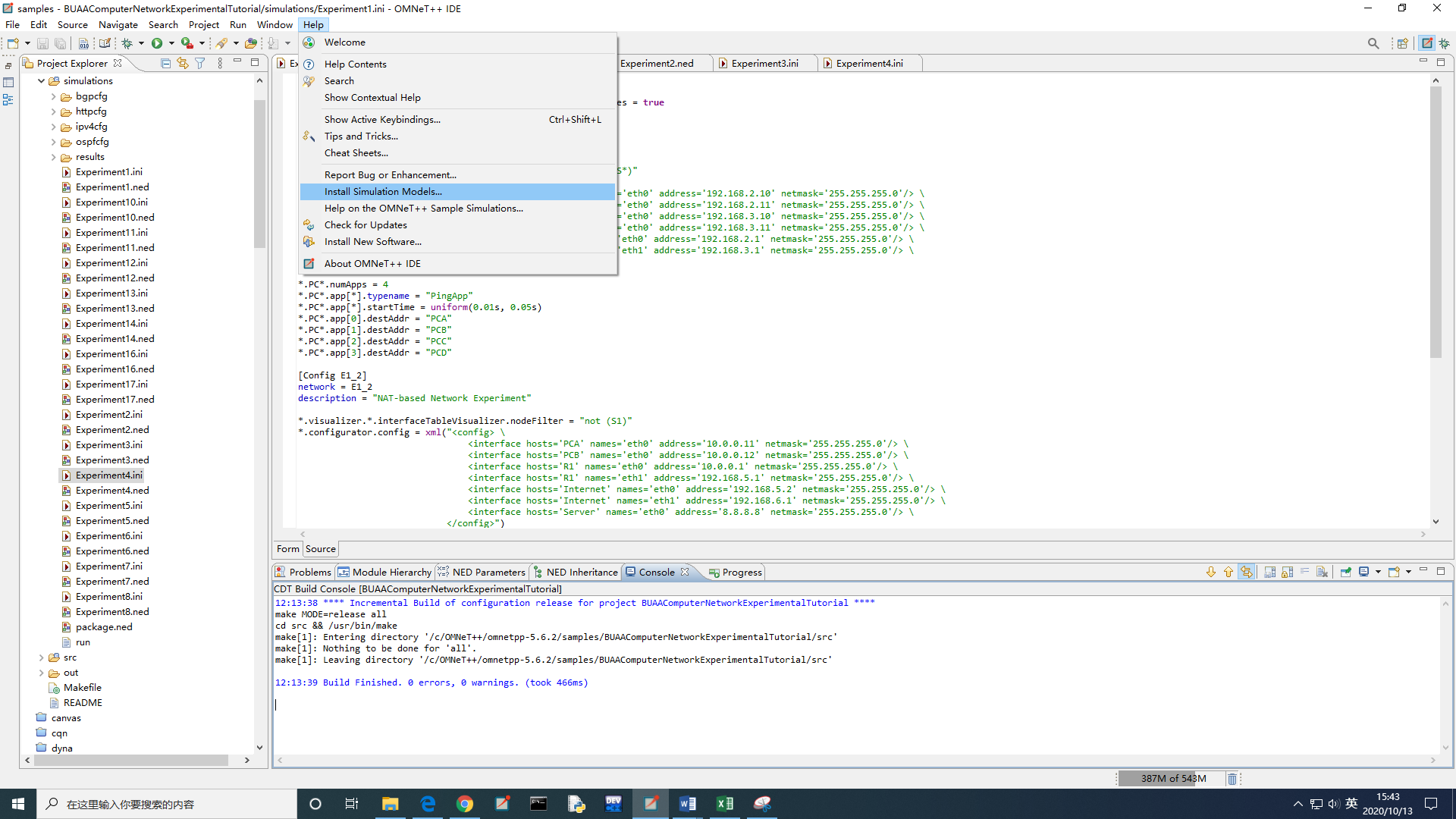
* **请参考官网和安装提示步骤安装OMNeT++（下载：https://omnetpp.org/download/）**



**图0.1 OMNeT++IDE截图**

* **自动或手动安装INET框架至WorkSpace并Build。**

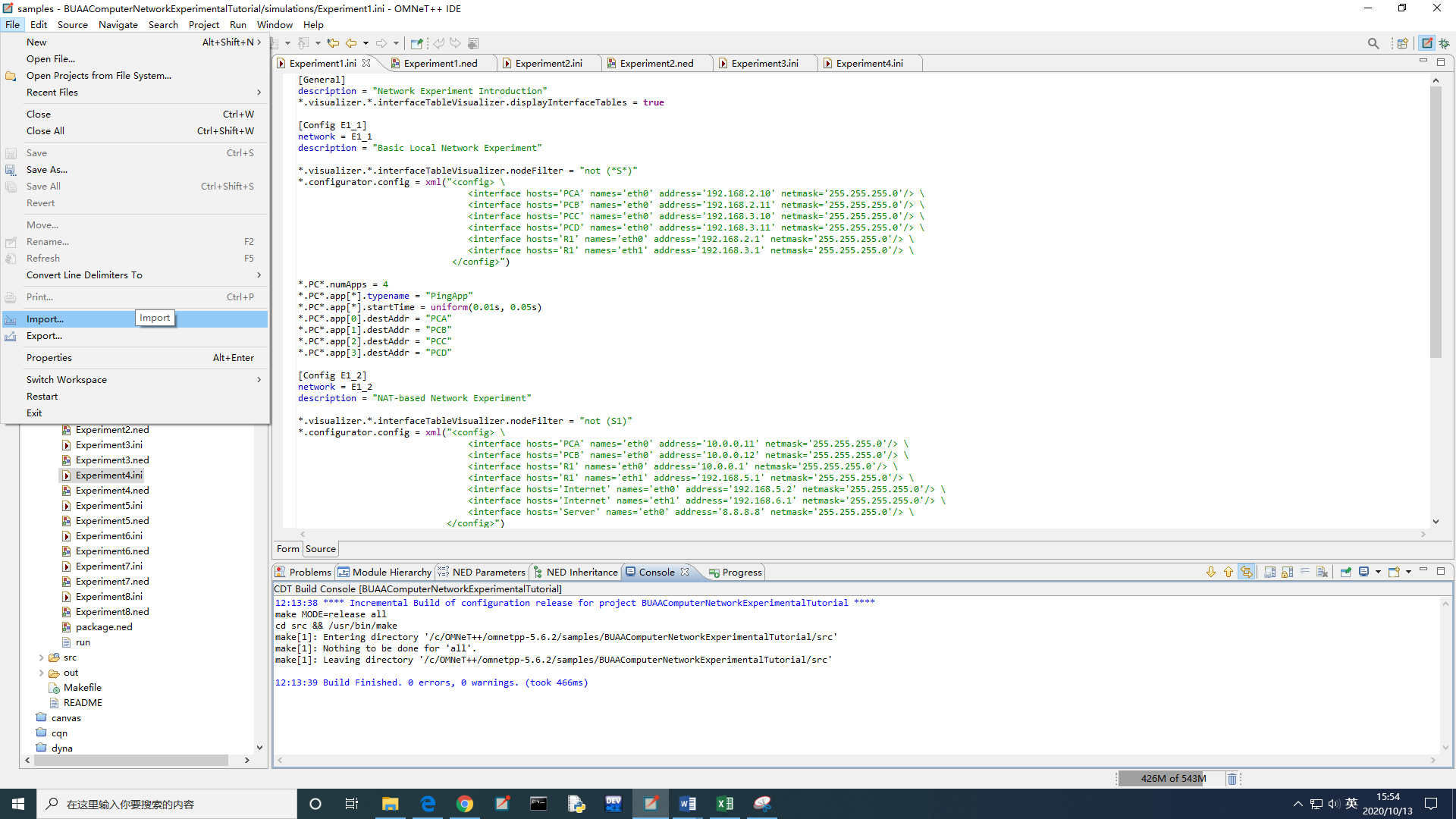
第一次运行OMNeT++IDE时会提示安装INET Framework，之后也可以点击Help🡪Install Simulation Models ，自动安装INET Framework。



**图0.2 自动安装INET Framework的菜单选项**

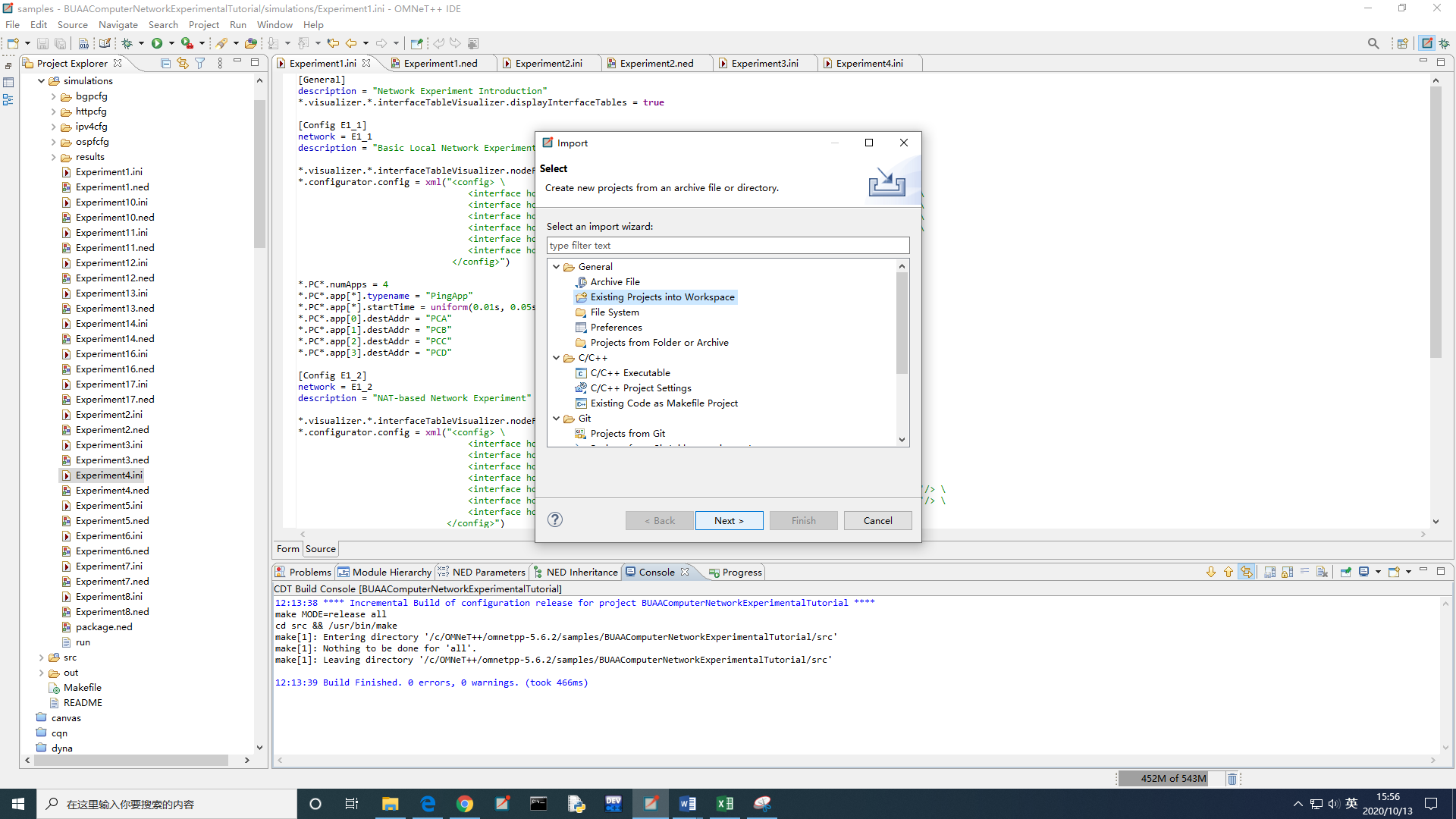
**提示**：自动安装时可能会因为网络原因导致下载超时无法正常安装，如有类似情况请使手动安装INET Framework。

手动安装INET Framework需要预先从官方网站下载INET Framework4.2.0+的工程文件，并解压至WorkSpace。点击File 🡪 Import。



