浙江工业大学

硕士学位论文

基于 Scilab 和 ns-3 的网络化控制系统联合仿真平台设计

作者姓名	卢帅领
指导教师	赵云波　教　授
学科专业	控制工程
学位类型	工程硕士
培养类别	全日制专业学位硕士
所在学院	信息工程学院

提交日期：2022 年 01 月

Design of a Cooperative Simulation Platform for Networked Control Systems Based on Scilab and ns-3

Dissertation Submitted to

Zhejiang University of Technology

in partial fulfillment of the requirement

for the degree of

Master of Engineering

by

Shuai-ling LU

Dissertation Supervisor: Prof. Yun-bo ZHAO

Jan., 2022

浙江工业大学学位论文原创性声明

本人郑重声明：所提交的学位论文是本人在导师的指导下，独立进行研究工作所取得的研究成果。除文中已经加以标注引用的内容外，本论文不包含其他个人或集体已经发表或撰写过的研究成果，也不含为获得浙江工业大学或其它教育机构的学位证书而使用过的材料。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人承担本声明的法律责任。

作者签名：卢帅领　　日期：2021 年 12 月

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权浙江工业大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于：　1、保密□，在一年解密后适用本授权书。

2、保密□，在二年解密后适用本授权书。

3、保密□，在三年解密后适用本授权书。

4、不保密☑。

（请在以上相应方框内打“√”）

作者签名：卢帅领　　日期:2021年 12 月

导师签名：赵云波　　日期:2021年 12 月

中图分类号　TP391.9　　学校代码　10337

UDC　681.5　　密级　公开

研究生类别　全日制专业型硕士研究生

浙江工业大学

工程硕士学位论文

基于 Scilab 和 ns-3 的网络化控制系统联合仿真平台设计

Design of a Cooperative Simulation Platform for Networked Control Systems Based on Scilab and ns-3

作者姓名　卢帅领　　第一导师　赵云波　教　授

学位类型　工程硕士

学科专业　控制工程　　培养单位　信息工程学院

研究方向　网络化控制

答辩日期：　2021　　年　12　月　08　日

基于 Scilab 和 ns-3 的网络化控制系统联合仿真平台设计

摘　要

基于控制与通信融合的联合仿真是网络化控制系统一种主要且具有独特优势的仿真实验方式，在这种仿真方式中，由擅长于控制系统仿真的数学计算软件仿真控制系统模块，由网络仿真软件仿真网络通信模块，之后将两个模块的仿真融合，形成完整的网络化控制系统仿真。这样既实现对网络化控制系统分模块精确仿真，又不破坏仿真实验的整体性，在网络化控制系统仿真实验中具有巨大的优势。

然而，现有的网络化控制系统联合仿真平台大多基于 MATLAB 开发，这种基于 MATLAB 搭建的仿真平台往往有难以深度定制功能、不能二次发布、高成本、高版权风险和难以推广应用等缺点，也不符合国内开源生态发展的趋势。因此，设计并开发一种开源免费的网络化控制系统联合仿真平台具有相当重要的现实意义。

本文对网络化控制系统的仿真实验做了深入研究，分析了近年来网络化控制系统仿真工具的发展，提出了一种基于开源免费软件的网络化控制系统联合仿真平台，选取用于控制系统仿真的开源数学计算软件 Scilab 和开源网络仿真软件 ns-3 作为搭建网络化控制系统联合仿真平台的软件。本文的主要工作如下：

(1) 设计了基于 Scilab 和 ns-3 的联合仿真平台总体架构和用于 Scilab 和 ns-3 软件间通信的联合仿真接口程序。联合仿真平台架构中确定了联合仿真平台的各模块功能以及模块之间的通信方式，还规定了各功能模块运行的次序。联合仿真接口程序连接 Scilab 和 ns-3 的程序接口，统一两个软件数据交换的格式，实现 Scilab 和 ns-3 仿真结果的交换与解析。

