浙江科技学院学报,第 23 卷第 4 期,2011 年 8 月
Journal of Zhejiang University of Science and Technology
Vol. 23 No. 4, Aug. 2011

DOI: 10.3969/j.issn.1671-8798.2011.04.006

网络仿真平台 OMNetPP 在网络类课程教学中的应用

王　华
(浙江科技学院 自动化与电气工程学院,杭州 310023)

摘　要: 网络类课程的教学往往显得空洞抽象,学生很难全面理解网络协议和网络通信原理。如果借助 OMNetPP 网络仿真平台和 VC2005 编程工具,则可以将抽象的理论和网络协议等具体化,将复杂的通信过程简单化,而且 OMNetPP 仿真平台还能以动画的形式来表现网络数据包的流向,并统计出相关的参数结果,让学生能够通过图形化的仿真平台快速地理解计算机网络通信的原理,掌握网络通信协议。

关键词: OMNetPP;网络仿真;图形化;仿真平台
中图分类号: TP319;G642.0　　**文献标志码**: A　　**文章编号**: 1671-8798(2011)04-0282-07

Application of network simulation platform OMNetPP in teaching of networking courses

WANG Hua
(School of Automation and Electrical Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China)

Abstract: Network courses appear empty and abstract, it is difficult for the students to understand the network protocol and network communication principle comprehensively. Network simulation platform OMNetPP together with programming tool VC2005 can make the abstract theories materialization and simplify the process of network communication. Also, the simulation kernel of OMNetPP can show the flow of network packets with the form of animation and give the statistical result of key parameters. With the help of the Graphic simulation platform OMNetPP, students can fully understand the theory of computer network communication and grasp the network communication protocol rapidly.

Key words: OMNetPP; network simulation; graphic; simulation platform

收稿日期: 2010-12-25
基金项目: 浙江科技学院教学研究项目(2009Ⅱ B－a02)
作者简介: 王　华(1978—　),男,浙江省淳安人,讲师,博士研究生,主要从事大型网络智能监控系统的研究。

网络通讯类课程是许多工科专业重要必修课程,也是一门非常实用,能很好适应社会需求的课程。但是,在该类课程教学过程中往往存在理论较多、较空洞抽象的问题,学生看不到具体的网络运行方式,而仅仅停留在概念层面的认识,而且网络类课程实验硬件平台价格不菲,不同的实验,需要不同的硬件支撑。采用网络仿真软件可以很方便快捷地建立网络拓扑、仿真经典网络模型,并且可以查看网络拓扑中每个网络模块的动态变化的参数结果,同时还可以生成实时仿真数据图。OMNetPP 就是这样一个优秀的开源的基于模块化的网络仿真平台,作为离散事件仿真器,OMNetPP 具备强大且完善的图形界面接口和嵌入式仿真内核;OMNetPP 可运行于多个操作系统平台,并且可简便定义网络拓扑结构,具备编程,调试和跟踪支持等功能[1]。OMNetPP 主要用于通信网络和分布式系统的仿真,同时还适合离散事件系统的模拟[2]。

1　OMNetPP 体系结构

OMNetPP 主要由仿真内核库(simulation kernel library)、网络描述语言的编译器(network description compiler)、图形化的网络编辑器(graphical network description editor, GNED)、仿真程序的图形化用户接口环境 TKEnv 和仿真程序的命令行用户接口环境 CmdEnv 等组成。另外,对于数据的输出,平台还提供了图形化的矢量输出工具 Plove 和标量输出工具 Scalar[3]。OMNetPP 仿真平台同时还提供了可供其他高级编程语言如 VC2005 等编程接口,通过 VC2005 程序调用 OMNetPP 仿真平台的 Sim 内核和类库,可以实现自定义的网络模型的仿真模拟。OMNetPP 的结构体系如图 1 所示。