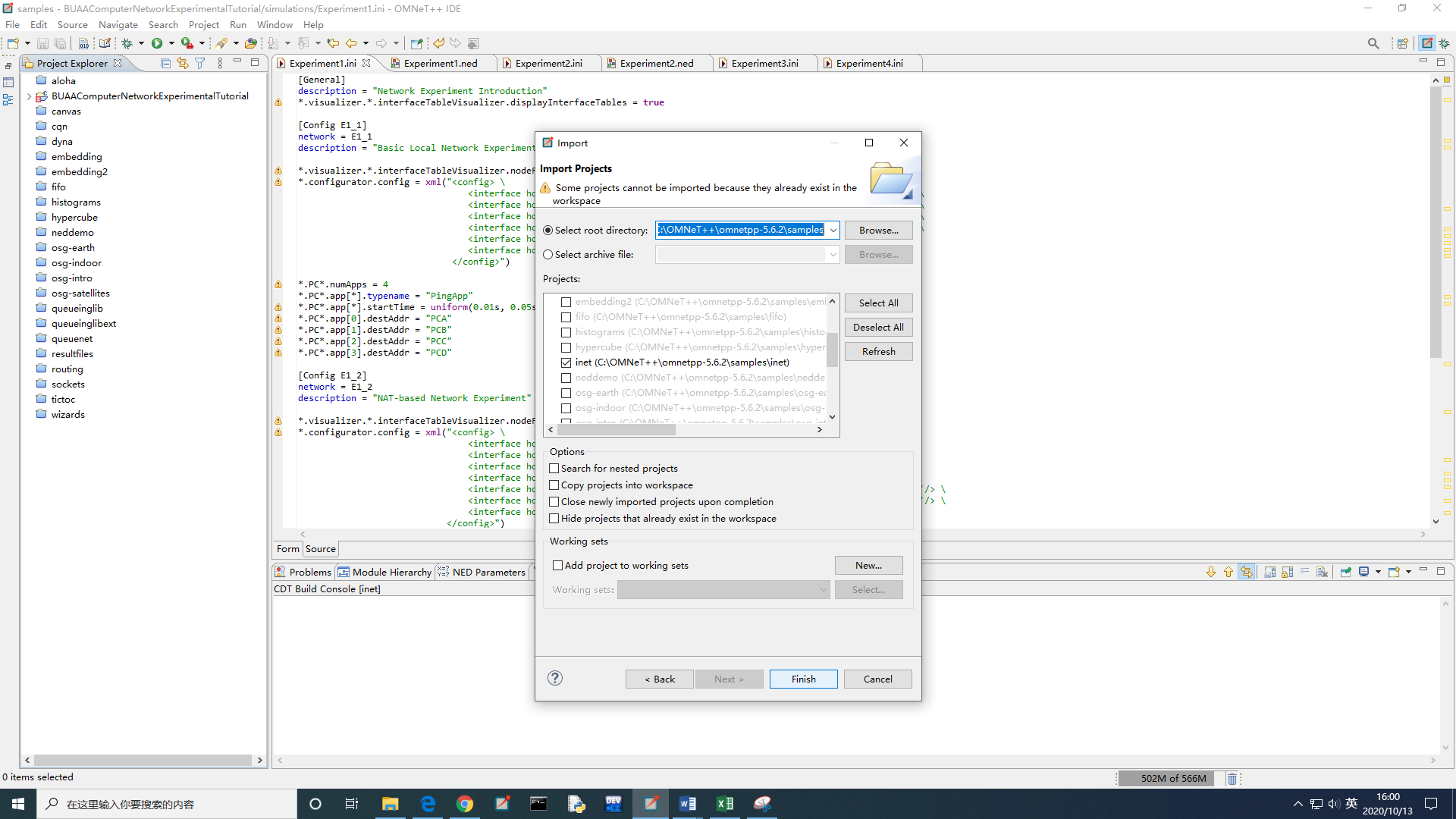
**图0.3 手动导入INET Framework的菜单选项1**

点击Existing Projects into Workspace 🡪 Next >。



**图0.4 手动导入INET Framework的菜单选项2**

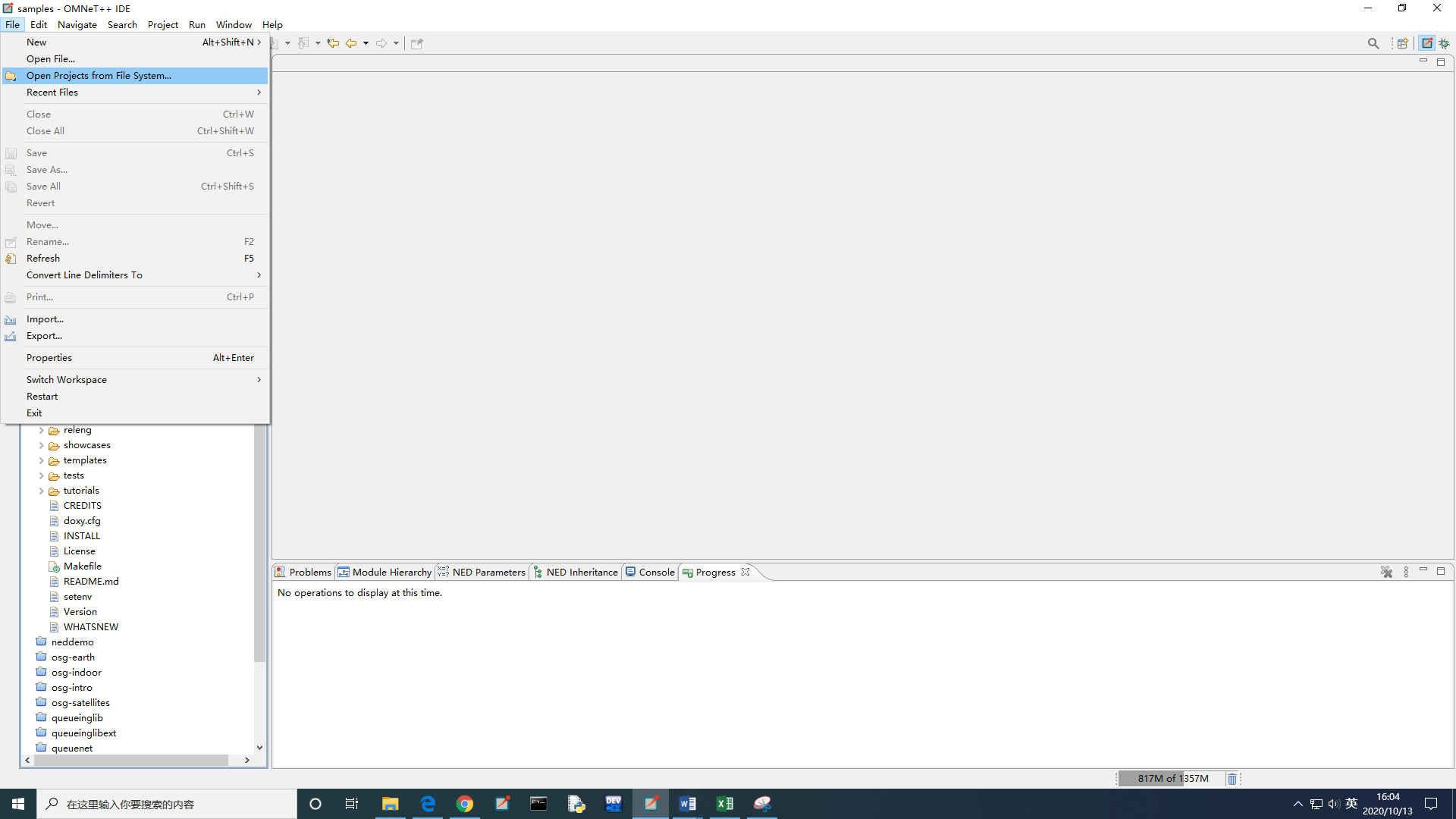
勾选inet，点击finish完成导入。



**图0.5 手动导入INET Framework的菜单选项3**

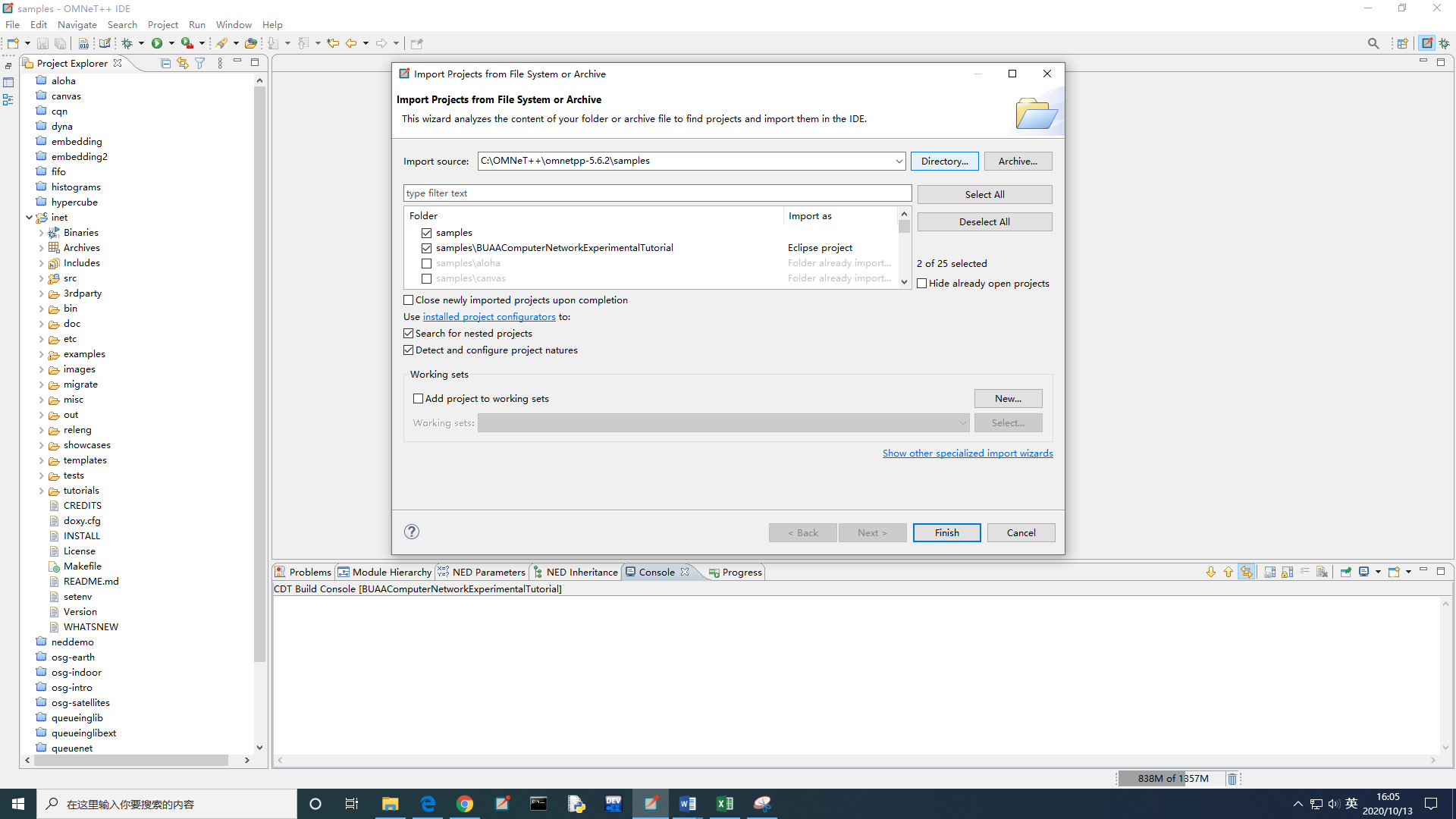
* **安装并打开BUAANetworkExperimentTutorial工程**

将解压后的工程文件复制到Workspace，点击File 🡪 Open Projects from File System。



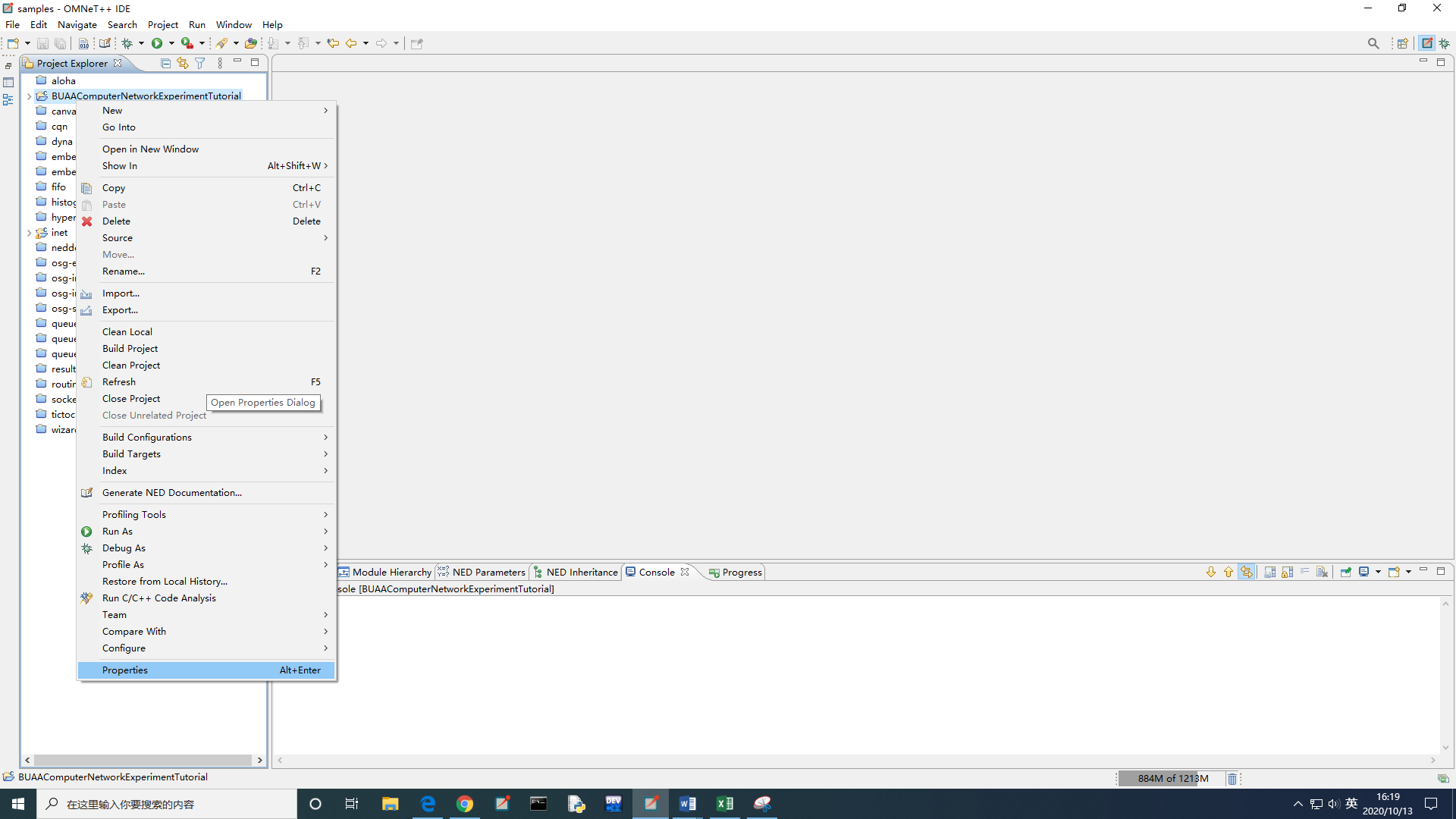
**图0.6 打开BUAANetworkExperimentTutorial工程的菜单选项1**