(2) 设计了用于同步 Scilab 和 ns-3 仿真时间的主从式周期同步机制。根据 Scilab 和 ns-3 两种软件驱动方式的不同以及联合仿真对时间同步的需求，设计了一种主从式周期同步机制用于协调 Scilab 和 ns-3 联合仿真的次序并同步两软件的仿真时间，能够保证联合仿真平台的正常运行。

(3) 通过典型实例验证了联合仿真平台的可用性。一方面，通过仿真有损多包传输的无线网络控制系统验证了联合仿真平台对多包传输的无线网络控制系统的有效性，另一方面，通过与其他仿真方式的仿真实验结果对比，验证了联合仿真平台的可靠性。

最后，对本文工作进行了总结与分析，指出了联合仿真平台在设计方面的优

点和不足，并展望了未来的研究方向。

关键词：网络化控制系统，网络通信，Scilab 和 ns-3，联合仿真平台，开源软件

DESIGN OF A COOPERATIVE SIMULATION PLATFORM FOR NETWORKED CONTROL SYSTEMS BASED ON SCILAB AND NS-3

ABSTRACT

The co-simulation based on the fusion of control and communication is a major and uniquely advantageous simulation experiment method for networked control systems. In this simulation method, control system modules are simulated by mathematical calculation software that is good at control system simulation, and network communication modules are simulated by network simulation software, and then merges the simulations of the two modules to form a complete networked control systems simulation. This method not only realizes the accurate simulation of the sub-modules in networked control systems, but also ensure the integrity of the simulation experiment, so the co-simulation method has a huge advantage in the simulation experiment of the networked control systems.

However, most of the existing networked control systems co-simulation platforms are developed based on MATLAB. Such simulation platforms often have some shortcomings, such as difficulty in deep customization of functions, inability to re-release, high cost, high copyright risk, difficulty in popularizing applications and out of the development trend of open-source software. Therefore, the design and development of an open-source and free networked control systems co-simulation platform has very important practical significance.

This paper has done an in-depth study on the simulation of networked control systems, analyzed the development of networked control systems simulation tools in recent years, and proposed a networked control systems co-simulation platform based on open source and free softwares. The open-source mathematical calculation software Scilab and the open-source network simulation software ns-3 are selected as the software to build a networked control systems co-simulation platform. The main work of this paper is as follows:

(1) Designed the overall architecture of the co-simulation platform based on Scilab and ns-3, and the co-simulation interface program for communication between Scilab and ns-3 software is designed. The architecture determines the functions of each module of the co-simulation platform and the communication mode between the

modules, and the architecture of the co-simulation platform also specifies the order in which each functional module runs. The co-simulation interface program connects the APIs of Scilab and ns-3, can unify the format of data exchange between the two software, and realize the exchange and analysis of the simulation results of Scilab and ns-3.

(2) A master-slave synchronization mechanism for synchronizing the simulation time of Scilab and ns-3 is designed. According to the different driving methods of the two software and the requirements of time synchronization for co-simulation, a master-slave synchronization mechanism is designed to coordinate the sequence of Scilab and ns-3 and synchronize the simulation time of the two software, which can ensure the normal operation of the co-simulation platform.

(3) The usability of the co-simulation platform is verified through typical examples. On the one hand, by simulating the wireless network control system of lossy multi-packet transmission, the effectiveness of the control system of the co-simulation platform for multi-packet transmission is verified. On the other hand, the reliability of the co-simulation platform is verified by comparing the results of simulation experiments with other simulation methods.

Finally, the work of this paper is summarized and analyzed, the advantages and disadvantages of the design of the co-simulation platform are pointed out, and the future research direction is prospected.

KEY WORDS: Networked control systems, Network communication, Scilab and ns3, Cooperative simulation platform, Open-source software

目 录

第一章 绪 论

1.1 研究背景及意义

随着现代通信技术和现场总线技术的发展和成熟，远程控制系统和无人操控设备的应用越来越广泛，在传统控制系统中加入通信网络形成网络化控制系统(Network Control Systems , NCS)成为一种趋势。通过通信网络将现场智能设备网络化和分布化，既增加了设备在同一场景的协同性，也加强了现场设备与上层控制系统的联系。网络化控制系统是一种通过传输网络将控制系统中各节点连接起来的闭环控制系统[1]。网络化控制系统通过通信网络来实现各节点间的数据传输，与传统的控制系统相比，网络化控制系统具有应用成本低、布线简便、方便维护、结构灵活等优点[2]。网络化控制系统广泛应用在智能家居[3]、无人车、无人机、分布式系统[4]、工厂自动化[5]、智能电网等领域中（图 1-1）。