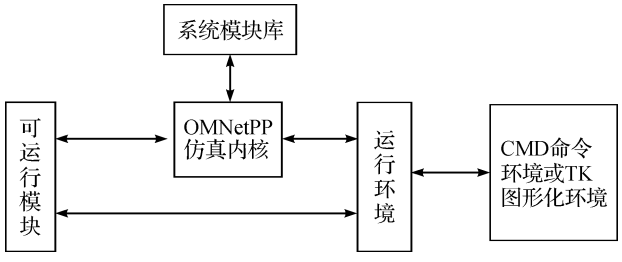

图 1　OMNetPP 体系结构图

Fig. 1　Structure of OMNetPP

一个网络模型在 OMNetPP 仿真平台中通常是由分层的嵌套的模块组成的,顶层模块为系统模块,系统模块由组合模块组成,组合模块由简单模块组成[3],简单模块还可以由它们自身的子模块组成,如图 2 所示。OMNetPP 模块嵌套的深度可以是无穷的,模块与模块之前是通过消息来进行通信的,OMNetPP 的简单模块是整个网络模型的活动模块,采用 C++语言简定义功能。

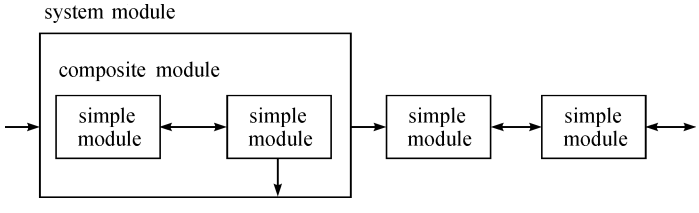

图 2　简单模块和组合模块关系

Fig. 2　Relationship between simple module and composite module

在 OMNetPP 仿真平台中,图形化的网络编辑器 gned. exe 可以将所有网络模块图形化地实现,包括每个模块之间的连接关系,相应的文件扩展名为 ned 的文件只需要将编辑好的 ned 文件连同每个模块的 C++文件及 OMNetPP. ini 配置文件放置在一个目录下面,然后启动 VC2005 工程就可以进行联合编译。

2　客户端与服务端网络通信的仿真实例

通信流程:多个客户端通过交换机与服务器相连,每个客户端发送的网络数据包首先经过交换机,然后被转发到服务器。服务器接收到数据包后向交换机发送响应数据包,交换机根据数据包中的目的地址,

转发给相应的客户端。该客户端接收到交换机转发来的服务端数据包后延迟一定时间后继续向服务端发送网络数据包，一直循环下去。

网络描述(ned)文件设计如下：整个网络模块 CSNetwork 由客户端简单模块、交换机简单模块、服务器简单模块组成。3 个简单模块之间的通信是用消息(网络数据包)来实现的，3 个简单模块之间存在一系列的连接关系，这里每个客户端模块都和交换机模块进行双向连接，交换机模块和服务器模块进行双向连接，如图 3 所示。