Directory选择Workspace所在的文件夹，勾选工程文件，点击Finish。



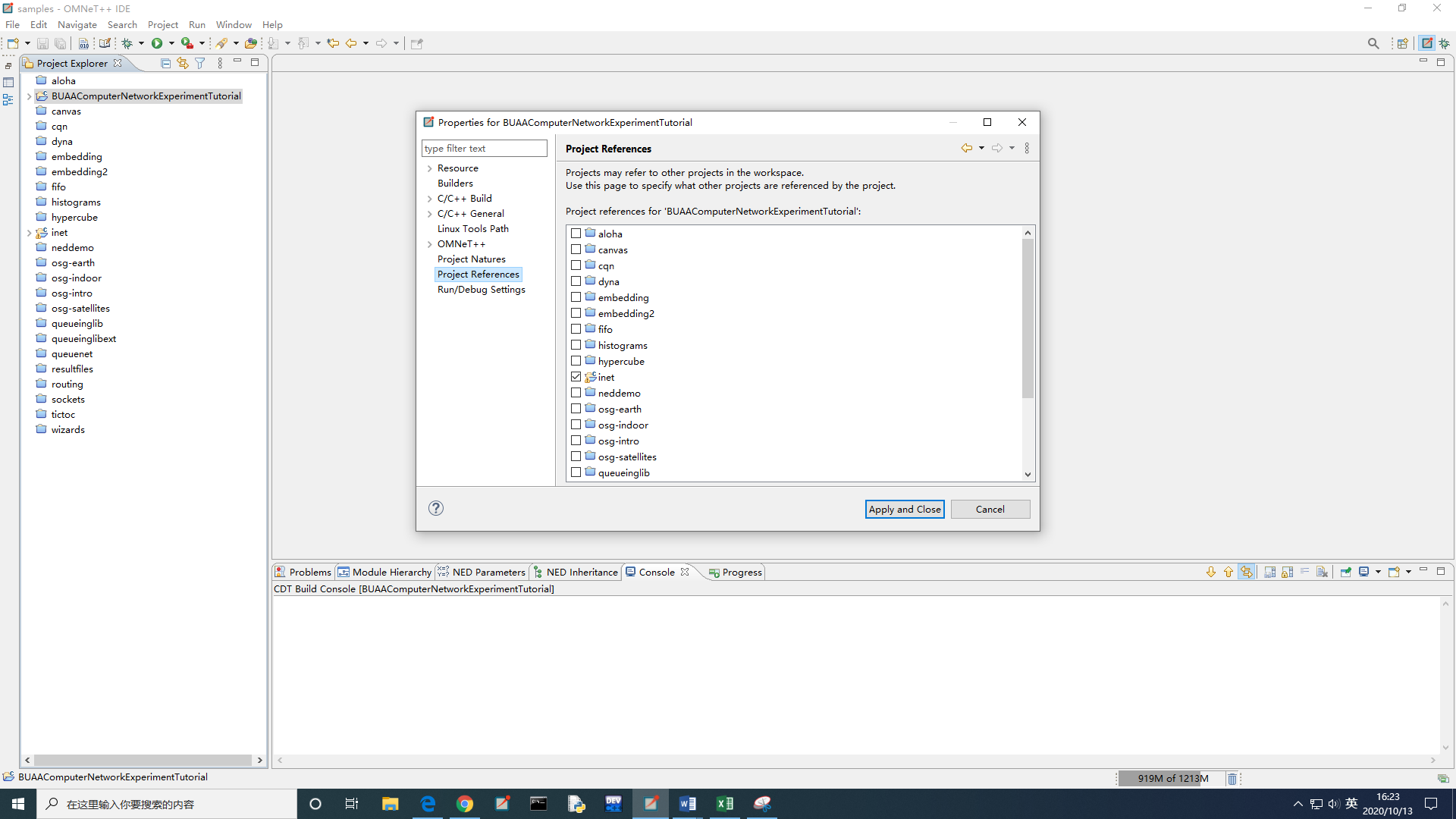
**图0.7 打开BUAANetworkExperimentTutorial工程的菜单选项2**

此时工程配置还未完成，需要配置依赖关系。右键工程，点击Properties。



**图0.8 打开BUAANetworkExperimentTutorial工程的菜单选项3**

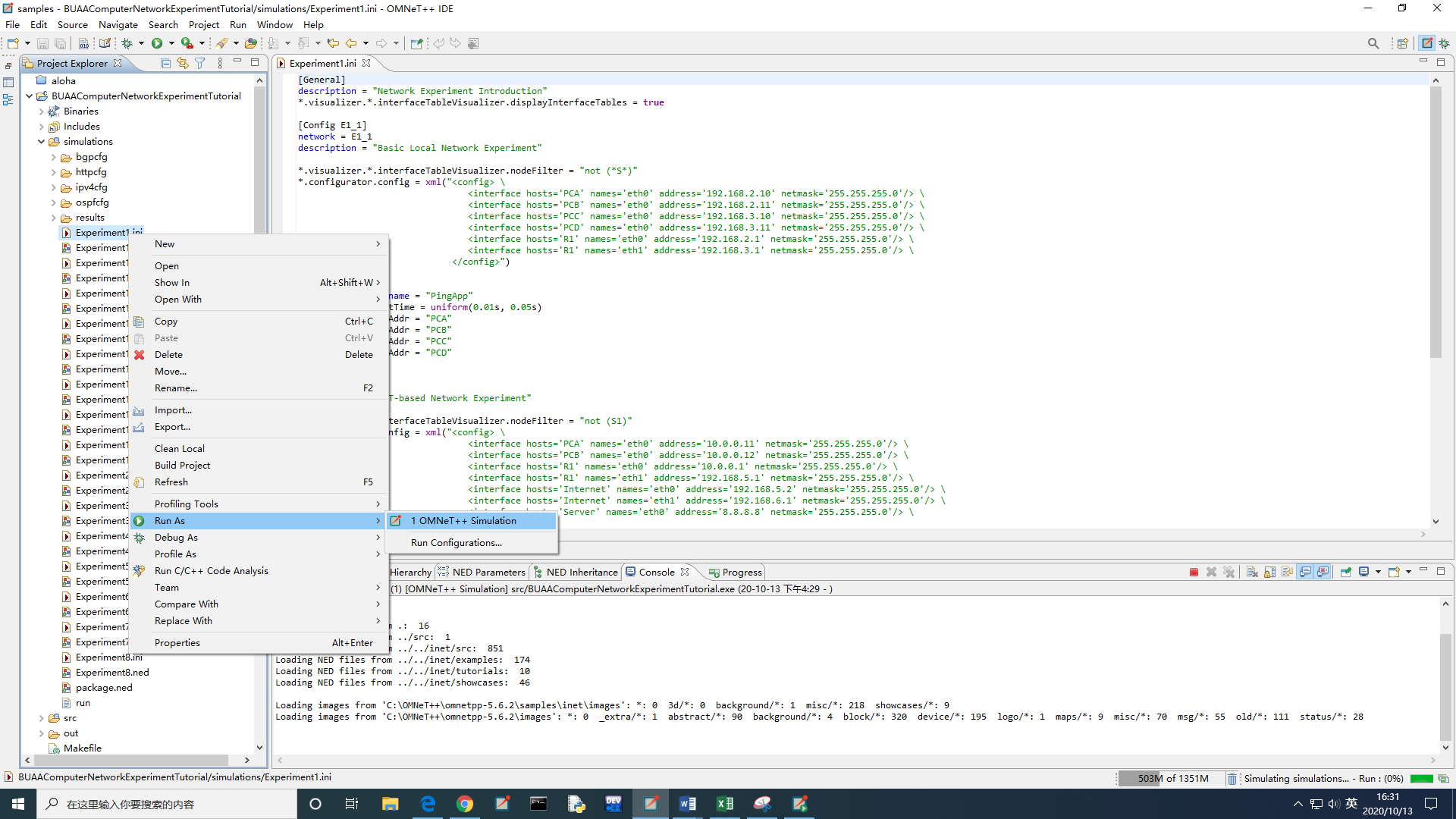
点击Project References 🡪 勾选之前导入的inet工程文件🡪点击Apply and Close。至此工程导入完成，Build All后应可以正常运行。