图1-1 网络化系统应用场景（a）智能家居（b）无人车（c）无人机

Figure 1-1. Application scenarios of Network control system（a）Smart home（b）Driveless car（c）Unmanned plane

随着网络化控制系统的广泛应用，对网络化控制系统的研究变得越来越深入，从而促进了对网络化控制系统的仿真需求。网络化控制系统仿真中需要建立的数学模型变得越来越复杂，但网络化控制系统的数学模型离不开最基本的经典模型。网络化控制系统经典模型框图如图 1-2 所示，网络化控制系统可以抽象的看作由四个节点和把它们连接到一起的通信网络组成，这四个节点按其功能可划分为控制器节点、传感器节点、执行器节点和被控对象节点[6]。

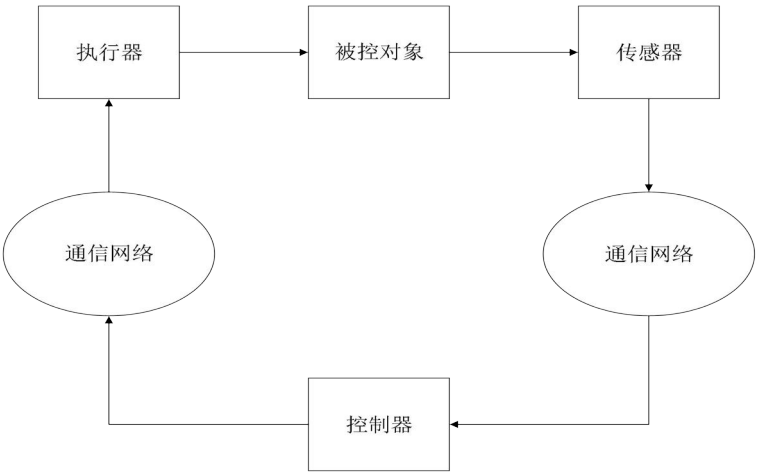


图1-2　网络化系统框图

Figure 1-2.　Network control system

对网络化控制系统的仿真离不开仿真工具，仿真工具按照仿真形式可分为仿真软件和仿真硬件。仿真软件是能够依托计算机平台模拟物理世界效果的一类软件，它与仿真硬件都是用于仿真实验的工具。仿真软件是从 20 世纪 50 年代中期逐渐发展起来的，仿真软件的发展离不开仿真的应用、仿真算法、计算机和建模技术的发展。相比于仿真硬件，用仿真软件模拟现实中的实验具有低成本、高效率等优点。随着计算机等相关技术的发展，仿真软件也得到了长足进步，其精确度和实验效率也越来越高，仿真软件的使用范围越来越多[7]。

科学研究领域因其需要大量实验，为了提高效率，仿真软件的使用在科研领域应用非常广泛。同时科学研究领域对实验的准确性要求较高，这就需要所使用的仿真软件具有很高精确度。

1.2 网络化控制系统仿真平台研究现状

本节主要介绍了现今常用的控制系统仿真平台形式以及网络化控制系统仿真平台的研究现状，之后分析了在网络化控制系统仿真中具有较大优势的联合仿真平台。

1.2.1 网络化控制系统主流仿真平台

网络化控制系统的仿真实验按照仿真形式可分为全物理仿真、半物理仿真和数值仿真[8]。

（1）全物理仿真平台

全物理仿真是整个实验过程全部采用物理模型的仿真，又称全实物仿真。例如在航天器控制系统中利用气浮台模拟航天器作为被控对象[9]，控制系统采用了航天器的实际控制系统进行仿真实验。全物理仿真不用建立航天器控制系统和被控