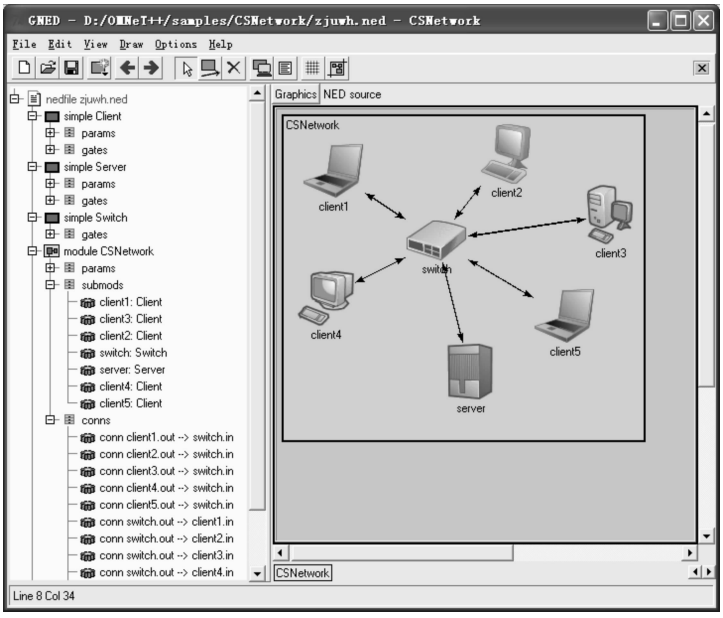

图 3 客户端与服务端网络通信拓扑结构图

Fig. 3 Topology of communication between client and server

图 3 中，左侧属性列表中列举了客户端简单模块、服务端简单模块、交换机简单模块和 CSNetwork 组合模块。客户端简单模块声明了字符串类型的客户端 IP 地址参数 IPAddress，以及消息输入输出门(就是指网络连接)；交换机简单模块仅仅声明了所有的进出交换机的连接，也就是 in[]和 out[]门矢量；服务端模块声明了 IP 地址和接收的总帧数 2 个参数，同时也声明了和交换机连接的数据包进出的 2 个连接。整个 CSNetwork 通信网组合模块的描述如下：

```
module CSNetwork
    parameters:
        ClientNum: numeric const;
    submodules:
        client1: Client;
            display: "i=device/laptop_l;p=85,77";
        client3: Client;
            display: "p=448,172;i=device/pc3_l";
        client2: Client;
            display: "p=348,52;i=device/pc_l";
        switch: Switch;
            display: "p=228,156;i=device/switch_l;o=#0000ff";
        server: Server;
            display: "p=233,314;i=device/server_l";
```

```
            client4: Client;
                display: "p=48,276;i=device/pc4_l";
            client5: Client;
                display: "p=432,304;i=device/laptop_l";
        connections:
            client1.out-->switch.in++;
            client2.out --> switch.in++;
            client3.out --> switch.in++;
            client4.out --> switch.in++;
            client5.out --> switch.in++;
            switch.out++ --> client1.in;
            switch.out++ --> client2.in;
            switch.out++ --> client3.in;
            switch.out++ --> client4.in;
            switch.out++ --> client5.in;
            switch.out++ --> server.in;
            server.out --> switch.in++;
        display: "b=471,397";
    endmodule
    network CSNetwork : CSNetwork
    endnetwork
```

该组合模块中定义了客户端的个数、每个子模块的显示位置、显示的图标参数、每个子模块之间存在的连接关系等。

图 3 右侧是客户端与服务端网络通信的拓扑结构图,程序运行后 OMNetPP 内核将会以动画的形式显示拓扑结构的动态数据流动。每个子模块都是用 C++语言实现具体的功能的,包括数据的初始化,对接收的网络数据包进行处理等。考虑到传递消息的功能特殊性,包括获取消息源、设定目的地址等。本例中特别定制了消息的特殊部分,也就是对 cMessage 类进行了重载,生成 CComMsg 类。步骤如下:首先用文本编辑器编辑一个扩展为 msg 的消息文件 ComMsg.msg,该消息的具体数据成员定义如下:

```
message CComMsg
{
    fields:
        string source;//源地址
        string destination;//目的地址
        int PackSize;//包大小
}
```

新的消息格式中定义了数据源、目的地址及数据包大小,然后用 OMNetPP 安装目录下 Bin 子目录中的 opp_msgc 工具将 ComMsg.msg 自定义消息文件转化为 ComMsg_m.cc 这样一个 C++文件,以后客户端模块、交换机模块、服务端模块相互通信的消息都从 CComMsg 类继承。

客户端简单模块的 C++文件中,必须调用 OMNetPP 核心注册宏 Define_Module()注册用户定义的客户端 C++类 CClient,只有注册后才可被 OMNetPP 内核调用[4]。在客户端类(CClient)中,有默认的 initialize()和 handleMessage(cMessage * msg)两个方法,initialize()主要是做一些数据的初始化,比如本例中客户端 IP 地址的读取,服务器端地址的读取,以及客户端等,定义如下:

```
void CClient::initialize()
{
    std::string strClient=name();//获取当前客户端的名称
    CComMsg * pMsg = new CComMsg ("Client create msg ");//客户端创建消息
    pSelfMsg = new cMessage("self message");//创建发送给自己的消息用于延时
    pkLenBits = &par("pkLenBits");//读取当前客户端的包长
    txRate = par("txRate");//读取当前客户端数据包发送速率
    cPar * ParObj=&par("IPAddress");//获取 IP 地址参数对象
    strIPAdd=ParObj->stringValue();//读取当前客户端的 IP 地址
    pMsg ->setSource(strIPAdd.c_str());//设置要发送消息的数据源地址
    cModule * server = simulation.moduleByPath("server");//获取 server 模块指针
    ParObj=&(server->par("IPAddress"));//获取服务端的 IP 地址
    pMsg ->setDestination(ParObj->stringValue());//设置要发送消息的目的地址
    send(pMsg, "out");//向输出连接发送消息
}
```