**图0.9 打开BUAANetworkExperimentTutorial工程的菜单选项4**

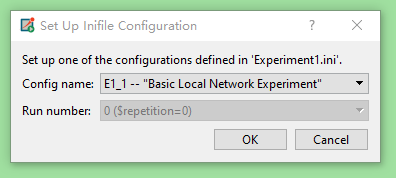
### 运行仿真

以第一个实验为例，右键点击.ini配置文件 🡪 Run As 🡪 OMNeT++ Simulation。



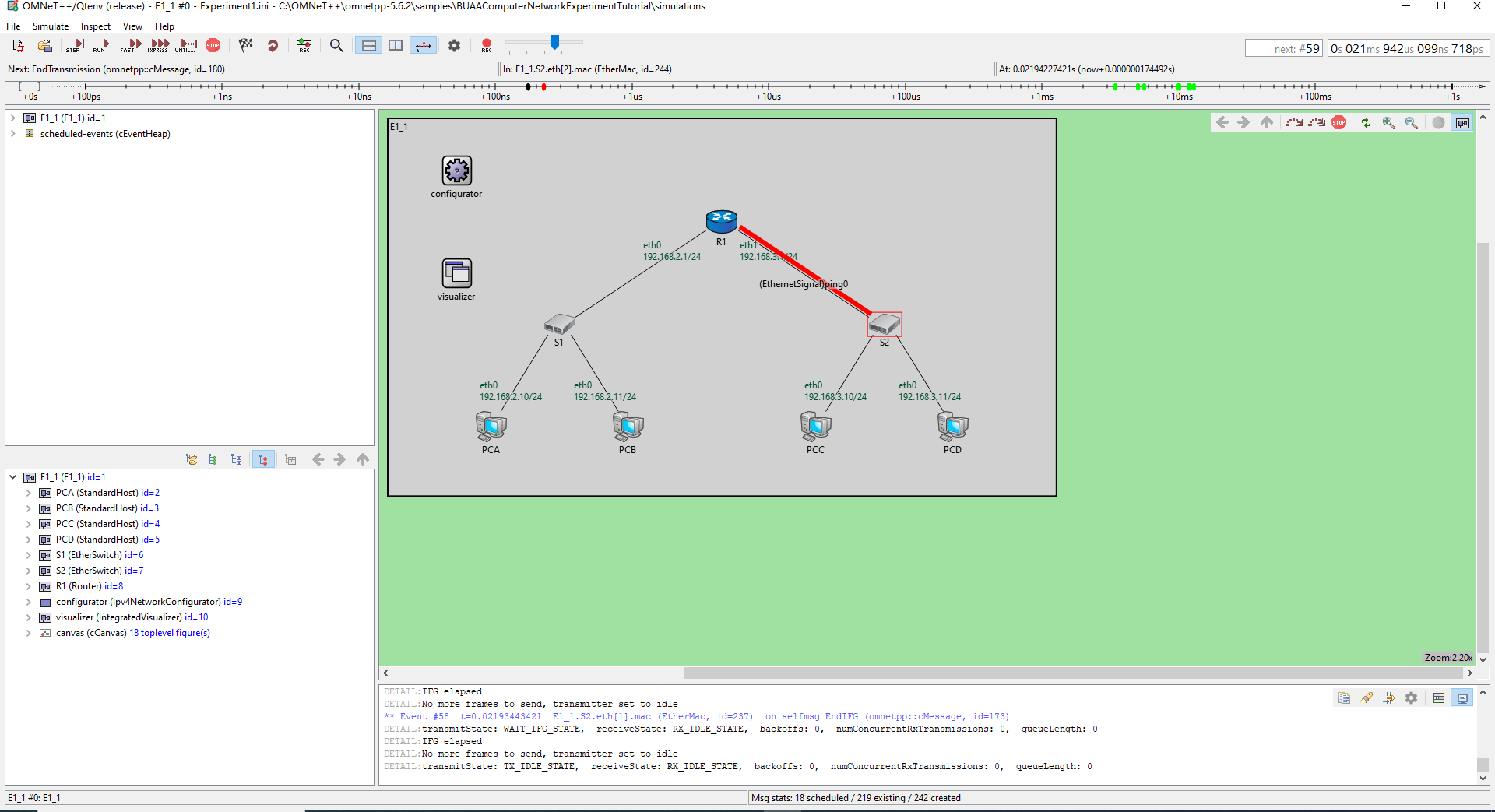
**图0.10 运行仿真的菜单选项1**

等待运行仿真的窗口弹出后选择对应的实验，此处为实验E1\_1，点击OK。



**图0.11 运行仿真的菜单选项2**

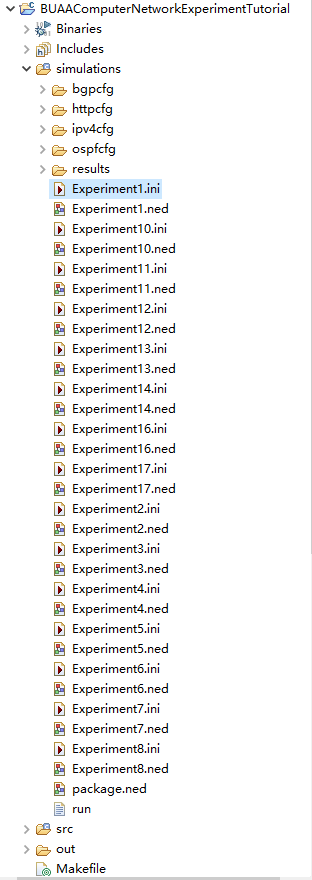
点击Run按钮开始运行仿真。



**图0.11 网络仿真运行截图**

### NED与INI文件

仿真运行由OMNeT++和INET框架提供运行环境和仿真模型（如StandardHost、Router等）。详细内容请参考官方文档。工程文件的目录结构如图所示



**图0.12 工程目录部分结构截图**

NED文件包含仿真网络拓扑和模型参数，INI编写的是仿真运行参数配置等。工程中实验网络拓扑与配置参数文件一一对应。每个文件都包含多个实验的拓扑或配置。

## 网络实验入门

本实验将介绍OMNeT++环境和INET框架的网络描述，模型类型，与基本配置方法。

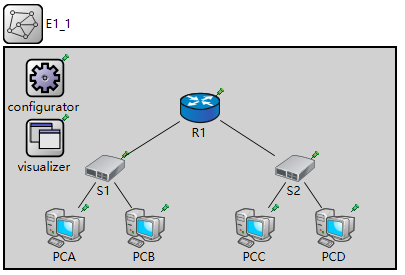
在NED文件中，实验的网络通过network声明，在submodules中添加模块，如StandardHost，EtherSwitch等，在connections中对模块进行连线。需要注意的是，configurator和visualizer模块为网络运行提供配置和信息可视化，后续的实验网络中都默认这两个模块的存在。在INI文件中，对网络中的节点进行仿真的配置，例如配置接口IP地址信息，路由和App的目的地址等。需要注意的是，由于受INET设备功能所限，以及考虑到编码的简洁，仿真网络拓扑并不是完全按照教材中的组网编写的，会有细微的差别，例如会有设备的接口的编号不同，接口的IP地址信息不同等。

**本章涉及的有关配置：**

* 运用Visualizer显示接口信息；
* 运用Configurator配置网络接口；
* 配置PingApp；
* 配置InternetCloud（模拟传输时延和丢包的模型）；
* 配置NAT；
* 配置TcpSessionApp和TcpSinkApp。

### 简单局域网组建实验

参考《计算机网络实验教材》（后面统一简称教材）P28页组网图。网络拓扑为Experiment1.ned中的network E1\_1。拓扑如图所示。



**图1.1 实验1.1网络拓扑图**

本实验为简单局域网组建实验，运行后可观察到四台主机分别向其他主机发送Ping数据包。

**有关配置说明**：

配置支持通配符‘\*’和‘\*\*’。详情参考OMNeT++官方文档《SimulationManual》

* \*.visualizer.\*.interfaceTableVisualizer.nodeFilter = "not (\*S\*)"

过滤交换机上接口的空白信息，不做显示。

* \*.configurator.config = xml("<config> \

配置主机和路由器的接口信息

* \*.PC\*.numApps = 4

在所有主机上运行的App数量为4

* \*.PC\*.app[\*].typename = "PingApp"

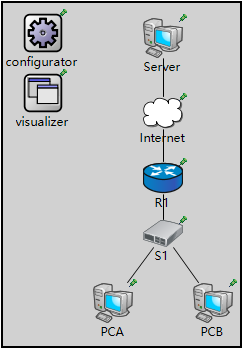
所有主机运行的所有App类型为PingApp

* \*.PC\*.app[0].destAddr = "PCA"

指定App的目的地址。

### 基于地址转换的组网实验（NAT功能仍需调试）

参考教材P34页的组网图。网络拓扑为Experiment1.ned中的network E1\_2。拓扑如图所示。



**图1.2 实验1.2网络拓扑图**

本实验为基于网络地址转换（NAT）的组网实验。PCA和PCB上运行着TcpSessionApp，会以固定的时间间隔向Server节点发送Tcp数据包。拓扑中的InternetCloud模块可模拟网络链路时延和丢包。

**有关配置说明：**

* \*.Internet.ipv4Delayer.config = xml("<internetCloud symmetric='true'> \

InternetCloud模块的相关配置，可根据源地址目的地址，配置随机时延，丢包等。详细情况可参考inet/example/internetcloud中的相关示例。

* \*.R1.ipv4.natTable.config = xml("<config> \

路由器R1中的NAT表配置，INET框架中的NAT功能是基于hook实现的。

* \*.PC\*.app[\*].typename = "TcpSessionApp"

在两台主机上配置TcpSessionApp，此App会在建立Tcp连接后向目的地址端口发送Tcp数据包。

* \*.PC\*.app[\*].connectAddress = "Server"

配置TcpSessionApp的目的地址。

* \*.PC\*.app[\*].connectPort = 1234

配置TcpSessionApp的目的端口。

* \*.PCA.app[\*].sendBytes = 300kB

配置TcpSessionApp要发送的内容总长度。

* \*.Server.app[\*].typename = "TcpSinkApp"

在服务器节点上配置TcpSinkApp。

* \*.Server.app[\*].localPort = 1234

## 数据链路层实验

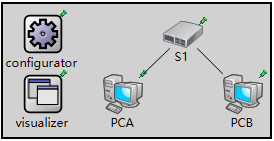
本实验介绍数据链路层相关的网络构建和配置方法。Ini文件支持全局配置，即[General]下的配置，这些配置适用在同一个文件中其他任何具体的仿真配置。

**本章涉及的有关配置：**

* 随机化配置；
* Vlan划分配置方法；
* Configurator默认路由、子网路由和静态路由配置；
* 使用pcapRecorder捕获报文。
* NED文件的extends特性

### 以太网帧格式分析

参考教材P40页的组网图。网络拓扑为Experiment2.ned中的network E2\_1。拓扑如图所示。



**图2.1 实验2.1网络拓扑图**

本实验为以太网帧格式分析，主机A和B上分别运行目的地址为对方的PingApp，通过在主机A和B上运行pcapRecorder来截获报文并分析。

**有关配置说明：**

* \*\*.fcsMode = "computed"
* \*\*.crcMode = "computed"

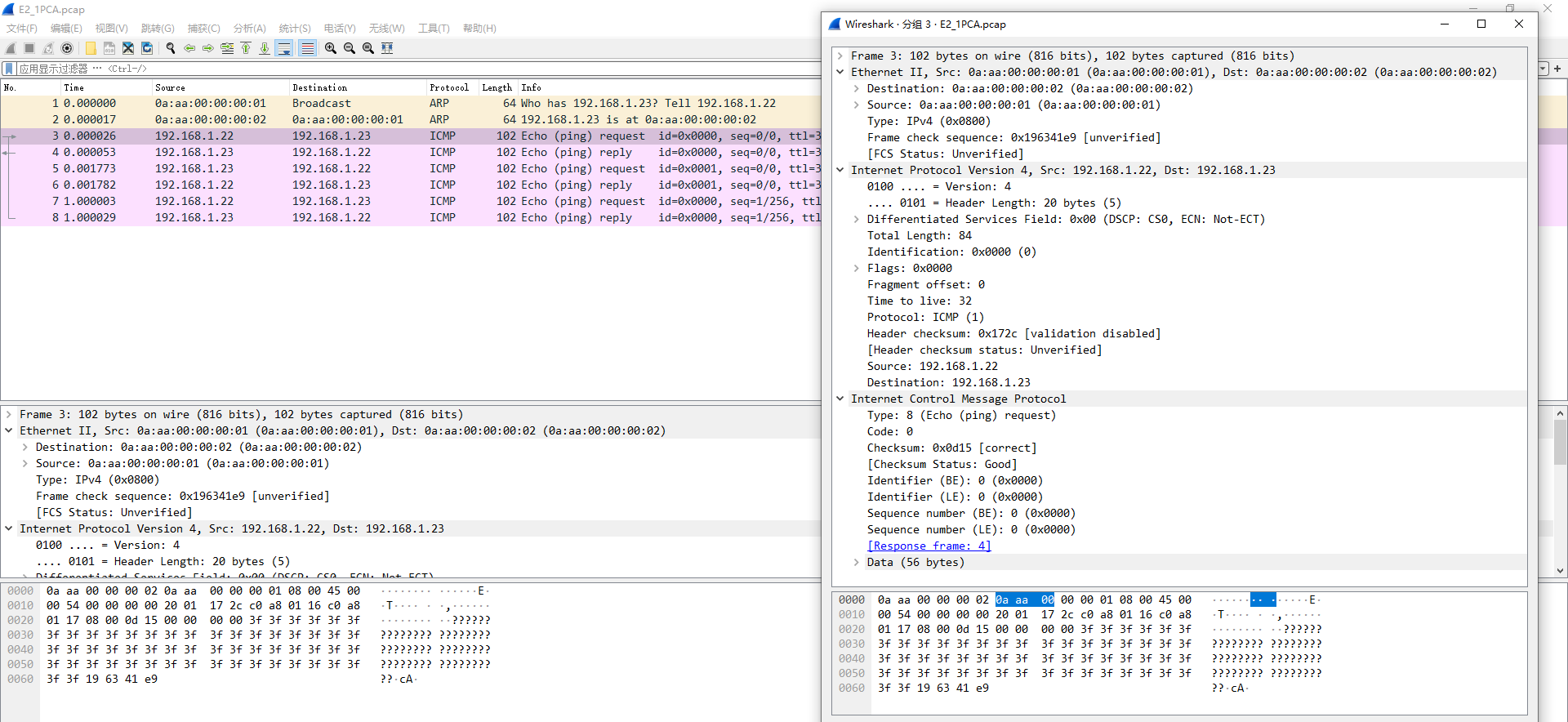
FCS帧校验序列和CRC循环冗余校验模式设为computed，否则运行过程中会报错。

* \*.PC\*.numPcapRecorders = 1

设置主机的pcapRecorder数量为1

* \*.PCA.pcapRecorder[\*].pcapFile = "results/E2\_1PCA.pcap"

保存相应抓包结果，路径为results/。

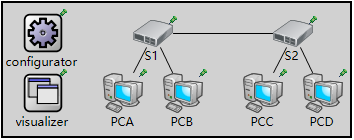


**图2.2 抓包内容示意图**

如图2.2所示，使用Wireshark软件打开捕获到的数据包文件，可以查看仿真网络运行期间的数据包详细信息。

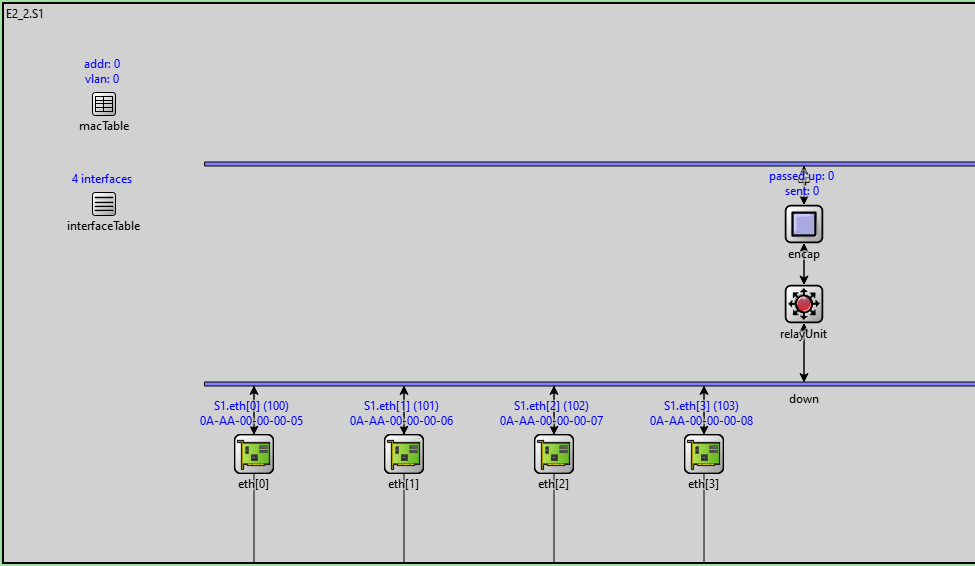
### 以太网帧格式分析

参考教材P44页的组网图。网络拓扑为Experiment2.ned中的network E2\_2。拓扑如图所示。



**图2.3 实验2.2网络拓扑图**

本实验为MAC地址表查看和广播风暴的演示实验，仿真网络暂时还没有实现端口聚合功能。需要注意的是，图中S1和S2之间有两条链路，但由于OMNeT++IDE界面展示的设计问题，两节点之间的多条链路目前只能以一条线的形式展示。在仿真网络开始运行后，双击S1节点如图所示，可以看到节点内部的模块层次结构和以太接口的MAC地址等信息。

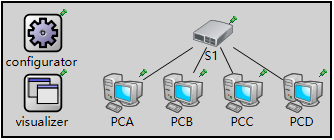


**图2.3 交换机内部层次结构截图**

随着仿真的运行，可以观察到交换机S1和S2频繁的进行ARP报文交互，即形成所谓的广播风暴。

### VLAN配置与分析（VLAN基本配置）

参考教材P53页组网图。网络拓扑为Experiment2.ned中的network E2\_3。拓扑如图所示。



**图2.4 实验2.3网络拓扑图**

本实验为Vlan配置与分析（VLAN基本配置）实验。INET框架中的Vlan划分方法是通过配置IEEE 802.1Q协议来实现的。运行仿真后四台主机分别会向其他主机发送数据包，但只有PCA、PCB之间和PCC、PCD之间可以正常通信。

**有关配置说明：**

* \*.\*.eth[\*].qEncap.typename = "Ieee8021qEncap"