对象（航天器本体）的数学模型，而是系统对被控对象（气浮台）直接控制。全物理仿真实验方式能够避免一些在控制系统的设计中难以发现的问题，也避免了航天器控制系统中一些实际部件难以精确建立模型的问题，这些难以精确建立模型的实际部件作为全物理仿真系统的一部分直接参与仿真。全物理系统在一些方面，如航天器太阳能电板的动力学研究实验中有不可替代的作用[10]，但是全物理仿真实现的技术复杂，成本较高，可推广性也较差，一般只在必要时才会采用。

（2）半物理仿真平台

半物理仿真也称半实物仿真，是一种采用部分物理模型和部分数学模型结合的仿真形式。控制系统的半物理仿真主要是分模块进行仿真，一部分模块采用实物，另一部分模块用数学模型替代，之后连接两部分形成完整的半物理仿真。半物理仿真这种仿真形式在飞行控制系统领域广泛应用[11]，如在飞行动力学仿真系统中的飞行控制系统、负载模拟器等单元加入可视化虚拟仿真现实环境，而各系统之间连接方式为实际的光纤和串行总线等。半物理仿真能够实现仿真的实时性、通用性和可靠性要求，对难以建立数学模型的控制系统复杂部件部分直接让实际部件参与实验，其他部分使用数值仿真。如在列车控制系统研究中搭建的半物理仿真平台中使用 ARCNET 光纤双环王组网通信[12]，将列车控制系统分为几个模块，中央装置内部逻辑计算和司机操作台等终端装置功能进行数值仿真，系统的模块之间连接方式为实际的光纤和串行总线。在实验中完成了设备端装置功能仿真和显示器功能仿真的数据管理等功能，并设计实现了各功能模块仿真和 ARCNET 光纤双环网间通信接口，最终完成了整个列车控制系统的半物理仿真。半物理仿真将整个控制系统分成几个模块，根据各模块特性分别选用数值仿真和实物仿真。半物理仿真相比于全物理仿真可显著降低平台搭建的困难和成本[13]，但是半物理仿真存在实物与数学模型间通信不便、模块仿真实时性差等缺点，难以广泛推广和应用。

（3）数值仿真平台

数值仿真也称软件仿真，数值仿真是一种完全依赖计算机软件，整个仿真实验过程都在计算机上完成的仿真方式。控制系统的数值仿真是在仿真软件上建立描述控制系统动态过程的数学模型，并在这个数学模型上对控制系统进行定量的研究和实验。这种仿真方法常用于控制系统的方案设计阶段和一些难以做实物仿真的场合以及低成本的仿真方案中。它的特点是重复性较好、可灵活配置、精度高、使用比较方便、成本低、可以满足实时性或者非实时的要求。数值仿真的逼真程度和精确度取决于仿真计算机软件的精度和数学模型的正确性与精确性，因此在采用数值仿真的实验中选择优秀的软件仿真平台非常重要。数值仿真有多种实现方式，如使用模拟计算机、数字计算机和数字-模拟混合计算机等方式。

在网络化控制系统的研究过程中，对通信网络的变量精准控制非常重要，全物理仿真和半物理仿真中如果加入实际的通信网络，因为网络延时和丢包的不确定性，可能会导致在对比试验中不能较好的控制变量。数值仿真作为一种便利、低成本、易推广和能较好控制变量的仿真形式，正在成为网络化控制系统仿真的主流仿真形式。在网络化控制系统的研究中大多依托于软件仿真平台[14]。

1.2.2 联合仿真平台

网络化控制系统的研究涉及到控制系统和通信网络的研究，当通信网络对控制系统的动态特性产生重大影响时，必须在研究中加入精准的网络特性描述。网络性能主要受到两方面的影响：数据包的丢失和延时。在通信网络领域的研究中，许多研究者通过建立数学模型来预测通信的延时[15]，通过随机过程模拟数据包的丢失[16]，在控制系统的研究中也有学者提出了一些容忍或补偿网络不确定性对控制系统的影响[17,18]。