在 handleMessage 事件中主要接收各种消息包括自定义消息、交换机转发过来的消息等，根据不同的要求对不同的消息进行处理。本例中对于客户端自身发送的消息主要是用于延时，当接收到自身消息以后，此时可以对上次接收的交换机转发的网络数据包进行处理。如果接收的消息是交换机转发过来的消息，则此时客户端并不立即发送响应数据包给服务器，而是先保存此消息，然后给自己发送一个延时数据包，延时的时间由客户端的发送速率和数据包长度决定。核心处理代码如下：

```
void Client::handleMessage(cMessage * msg)
{
    if (! msg->isSelfMessage())//如果不是自定义消息
    {
        pMyMsg = check_and_cast<CHost *>(msg);//将交换机转发的消息保存下来
        double DelayTime=pkLenBits->doubleValue()/txRate;//计算延时时间
        scheduleAt(simTime()+DelayTime ,pSelfMsg);//给自己发送消息，以作延时用
        return;
    }
    else//自定义消息，则继续向服务器发送消息
    {
        if (pMyMsg)//如果已经接收到服务器发送来的消息
        {
            char Tip[100]="";
            sprintf_s(Tip,"%s receive self-message and send msg to server", name());
            ev << Tip << endl;//离散事件界面显示提示信息
            std::string strDestIP=pMyMsg->getDestination();//获取消息的目的地址
            pMyMsg->setDestination(pMyMsg->getSource());//设置发送消息的目的地址
            pMyMsg->setSource(strDestIP.c_str());//设置消息的数据源地址
            send(pMyMsg, "out");//向服务器发送消息
        }
    }
}
```

与客户端简单模块的 C++实现一样,服务端简单模块的 C++实现也要首先调用 OMNetPP 核心注册宏 Define_Module(),注册服务端模块后,在服务端模块的 initialize()方法中初始化模块的参数,同时在 handleMessage 事件响应方法中处理来自各客户端通过交换机转发过来的消息(数据包),核心代码如下:

```
void Server::handleMessage(cMessage *msg)
{
    CHost *package = check_and_cast<CHost *>(msg);//强制类型转换
    std::string strDestIP = package->getDestination();//提取网络数据包的目的地址
    cPar *ParObj=&par("IPAddress");//获取 IP 地址参数信息
    std::string strServIP=ParObj->stringValue();//将 IP 地址对象转为字符串类型
    std::string strSrcIP=package->getSource();//数据包的来源(IP 地址)
    if (strDestIP==strServIP)//如果是地址匹配
    {
        bubble("send msg to client");//提示发送消息
        package->setSource(strServIP.c_str());//设置消息(数据包)来源
        package->setDestination(strSrcIP.c_str());//设置消息(数据包)地址
        send(package, "out");//发送消息
        }
}
```

交换机简单模块 C++代码的结构和客户端、服务端代码结构类似,在其 initialize 方法中主要是获取所有客户端的 IP 地址信息和服务端的 IP 地址信息,在其 handleMessage 方法中主要是将服务端发送来的消息根据目的地址转发给相应的客户端,将客户端发送来的消息转发给服务端,转发不做任何延时。

OMNetPP 程序启动后,首先加载程序相同目录下的所有 NED 文件,这些文件在 OMNetPP.ini 文件中被指定加载[5],然后内核加载所有相关的模块对应的图片信息,还有相应配置文件中的参数的初始化值[6]。选择相应的网络模型名称,并且运行后生成图 4 和图 5。