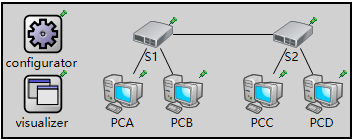
配置所有eth接口的封装类型为Ieee8021q

* \*.S1.eth[0].qEncap.inboundVlanIdFilter = "-1"
* \*.S1.eth[0].qEncap.inboundVlanIdMap = "-1 2"
* \*.S1.eth[0].qEncap.outboundVlanIdFilter = "2"
* \*.S1.eth[0].qEncap.outboundVlanIdMap = "2 -1"

通过过滤出入接口流量和为出入接口流量进行Vlan标签映射，实现将S1的eth0接口划分至Vlan2中。

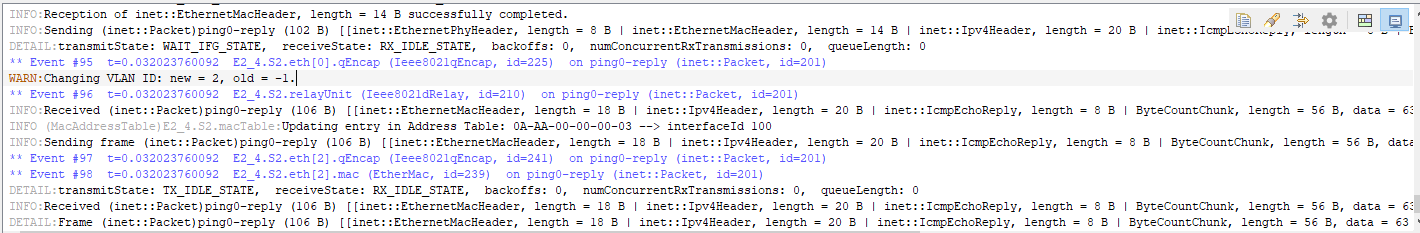
### VLAN配置与分析（Trunk和Hybrid配置）

参考教材P54页的组网图。网络拓扑为Experiment2.ned中的network E2\_4。拓扑如图所示。



**图2.5 实验2.4网络拓扑图**

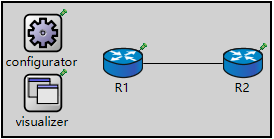
本实验为VLAN配置与分析（Trunk和Hybrid配置）实验。实验拓扑的声明方式采用继承，可以在继承的拓扑描述中添加新的组件和连接构建新的网络实验拓扑。Trunk配置中，S1和S2连接的端口允许带有VLAN标签2或3的数据包通过，使得跨交换机之间的VLAN可以相互通信。与Trunk端口配置不同的是，Hybrid端口配置使得VLAN3数据包在从一个交换机发出，到另一个交换机接收的过程中，数据包不带有VLAN id标签。有关Vlan 标签的变化信息可以在仿真运行界面的日志窗口中查看。



**图2.6 仿真界面日志窗口中的VLAN ID变化信息**

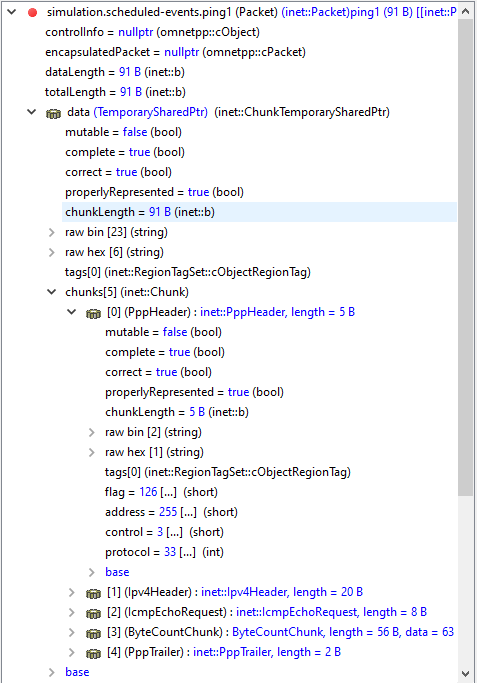
### 广域网数据链路层实验

参考教材P61页的组网图。网络拓扑为Experiment2.ned中的network E2\_5。拓扑如图所示。



**图2.7 实验2.5网络拓扑图**

本实验是广域网数据链路层实验，R1和R2之间采用ppp协议的方式进行连接，运行仿真的过程中，可以点击任意传输中的数据包查看相应的字段，例如ppp头部，如图所示。



**图2.8 数据包相关字段的查看位置截图**

## 网络层实验

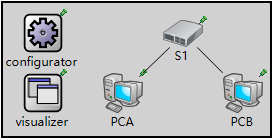
本实验介绍网络层的网络构成与配置方法。

本章涉及的有关配置：

* 设置ethernet接口的MTU（最大传输单元）；
* Configurator的默认路由、子网路由和静态路由配置项；
* Configurator的xml格式静态路由配置。

### ARP分析

参考教材P67页的组网图，网络拓扑为Experiment3.ned中的network E3\_1\_1和network E3\_1\_2。拓扑如图所示。

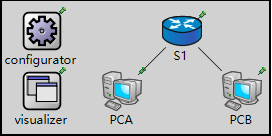


**图3.1 实验3.1网络拓扑图**

本实验为Arp分析实验，由于INET框架中的EtherSwitch为二层设备，与计算机网络实验中提供的三层交换机不同，故实验中的不同网段的ARP实验通过路由器来实现。仿真开始后PCA和PCB两台主机会分别向对方发送Ping数据包，在发送前，二者都会先解析通过ARP协议解析目的IP地址。仿真运行结束后使用Wireshark软件查看/results目录下的E3\_1\_1PCA.pcap和E3\_1\_2PCA.pcap。

### ICMP分析

参考教材P74页的组网图。网络拓扑为Experiment3.ned中的network E3\_1\_2。拓扑如图所示。

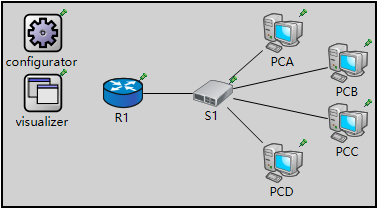


**图3.2 实验3.2网络拓扑图**

OMNeT++平台支持对于NED文件中的network结构的复用来简化代码，即使用同一网络拓扑多组不同的网络配置进行仿真。本实验为ICMP分析实验。仿真开始后，主机会相互发送Ping数据包。仿真运行结束后使用Wireshark软件查看/results目录下的E3\_2PCA.pcap。

### 网络层分片实验

参考教材81页的组网图。网络拓扑为Experiment3.ned中的network E3\_3。拓扑如图所示。



**图3.3 实验3.3网络拓扑图**

本实验为网络层分片实验。通过在交换机的相应Ethernet接口上配置最大传输单元实现对数据包内容的分片。

**有关配置说明：**

* \*.R1.eth[\*].mac.mtu = 100B

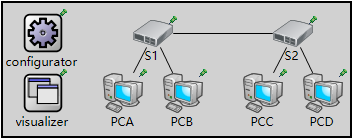
设置路由器R1的Ethernet接口最大传输单元为100B

* \*.R1.app[\*].packetSize = 300B

设置PingApp的数据包大小为300B

### VLAN间通信

参考教材P84页组网图。网络拓扑为Experiment3.ned中的network E3\_3。拓扑如图所示。



**图3.4 实验3.4网络拓扑图**

本实验为Vlan间通信实验。运行仿真后注意观察PCC向PCD发送的Ping数据包和PCD回复PCC的Ping数据包的流向。在默认情况下，Configurator会配置使得全网互通的路由，这会影响例如路由协议实验的进行，所以在一些实验配置中会关闭有关路由配置项。

**有关配置说明：**

* <route hosts='PCA' destination='\*' gateway='S1' interface='eth0' metric='60'/> \

路由配置，详细情况请阅读INET Framework官方文档有关Ipv4Configurator的章节。

* \*.configurator.addStaticRoutes = **false**
* \*.configurator.addSubnetRoutes = **false**
* \*.configurator.addDefaultRoutes = **false**

Configurator的配置项，关闭静态理由，子网路由和默认路由的配置。

## 传输层实验

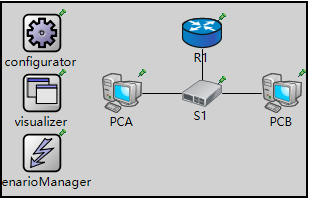
本实验介绍传输层协议如TCP和UDP的配置方法。

**本章涉及的有关配置：**

* NED文件中的ThruputMeteringChannel子类链路声明；
* 仿真时间控制；
* Visualizer中的丢包可视化配置；
* TCP滑动窗口相关配置；
* Tcp算法类型配置；
* Tcp最大报文段长度配置；
* ScenarioManager场景变化控制配置；
* Tcp协议Nagel算法配置。

### TCP基本分析

参考教材P101页组网图。网络拓扑为Experiment4.ned中的network E4\_1。拓扑如图所示。



**图4.1 实验4.1网络拓扑图**

需要注意的是，图中S1和R1中间有两条连线。为了更好地展示传输层的实验效果，本实验将在链路中引入时延。实验中使用ThruputMeteringChannel作为父类的链路C构建网络。链路C声明如下：

**types**:

**channel** C **extends** ThruputMeteringChannel{

**parameters**:

datarate = 100Mbps;

delay = 5ms;

per = 0;

ber = 0;

thruputDisplayFormat = **default**("p b u");

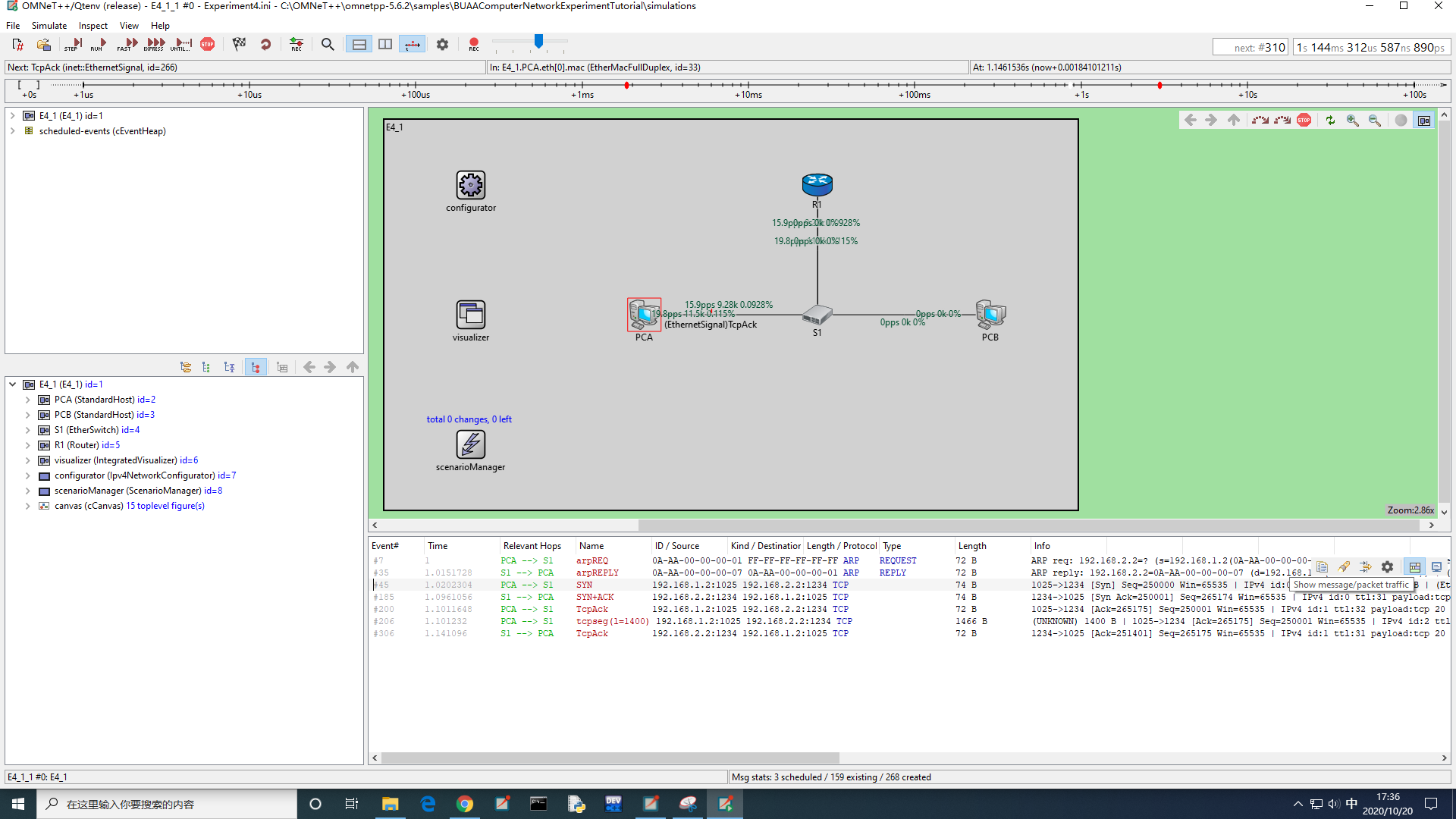
}

有关thruputDisplayFormat参数的整理（参考inet.common.misc.ThruputMeteringChanan-el.cc）：

* ‘N’：显示数据包的数量
* ‘V’：数据量（以字节为单位）
* ‘p’：当前数据包速率（pkt/sec）
* ‘b’：当前带宽
* ‘u’：链路利用率
* ‘P’：平均数据包速率（从开始到当前仿真时间）
* ‘B’：平均带宽（从开始到当前仿真时间）
* ‘U’：平均链路利用率（从开始到当前仿真时间）