网络化控制系统可分为控制系统部分和通信网络部分，要完成网络化控制系统的仿真，需要仿真平台具有同时仿真控制系统和通信网络的能力。在网络化控制系统的仿真研究领域，能够同时进行控制系统动态仿真和通信网络仿真的联合仿真方法主要有两种：第一种是对单一软件的扩展，使得原本只能仿真特定领域（控制系统领域或通信网络）的软件能同时仿真控制系统和通信网络，包括对控制系统仿真软件的扩展和对网络仿真软件的扩展；第二种是设计一种工具程序能够联合不同的软件进行联合仿真[19]。具体的仿真方式可分为以下三种：

（1）扩展数学计算软件

数学计算软件是一类专用于数值计算的计算机软件。数学计算软件能够对控制系统进行数值仿真。对数学计算软件进行扩展，使得数学计算软件不仅能够仿真控制系统也能拥有仿真通信网络的能力。扩展数学计算软件，必须选择一种能够在软件中搭建网络模块的数学计算软件，用网络模块代替传统控制系统仿真的有线连接。如在数学计算软件 MATLAB 内的 Truetime 工具箱、NCS_Simu、NCSLab 等，就是对 MATLAB 的功能扩展，使得 MATLAB 具备了较好的网络仿真能力[20]。

TrueTime 是一个知名的仿真工具箱[21]，是由瑞典的隆德大学（Lund University）开发，TrueTime 支持 CSMA/CD、TDMA、CSMA/AMP 等 MAC 层协议，无线网络协议 802.11b/g（WLAN）和 IEEE 802.15.4（ZigBee），被广泛应用在网络化控制系统的仿真中[22]。TrueTime 的网络仿真功能是由 Simulink 中的 sfuntion 模块实现，然而 TrueTime 只支持本地区域网仿真，只能仿真物理层和 MAC 层，使用限制较多。这限制了它在一般网络和上级网络协议的应用[23]。

NCS_Simu 是由英国 Sussex 大学杨泰澄教授主持开发的网络控制仿真包，演示案例选择经典的倒立摆弹簧系统作为被控对象，进行了网络化控制系统的仿真

实验[24]，但是 NCS_Simu 仿真包对 WAN 仿真的支持比较差，在网络仿真层面上仅支持物理层和 MAC 层，与 TrueTime 类似，这限制了它在一般网络和上级网络协议的应用。

NCSLab 是由哈尔滨工业大学航天学院刘国平教授主持设计的集成网络浏览器、网络服务器、MATLAB 服务器、被控对象和区块实验服务器的远程仿真虚拟实验平台。NCSLab 可满足通过网络进行远程实验，并且可进行观看 3D 虚拟现实仿真实验界面，该系统集成了倒立摆、电机等多种实验对象，可满足大部分的仿真实验要求[25]。尽管 NCSLab 系统设计精致而且方便使用，但是 NCSLab 基于网络服务有着其限制性，如网络拥堵、服务器满载可能导致用户不能及时实验，仿真数据安全挑战以及离线设备的使用限制等。

对数学计算软件进行扩展仿真，虽然实现了使用单独的软件就能仿真包含控制系统和通信网络两个模块的网络化控制系统，但对于主要用作控制系统仿真的数学计算软件来说，精确的仿真通信网络还是显得有些吃力。

（2）扩展网络仿真软件

对网络仿真软件扩展使得网络仿真软件能够仿真控制系统，实现方式主要是在网络仿真软件中建立控制系统模型。网络仿真软件在仿真时调用控制系统模型，使得网络仿真软件能够仿真控制系统。常见的可扩展网络仿真软件有 ns-2、ns-3 和 OPNET 等。在上述网络仿真软件内建立控制系统模型，如在网络仿真软件 ns-3 中建立控制系统的微分方程模型，这样 ns-3 就可以仿真控制系统。但是这种方式的实现难度较大，在仿真大型控制系统时建模的周期将会很长，而且容易出错，网络仿真软件对微分方程的计算速度慢也限制了仿真实验的应用场景。

有许多研究者对网络仿真软件进行扩展实验研究，如西南交通大学的童晓阳教授设计了一种扩展 OPNET 的仿真工具用于扩展 OPNET 网络仿真器，并将其应用在网络化电力领域控制系统的研究中，但是该方法限制在一个控制系统和一个通信系统之间进行数据包的传输，无法进行多控制系统和多包传输的研究[26]。