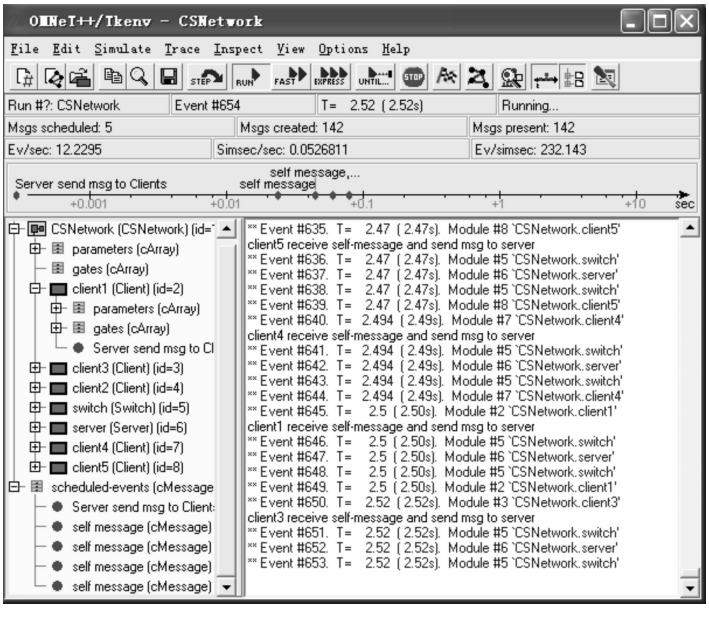

图 4　TKEnv 模拟环境离散事件状态显示

Fig. 4　Discrete event of TKEnv

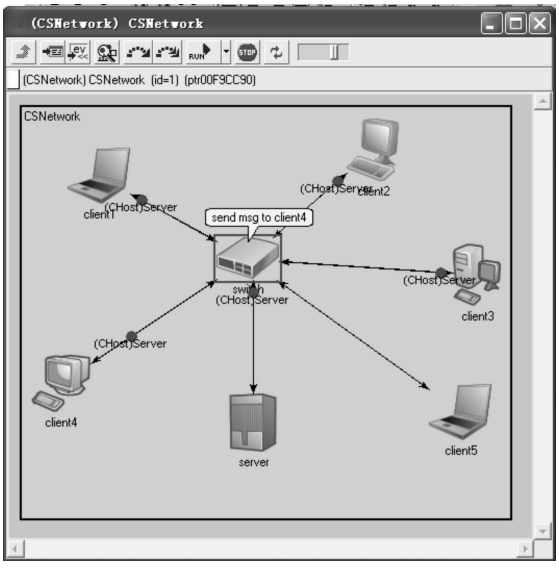

图 5 网络数据包交换动态效果图

Fig. 5 Dynamic exchange of network packet

图 4 中，左侧树形列表中呈现的是网络组合模块中所有的子模块信息，包括每个子模块当前触发的事件信息、参数信息等；树列表的下部是当前组合模块中所有事件类型，包括客户端发送数据包给服务端的事件和服务端发送数据包给客户端的事件；右侧下部区域蓝色字体表示仿真开始后某一个时间点发生的事件，本例中很多事件是并行发生的。在图 4 中，可以很清楚地看到子模块发送的自消息数目（发送给自身的数据包），发送给其他子模块消息数目，整个网络的事件总数，每个事件的触发时间，仿真的速度等信息。而图 5 则是网络数据包的动态交换效果图，该图实时地现实网络数据包的流向。

3 结 语

通过基于 OMNetPP 仿真平台，可以对数据通讯与计算机网络课程中的各种网络模型，诸如 ALOHA 协议、路由协议、令牌环及 Socket 通信等，进行仿真，用 C++语言实现网络模型中的每个模块，然后用 VC2005 连接 OMNetPP 内核编译后就可实现网络模型中数据包流动的动态效果。这些网络仿真对于数据通讯与计算机网络等课程的教学起到重要的作用，学生可以很直观地看到网络模型的运行效果，同时学生自己还可以扩展每个模块，让网络模块按照自己的思路处理网络数据包。OMNetPP 仿真平台不仅可以用于有线网络的仿真，而且还可以用于诸如 3G 通信、无线传感器网络等无线网络的模拟仿真。

参考文献：

[1] OMNetPP 工作组. OMNetPP 用户帮助手册[EB/OL]. [2010-11-20]. http://www.omnetpp.org/doc/omnetpp41/manual/usman.html.

[2] OMNetPP 工作组. OMNetPP 中文使用手册[EB/OL]. [2010-11-20]. http://wenku.baidu.com/view/3f3f0d3b87c24028915fc3b9.html.

[3] VARGA A. OMNeT++ Discrete Event Simulation System User Manual (Version 3.2)[EB/OL]. [2011-11-20]. http://www.omnetpp.org/doc/manual/usmen.html.

[4] KÖPKE A, SWIGULSKI M, WESSEL K, et al. Simulating wireless and mobile networks in OMNeT++ the MiXiM vision[C]//Proceedings of the 1st international conference on simulation tools and techniques for communications, networks and systems & workshops. Marseille:[s.n.],2008.

[5] 操敏，李文锋，袁兵. 基于 OMNeT++的 Leach 协议的仿真研究[J]. 交通与计算机，2007(1)：125-128.

[6] 钟幼平，黄佩伟，汪波. 基于 OMNeT++平台的 SMAC 协议仿真实现[J]. 信息技术，2008(2)：29-33.