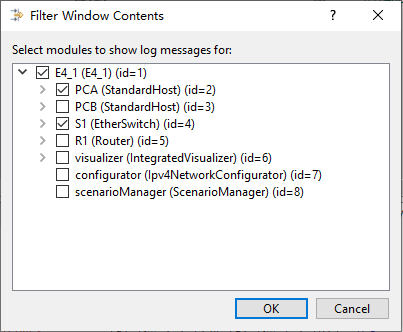
#### 滑动窗口与窗口侦察

本实验是滑动窗口和拥塞控制。网络拓扑为Experiment4.ned中的network E4\_1。仿真运行后PCA会与PCB建立Tcp连接，随后发送数据包。点击日志窗口的显示消息或数据包按钮可以查看网络接收和发送数据包的情况。

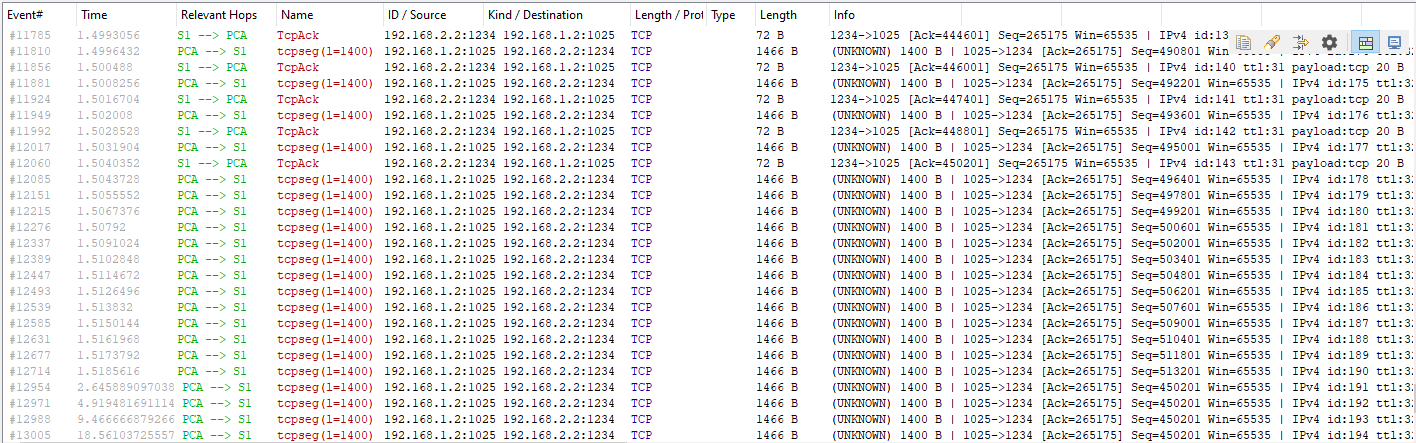


**图4.2 日志窗口流量视图截图**

点击旁边的过滤图标（左数第三）可以实现对流量内容的过滤，本实验中可以勾选PCA和S1来查看相关的流量信息。



**图4.3 流量内容过滤窗口**



**图4.4 过滤后的流量内容截图**

**有关配置说明：**

* \*\*.tcp.windowScalingSupport = **true**

滑动窗口支持

* \*\*.tcp.advertisedWindow = 65535

窗口通告，以字节为单位

* \*\*.tcp.timestampSupport = **true**

支持数据包时间戳

* \*\*.tcp.tcpAlgorithmClass = "TcpReno"

使用TcpReno算法（包括慢启动、拥塞避免、快送重传和快速恢复）

* \*\*.tcp.typename = "Tcp"

（这个暂时不知道啥用）

* \*\*.tcp.mss = 1400

最大报文段长度（默认）

* \*\*.arp.retryTimeout = 1s

ARP超时重传时间设为1s

* \*\*.arp.retryCount = 3

ARP超时重试次数为3

* \*\*.arp.cacheTimeout = 100s

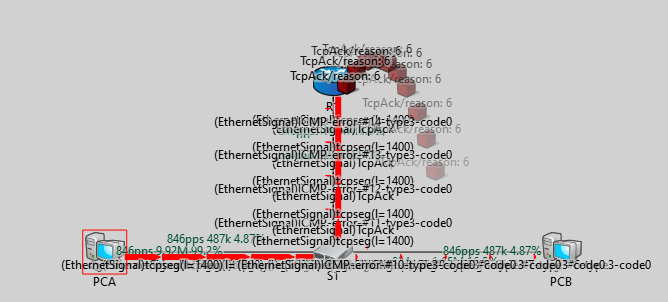
ARP缓存时间设为100s

* sim-time-limit = 5.0s

仿真的运行总时间限制为5s

#### 慢启动、拥塞避免及拥塞处理和超时与重传

本实验为TCP慢启动、拥塞避免及拥塞处理和超时与重传机制。网络拓扑为Experiment4.ned中的network E4\_1。在上一节实验的基础上，本节实验在ScenarioManager中添加了控制设备启停的配置项，以及使用visualizer中的packetDropVisualizer进行丢包的可视化。经过测试，实验网络中的VLAN划分配置（qEncap）会导致ARP类型数据包丢包可视化时产生运行错误。故在运行中需要将相关ARP类型的数据包过滤掉，不做可视化。



**图4.5 丢包可视化效果示意图**

**有关配置说明：**

* \*.visualizer.\*.packetDropVisualizer.displayPacketDrops = **true**

在Visualizer中启用显示丢包

* \*.visualizer.\*.packetDropVisualizer.packetFilter = "not arp\*"

过滤arp开头的数据包（包括arpREQ和arpREPLY等）

* \*.visualizer.\*.packetDropVisualizer.labelFormat = "%n/reason: %r"

丢包动画效果的标签格式，%n代表数据包名，%r代表原因

* \*.visualizer.\*.packetDropVisualizer.fadeOutTime = 3s

丢包动画效果的淡出时间（真实时间）

* \*\*.scenarioManager.script = xml( \

场景管理脚本配置项（xml格式）

* "<at t='1.5s'><shutdown module='R1.eth[0]'/></at>\n"

1.5秒关闭R1 eth0

* "<at t='20.5s'> <startup module='R1.eth[0]'/></at>\n"

20.5秒再开启R1 eth0

#### 快重传与快恢复

本实验为TCP快重传和快恢复机制。网络拓扑为Experiment4.ned中的network E4\_1\_3。教材中通过内核丢包的方式出发快重传快恢复机制，与network E4\_1不同的是，E4\_1\_3将PCA与S1之间的链路改为了一条含有数据包错误率的链路，从而达到类似的效果。错误率链路代码声明如下。其中per为数据包错误率（packet error rate）。

**channel** C\_err **extends** ThruputMeteringChannel{

**parameters**:

datarate = 10Mbps;

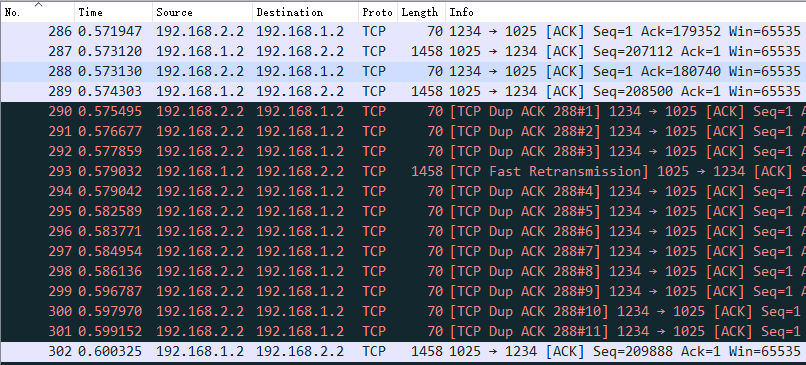
delay = 5ms;

per = 0.02;

ber = 0;

thruputDisplayFormat = **default**("p b u");

}



**图4.6 实验4.1.3流量捕获截图**

#### 糊涂窗口与Nagel算法

本实验为糊涂窗口与Nagel算法。网络拓扑为Experiment.ned中的networkE4\_1。本实验中用到TcpSessionApp中的发包脚本功能。TcpSessionApp支持在给定仿真时间和发送字节数的情况下，预配置发送数据包。tcpscript中是一个用C语言写的小脚本用于生成sendScript。仿真运行结束后查看results目录下的E4\_1\_4PCA.pcap。可以尝试开启与不开启Nagle算法进行实验，对比两次实验的不同。

**有关配置说明：**

* \*.PCA.app[\*].sendBytes = 0

SendBytes设为0，因为默认情况不为零会导致sendscript无法使用。

* \*.PCA.app[\*].sendScript = "0.1 800;0.2 800;"

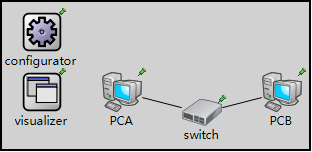
sendScript为字符串类型，格式为“<time> <numBytes>;”。

* \*\*.tcp.nagleEnabled = **true**

启用Tcp的Nagle算法。

### UDP分析

本实验为UDP分析。网络为拓扑为Experiment4.ned中的network E4\_2。拓扑如图所示。



**图4.7 实验4.2网络拓扑图**

**有关配置说明：**

* \*.PCA.app[\*].typename = "UdpBasicApp"

PCA应用程序类型为UdpBasicApp，模拟Udp传输的应用程序模块

* \*.PCB.app[\*].typename = "UdpSink"

PCB上运行UdpSink应用程序，用于接收来自PCA的数据包。

* \*.PCA.app[\*].sendInterval = 0.01ms

PCA发送Udp数据包的时间间隔

* \*.PCA.app[\*].messageLength = 1000Byte

PCA发送的数据包的Data字段长度。

## 应用层实验

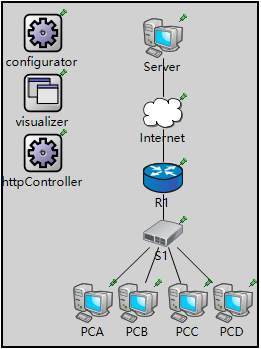
本实验介绍应用层协议HTTP和DHCP的配置方法。目前INET框架暂时不支持DNS，FTP和SMTP等教材上涉及的应用程仿真。

**本章涉及的有关配置：**

* HTTP Browser，Server和Controller相关配置；
* DHCP Client和Server相关配置。

### HTTP分析

本实验为HTTP分析实验。网络拓扑为Experiment5.ned中的network E5\_1。拓扑如图所示。



**图5.1 实验5.1网络拓扑图**

HTTP协议仿真通过INET框架提供的httptools实现。如采用PcapRecorders抓包会报出有关HttpRequestMessage无法序列化的错误，故实验暂时不支持抓包。可以通过日志窗口的流量视图对仿真中的数据包进行查看。

**有关配置说明：**

* \*\*.httpController.config = **xmldoc**("httpcfg/controller\_cfg.xml","//controller-profile[@id='uniform']")

httpController的相关配置，具体参考httpcfg目录下文件controller\_cfg.xml。

* \*.PC\*.app[\*].typename = "HttpBrowser"

PC应用程序类型设为HttpBrowser

* \*.PC\*.app[\*].httpProtocol = 11

协议类型为11

* \*.PC\*.app[\*].config = **xmldoc**("httpcfg/browser\_cfg.xml","//user-profile[@id='normal']")

HttpBrower的相关配置，参考httpcfg目录下的browser\_cfg.xml

* \*.Server.app[\*].typename = "HttpServer"

服务器应用程序类型设为HttpServer

* \*.Server.app[\*].hostName = "www.buaa.edu.cn"

按照教材设定hostName

* \*.Server.app[\*].port = 80

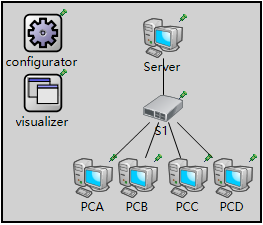
服务器的监听端口为80

* \*.Server.app[\*].config = **xmldoc**("httpcfg/server\_cfg.xml","//server-profile[@id='normal']")