（3）软件联合仿真平台

网络化控制系统通过通信网络构建成闭环控制回路，单独的控制系统仿真软件或网络仿真软件难以同时对网络化控制系统的控制系统模块和通信网络模块实现较为精确的仿真。于是网络化控制系统的仿真研究者便选择结合两种软件进行仿真。将数学计算软件和网络仿真软件进行联合仿真可以满足对网络化控制系统的精确仿真需求。

将数学计算软件和网络仿真软件结合起来能够避免上述（1）、（2）两种方式的缺点，逐渐成为网络化控制系统仿真的一种趋势。这种仿真方式不仅能够实现网络化控制系统高精度、低成本的仿真实验，还同时能够获取控制系统和通信

网络仿真过程中的信息，可以展示整个网络化控制系统的动态过程，便于分析和改进控制系统性能和网络对整个系统的影响。联合仿真平台的实现方式有以下两种：

（1）基于 HLA（高级体系架构）的联合仿真平台，最早是由美国国防建模与仿真实验室制定的建模与仿真计划中提出的高级建模与仿真架构。在 HLA 架构平台上运行的仿真软件间能够实现相互操作，联合仿真。HLA 为仿真软件提供接口，基于 HLA 搭建的联合仿真平台能够实现将每个独立的仿真软件在一个统一的仿真时间和环境中协调运行和互相操作，但是其学习难度大，编程与实现较为困难。

（2）其他的联合仿真平台如 MATLAB/Simulink 与 OPNET 联合仿真[27-29]、MATLAB 与 ns-3 联合仿真[30]。这些联合仿真平台大都是基于 MATLAB 开发，搭配网络仿真软件组成联合仿真平台，其虽能实现一般的网络化控制系统仿真，但是主导软件 MATLAB 属于商业软件，具有闭源，不能修改内核，难以定制设计功能，价格昂贵、软件的修改和二次发布受到限制等缺点，从而导致基于 MATLAB 的联合仿真受限制较多，灵活性差。也不符合国内开源发展的趋势。

现有的网络化控制系统联合仿真平台，存在可推广性差，仿真成本较高，搭建难度大，使用限制多等缺点。一些仿真平台存在实时性差，仿真速度慢，编程和搭建困难等缺点。随着近年来安全性、可靠性、便捷性和开源成为仿真平台使用者的关注点，需要一款解决现存仿真平台缺点的仿真工具。

1.2.3 联合仿真平台软件选择

针对上述联合仿真平台的痛点，本文旨在选择合适的数学计算软件和网络仿真软件，并结合国内开源生态发展的趋势，设计一种低成本，便以推广和使用的联合仿真平台。

（1）数学计算软件的选择

对控制系统的仿真研究，主要借助于数学计算软件，现如今比较成熟的数学计算软件如 MATLAB、Modelica、MathType、Mathematica、Maple、MathCad、Scilab、SAGE、Microsoft Mathematics、Octave、Spyder、Python、GMAT、GNU Radio、ROS 等，这些软件都能较好的进行数值仿真。有些软件还具有丰富的工具箱，其中包括 MATLAB、Scilab 、Octave 和 Mathematica 等，数学计算软件性能的对比如表 1-1 所示。

表1-1　数学计算软件性能对比

Table 1-1. Math calculate simulator

软件名称	运行平台	是否开源	可视化仿真工具	易用性
MATLAB	跨平台	不开源	有	好
Octave	跨平台	开源	无	中
Scilab	跨平台	开源	有	好
MathType	跨平台	不开源	无	好
Mathematica	跨平台	不开源	无	好
MathCad	跨平台	不开源	无	好
Maple	跨平台	不开源	无	好