服务器应用程序的相关配置，参考httpcfg文件夹下的server\_cfg.xml

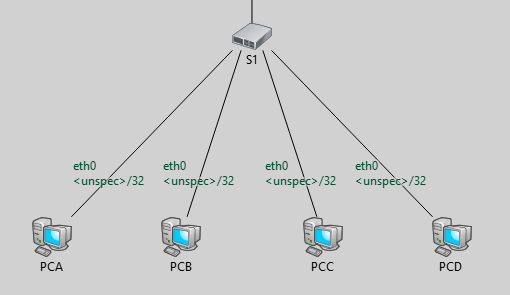
### DHCP分析

本实验为DHCP分析实验。网络拓扑为Experiment5.ned中的network E5\_2。拓扑如图所示。



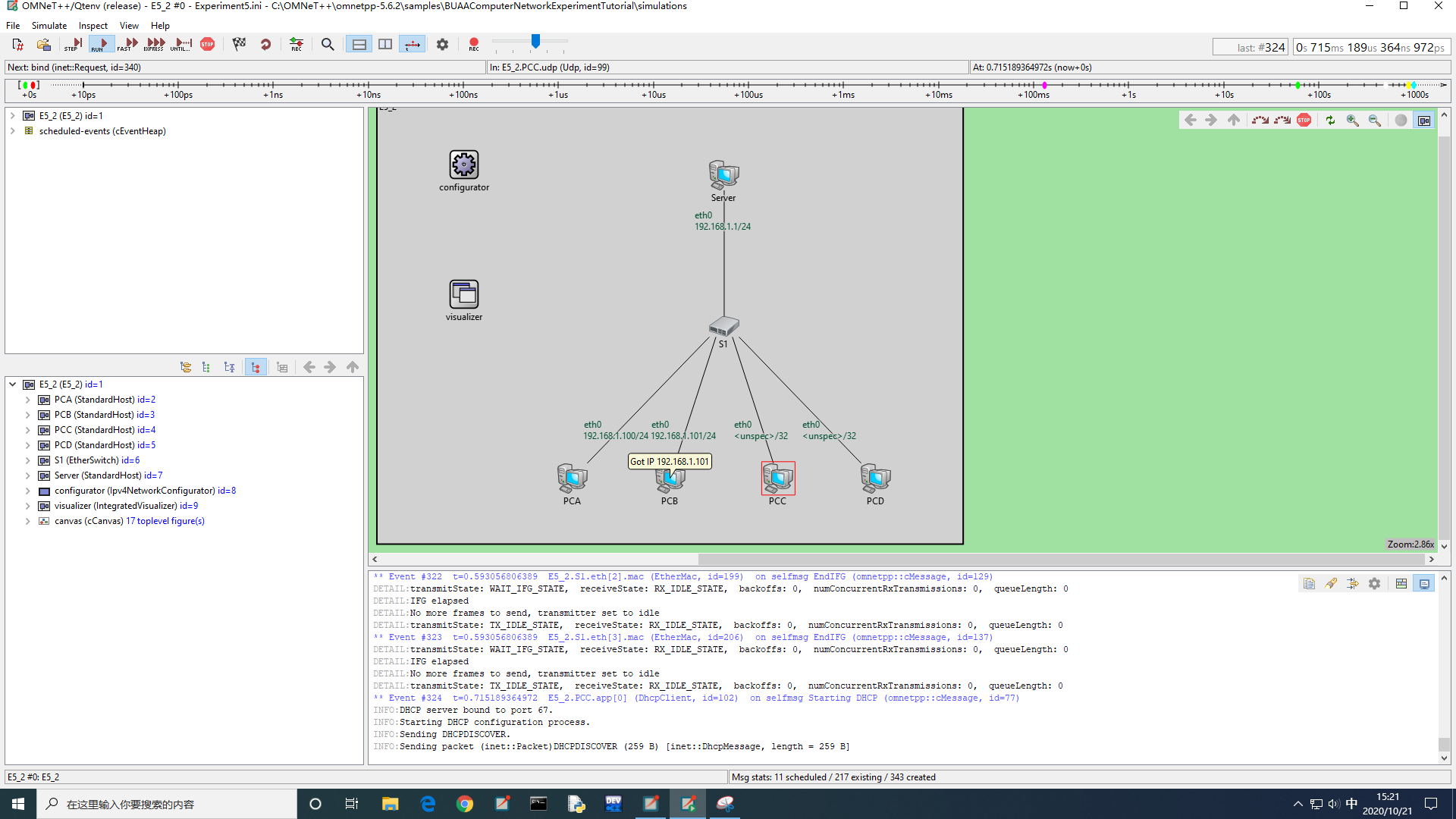
**图5.2 实验5.2网络拓扑图**

在仿真开始时，四台主机的接口地址信息都为<unspec>（未分配）。



**图5.3 主机未分配IP地址截图**

运行过程中DHCP服务器接收到来自客户端的请求，会为四台主机分配IP地址，当主机获得IP地址后，会弹出气泡提示。



**图5.4 主机分配IP地址时截图**

**有关配置说明：**

* \*.PC\*.app[\*].typename = "DhcpClient"

主机运行的App类型为DhcpClient

* \*.Server.app[\*].typename = "DhcpServer"

服务器运行的App类型为DhcpServer

* \*.Server.app[\*].numReservedAddresses = 100

DHCP服务器地址池大小为100

* \*.Server.app[\*].maxNumClients = 100

DHCP服务器支持的客户端数量

* \*.Server.app[\*].gateway = "192.168.1.1"

DHCP服务器网关地址为192.168.1.1

* \*.Server.app[\*].leaseTime = 1000s

地址分配的时间为1000s

## RIP实验

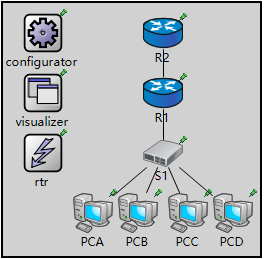
本章介绍路由器上的RIP相关协议的配置方法。

**本章涉及的相关配置：**

* 开启路由器的RIP功能；
* RouterID配置；
* RouterTableRecorder的使用与配置。

### 静态路由与默认路由配置

参考教材P152页组网图。拓扑为Experiment6.ned中的network E6\_1。拓扑如图所示。



**图6.1 实验6.1网络拓扑图**

本实验为静态路由与默认路由配置实验。由于INET对于三层交换机设备的不支持，故稍微调整组网图，以达到一样的效果。

**有关配置说明：**

* \*.configurator.dumpRoutes = **true**

在仿真运行开始前，输出设备上的路由信息。（供检查配置用）

### RIP配置和RIPv1报文结构分析

参考教材P157页的组网图。拓扑为Experiment6.ned中的network E6\_1。

本实验为RIP配置和RIPv1报文结构分析。本节中运用到一个信息记录模块RoutingTable-Recorder。该模块会记录仿真运行过程中网络拓扑中的路由与接口信息变化。仿真运行结束后，会在results目录下生成以网络名命名的.rt文件（例如本实验生成的文件，E6\_2-0.rt）。

文件内容如图所示。

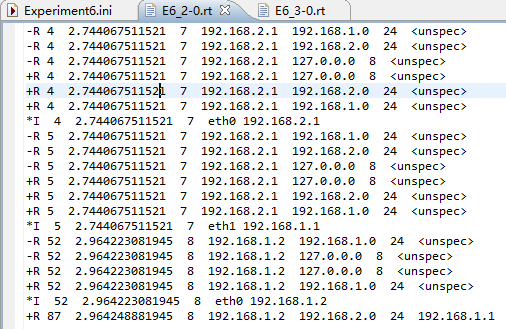


图6.2 E6\_2-0.rt文件内容截图

INET框架未给出文档，通过分析源码（inet/networklayer/ipv4/RoutingTableRecorder.cc）,可知每一列的内容为：行为（action）、事件序号（eventNo）、仿真时间（simtime）模块ID、（moduleId）、接口名（ifname）、地址(address)。其中行为包括一下6种类型：

* “+I”：接口创建
* “-I”：接口删除
* “\*I”：接口配置变更
* “+R”：路由添加
* “-R”：路由删除
* “\*R”：路由变更

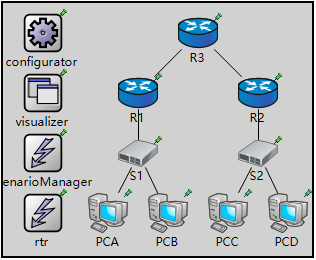
**有关配置说明：**

* \*.R\*.hasRip = **true**

在路由器上开启Rip协议。

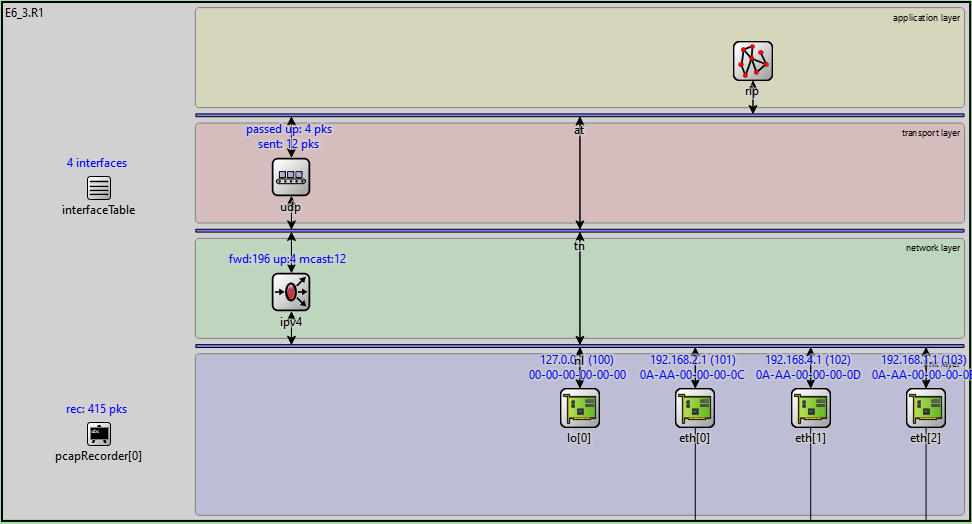
### 距离矢量算法分析

参考教材P161页组网图。拓扑为Experiment6.ned中的network E6\_3。拓扑如图所示。

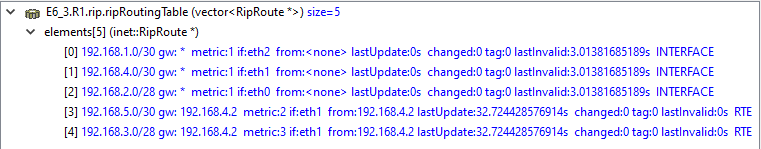


**图6.3 实验6.3网络拓扑图**

由于Ipv4NetworkConfigurator暂时不能对多个Loopback接口进行IP地址配置，所有添加的Loopback接口的IP地址都为127.0.0.1，lo0接口的IP地址由RouterID指定。在仿真运行中，可以点击任意路由器的中的rip模块查看有关Rip路由的信息。



**图6.4 路由器内部模块截图**



**图6.5 rip路由表信息截图**

在仿真结束后可以查看截取到的pcap文件以了解更多信息。

**有关配置说明：**

* \*.R1.\*\*.routerId = "192.168.1.1"

配置R1的lo0接口IP地址（routerID）为192.168.1.1

### 触发更新和水平分割

参考教材P166页组网图。拓扑为Experiment6.ned中的network E6\_3。

本实验为触发更新和水平分割。实验配置中采用了extends来继承上一节的拓扑和配置，缩小代码量。实验过程中，通过ScenarioManager控制R1的eth2接口的开关来实现Loopback的创建和删除。

**有关配置说明：**

* \*.R\*.rip.triggeredUpdate = **false**

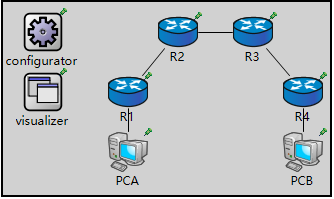
按实验要求关闭触发更新

* \*.R\*.rip.ripConfig = xml("<config><interface hosts='R\*' mode='NoSplitHorizon'/></config>")

将所有路由器接口的配置更改为无水平更新（也可将mode字段设为SplitHorizon或SplitHorizonPoisonedReverse等）

### 设计型实验2

参考教材P174页的组网图。拓扑为Experiment6.ned中的networkE6\_D2。拓扑如图所示。



**图6.6 设计型实验2网络拓扑图**

本实验为设计型实验2。由于INET不支持指定端口配置RIP协议，故按照相应配置组网后，全网自然互通。需要注意的是，由于配置Configurator不添加默认路由，故需要给主机配置一条目的地址为任意地址的默认路由以实现数据包发送。

**有关配置说明：**

* <route hosts='PCA' destination='\*' gateway='R1' interface='eth0' metric='60'/> \

Configurator不配置默认路由，故需要在主机上手动配置，以支持数据包正常发送。

## OSPF协议实验

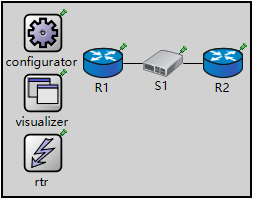
本章介绍路由器上的OSPF协议的配置方法。

**本章涉及的相关配置：**

* ospf配置文件指定
* ospf地址范围和接口配置
* routingTableVisualizer路由可视化配置
* networkRouteVisualizer数据包路径可视化配置

### OSPF协议概述及基本配置

参考教材P180页的组网图。拓扑为Experiment7.ned中的E7\_1，配置为Experiment7.ini中的[Config E7\_1]。拓扑如图所示。



**图7.1 邻居建立与报文交换拓扑图**

本实验为OSPF协议概述与基本配置。实验在ospfcfg/E7\_1-2Config中配置了R1，R2两个OSPF路由器的ospf协议配置，包括指定地址范围和OSPF接口声明等。详细配置请参考Ospfv2.ned模块的代码注释和官方文档。

有关配置说明：

* \*\*.ospf.ospfConfig = **xmldoc**("ospfcfg/E7\_1-2Config.xml")

指定所有OSPF路由器的配置为该文件；

* <Area id="0.0.0.0">

<AddressRange address="1.1.1.1" mask="255.255.255.255" />

</Area>

区域和地址范围配置（区域0，R1的lo0接口地址范围）；

* <Router name="R1">

<BroadcastInterface ifName="eth0" areaID="0.0.0.0" />

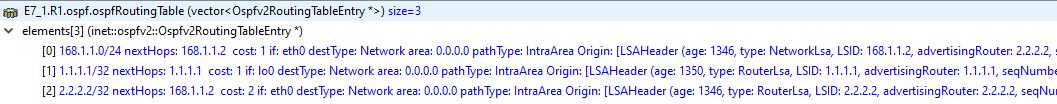
<LoopbackInterface ifName="lo0"/>

</Router>

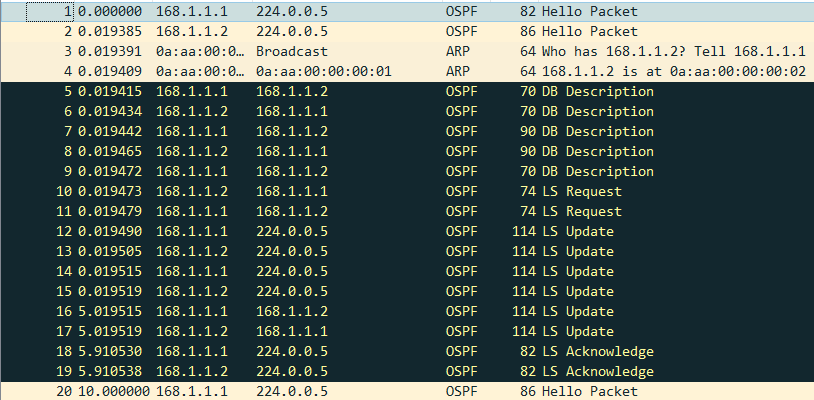
路由器配置，广播类型接口和loop接口配置。

### OSPF报文交互过程

本节继续上一节继续使用上一节的网络拓扑，配置为Experiment7.ini中的[Config E7\_2]。本节在上一节的基础上配置了R1上捕获数据包的配置。可以通过观察仿真实验的数据包交互过程和分析捕获的pcap文件了解ospf工作的原理。



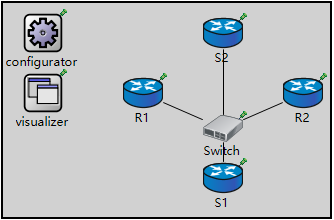
**图7.2 OSPF路由表截图**



**图7.3 OSPF报文交互过程截图**

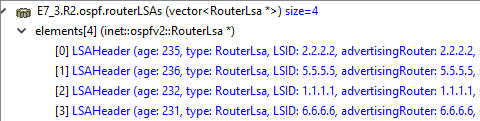
### OSPF协议的链路状态描述

参考教材P196页的组网图，实验网络拓扑为Experiment7.ned中的E7\_3，实验配置为[Config E7\_3]。拓扑如图所示。



**图7.4 实验7.3拓扑图**

本节实验为链路状态描述实验。INET框架中实现的OSPF协议的模块中只对以下四类LSA进行实现：RouterLSA、NetworkLSA、SummaryLSA和AsExternalLSA。实验中可在ospf模块中对相关信息进行观察分析。

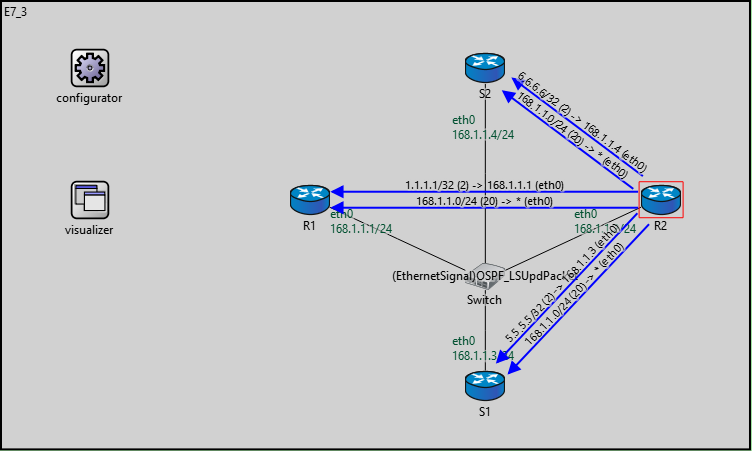


**图7.5 RouterLSA截图**



**图7.6 NetworkLSA截图**

实验中配置了路由信息可视化的内容，可直观地观察OSPF报文交互过程传递路由信息带来的变化。



**图7.7 路由信息可视化截图**

实验配置了S2关闭和重启的场景变化，可观察在接口信息中前后的DR/BDR路由器信息。



**图7.8 R2的OSPF接口检测到S2接口处于Down（仿真时间约130s）**



**图 7.9 重新选举后的DR和BDR信息（仿真时间约140s）**

**有关配置说明：**

* \*.visualizer.routingTableVisualizer[\*].displayRoutingTables = **true**

配置显示路由信息

* \*.visualizer.routingTableVisualizer[\*].labelFormat = "%d/%m (%e) -> %g (%n)"

路由信息标签的格式：目的网络/掩码（权值）-> 下一跳地址（接口）

* \*.visualizer.routingTableVisualizer[\*].lineColor = "blue"

配置路由信息标签的颜色

* \*.visualizer.routingTableVisualizer[\*].nodeFilter = "R2"

显示路由信息标签的节点过滤项（只显示R2的信息）

* \*\*.scenarioManager.script = xml("<script>\n

<at t='100s'><shutdown module='S2'/></at>\n

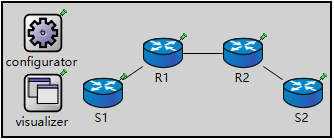
<at t='150s'> <start module='S2'/> </at>\n

</script>"

配置S2节点的关闭和再启动

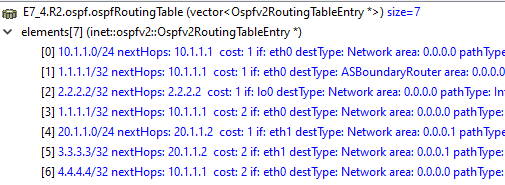
### 区域划分以及LSA种类

参考教材P205页的组网图，实验拓扑为Experiment7.ned中的E7\_4，配置为Experiment.ini中的[Config E7\_4]。拓扑如图所示。



**图7.10 实验7.4拓扑图**

实验与教材略有不同，将R1、R2和S2的Lo0接口地址（RouterID）配置到相应的区域。配置R1与S1相连的接口为OSPF外部接口。实验中可以观察到有关ABR和ASBR的信息，AsExternalInterface的相关信息。请注意，AsExternalLSA信息会在1800s后超时，导致R2和S2上有关S1的路由和链路状态随之超时。



**图7.11 R2的OSPF路由表信息**

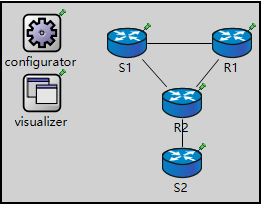
**有关配置说明：**

* <ExternalInterface ifName="eth0" advertisedExternalNetworkAddress="4.4.4.4" advertisedExternalNetworkMask="255.255.255.255" externalInterfaceOutputType="Type1" />

外部接口路由配置

### OSPF协议路由的计算

参考P215页的组网图，实验拓扑为Experiment7.ned的E7\_5，配置为Experiment.ini中的[Config E7\_5]。拓扑如图所示。



**图7.12 实验7.5拓扑图**

实验在接口上配置了Cost值，通过路由信息标签展示路由计算的结果，如下图所示。



**图7.13 路由计算信息展示**

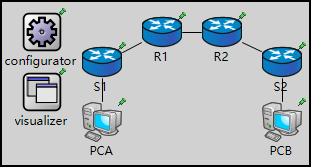
**有关配置说明：**

* <BroadcastInterface ifName="eth0" areaID="0.0.0.0" interfaceOutputCost="100"/>

interfaceOutputCost定义出接口路由的Cost值。

### 设计型实验1

参考教材P227页的组网图。拓扑为Experiment7.ned，配置为Experiment7.ini中的[Config E7\_D1]。拓扑如图所示。



**图7.14 设计实验1网络拓扑图**

实验在各个网络设备上配置了指向主机所在网段的静态路由。实验配置了路由信息展示，对手动配置的静态路由进行可视化，并且配置了网络数据包路径展示对联通性进行展示，可视化效果如下图所示。



**图7.15 设计实验2运行界面截图**

**有关配置说明：**

* \*.visualizer.numNetworkRouteVisualizers = 1

配置visualizer中数据包路径展示模块数为1

* \*.visualizer.networkRouteVisualizer[0].displayRoutes = **true**

显示数据包路径

* \*.visualizer.networkRouteVisualizer[0].packetFilter = "ping\*"

展示名称以“ping”开头的数据包路径

* \*.visualizer.networkRouteVisualizer[0].fadeOutTime = 3s

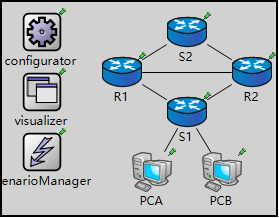
路径展示标签的淡出的时间为3s

* \*.visualizer.networkRouteVisualizer[0].lineColor = "yellow"

路径展示标签的颜色为黄色

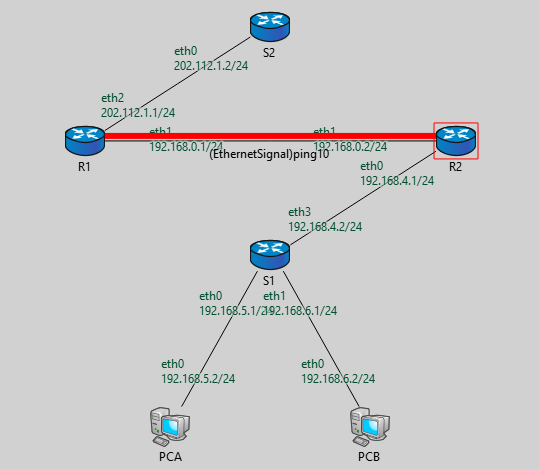
### 设计型实验2

参考教材P228页的组网图。拓扑为Experiment7.ned中的E7\_D2，配置为Experiment.ini中的[Config E7\_D2]。拓扑如图所示。



**图7.16 设计实验2网络拓扑图**

实验配置了满足实验要求的静态路由，并调整了路由的权值用作路由备份。实验通过ScenarioManager配置了特定在时间断开S1、R1间和S2、R2间的链路，仍然可以使得网络正常通信，如图所示。



**有关配置说明：**

* <at t='5s'> <disconnect src-module='S1' dest-module='R1'/></at>

在仿真时间5s时，断开模块S1和R1之间的连接。

## BGP协议实验

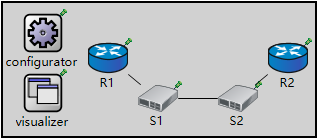
本章介绍BGP实验的相关配置。

本章涉及的配置：

* bgp模块参数配置
* bgp计时器参数配置
* bgp自治系统和内部外部地址配置

### BGP的基本分析

参考教材240页的组网图。拓扑为Experiment.8.ned中的E8\_1，配置为Experiment8.ini中的[Config E8\_1]。拓扑如图所示。



**图8.1 实验8.1网络拓扑图**

实验配置了R1和R2的lo0接口而非lo1接口。PingApp的配置中需要指定源地址为5.5.5.5（R1的RouterID），如指定R1会使用eth0接口的地址，导致Ping数据包无法到达R2。配置bgp的Network时还需要注意配置有关地址的路由，此处在configurator中添加了R1和R2的loopback0接口的路由。实验运行时可以观察bgp模块中的bgp路由表和会话信息。

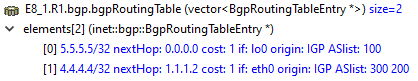
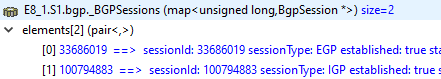


图8.2 R1的BGP路由表项



**图8.3 S1的BGP会话信息**

有关配置说明：

* <route hosts='R1' destination='5.5.5.5' netmask='255.255.255.255' interface='lo0'/>

lo接口路由配置

* <AS id="100">

<Router interAddr="5.5.5.5">

<Network address="5.5.5.5"/>

</Router>

</AS>

自治系统声明、Router声明和BGP网络地址声明

* <Neighbor address="3.1.1.1" nextHopSelf='true' />

对等体声明

* <Session id="1">

<Router exterAddr="1.1.1.1"/>

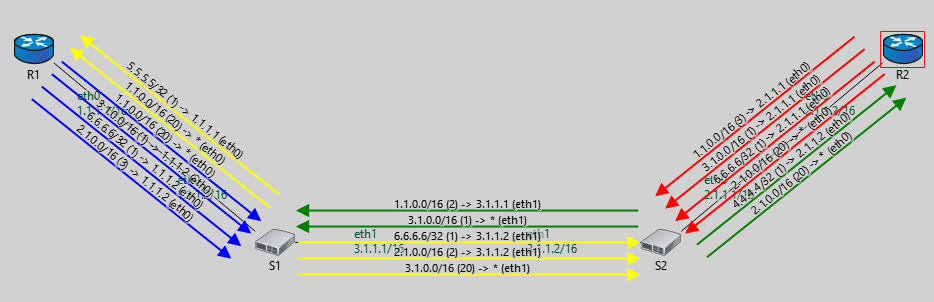
<Router exterAddr="1.1.1.2"/>

</Session>

bgp会话配置

### BGP的路由策略（无MED）

参考教材P246页。拓扑与上一节相同，配置为Experiment8.ini中的[Config E8\_2]。在上一节配置的基础上，添加了S2的lo0接口的路由和BGP网络地址配置，以及S1、S2之间的BGP网络地址配置，关闭了Router抓取数据包的配置。



**图8.4 配置了路由策略后的实验截图**

有关配置说明：

* <DenyRouteOUT Address="5.5.5.5" Netmask="255.255.255.255"/>

在自治系统300中配置不向外传递目的地址为5.5.5.5的路由信息

* <DenyASOUT> 200 </DenyASOUT>

不向外传递自治系统200的路由信息

### BGP的同步机制（需调试）

Bug: No configuration interface for peer address

参考教材P253页的组网图。拓扑与上一节相同，配置为Experiment8.ini中的[Config E8\_3]。本节实验中关闭了S2的Bgp协议配置，添加了S1、S2和R2上的OSPF协议相关配置。

**有关配置说明：**

* \*.S2.hasBgp = **false**

### 设计型实验1

参考教材P254页的组网图。拓扑为Experiment8.ned中的E8\_D1，配置为Experiment8.ini中的[Config E8\_D1]。

### 设计型实验2

### 设计型实验3

## 网络管理实验（未完成）

## 组播实验

## 参考文献

1. 计算机网络实验教程
2. OMNeT++
3. INET Framework
4. Wireshark