数学计算软件在科学研究领域非常受欢迎，它为现代科学研究各领域中出现的数学问题提供求解手段，数学计算软件也是组成许多应用软件的基本构件。随着计算机技术的发展，这类软件功能不断丰富，使用场景不断扩展，除了数学运算，还能够进行数学规划、统计运算、工程运算、绘制数学图形或制作数学动画等。这些优势也推动了数学计算软件在工业界和学术界的使用，以至于现今几乎每个实验室都需要数学计算软件。在数学计算软件中，MATLAB 应用最为广泛，但它却是一个昂贵的软件，而且 MATLAB 对跨平台的支持不够好，这限制了 MATLAB 的推广，在一些地区需要一款完美替代 MATLAB 的数学计算软件。在一系列数学计算软件中 Scilab 和 Octave 是为数不多的开源软件，他们的功能同样强大，更是兼具易用和免费的特点，用户可以根据自己的需求进行修改和二次发布。Octave 的语法完全兼容 MATLAB，但是与 MATLAB（包括可视化工具箱 Simulink）相比，Octave 提供的各种解决方案有限，比如 Octave 不提供 MATLAB 可视化仿真工具 Simulink 的等价物。Scilab 的语法方面设计并不完全按照 MATLAB 的路线，不兼容 MATLAB 的语法，Scilab 被证明是 MATLAB 的一个完整替代方案，因为 Scilab 的 Xcos 提供了 Simulink 的等价物，而且 Scilab 包括 MATLAB 基本工具包和某些专业包中几乎所有功能 [31]。本文选择 Scilab 作为网络化控制系统联合仿真平台中负责仿真控制系统模块的软件。

（2）网络仿真软件的选择

对通信网络的仿真主要借助于主流网络仿真软件，如 ns-2、ns-3、opnet、GlomoSim、OMNeT++、J-SIM 等。各仿真软件性能对比如表 1-2 所示。

表1-2　网络仿真软件性能对比

Table 1-2. Network simulator

软件名称	编程语言	运行平台	是否开源	运行速度①	灵活性②	易用性③	可扩展性④
ns-2	C++、Otcl	跨平台	开源	*	*	*	**
ns-3	C++	跨平台	开源	***	**	***	***
GlomoSim	Parsec	Linux	开源	**	*	**	***
QualNet	Parsec	跨平台	不开源	**	*	***	*
OMNet++	C++	跨平台	开源	**	**	**	***
OPNET	C++	跨平台	不开源	*	**	***	*
J-SIM	Java	跨平台	开源	**	***	**	***

注：①②③④“*”代表差，“**”代表一般，“***”代表较好。

在众多的网络仿真软件中 ns-3、OPNET 因其可跨平台运行、容易使用、运行速度快等优点而脱颖而出。相比于 OPNET，ns-3 更具有开源，可扩展性强的优点，已经成为网络仿真中广泛应用的软件[32]。本文选择 ns-3 作为网络化控制系统联合仿真平台中负责仿真通信网络模块的软件。

（3）操作系统的选择

本文搭建联合仿真平台是基于 Linux 的发行版 Ubuntu20.04 LTS 操作系统。Ubuntu 操作系统是广泛使用的 Linux 发行版，用户使用方面比较友好，上手简单。ns-3 虽然可以跨平台使用，但其主要运行平台是 Linux，对 Linux 的支持比较好，ns-3 虽然可以在 Windows 系统上借助 Cygwin 或 VisualStudio 运行，但是缺少对一些功能的支持（如与物理网络的交互），同时在 Ubuntu 上使用 Scilab 可以借助命令行查看辅助信息输出，方便对 Scilab 运行过程进行监控。本文选择 Linux 发行版系统 Ubuntu 作为搭建联合仿真平台的系统。

1.3 Scilab 和 ns-3 介绍

本节中将主要介绍 Scilab 软件和 ns-3 软件的发展历史以及常用功能。

1.3.1 Scilab 软件介绍

Scilab 是由法国国家信息与自动化研究所（INRIA）推出的一种比较出色的数学计算软件，它是开源且可跨平台运行的软件。软件的历史始于上世纪 80 年代，它的前身 Blaise 是一种计算机辅助控制系统设计（CACSD）软件，由法国计算机科学与控制研究所（IRIA）创建，目的是提供研究人员的自动控制工具。上世纪 90 年代初，Basile 停止发布，软件名称改为 Scilab。Scilab 开始作为免费的开源软件发布。Scilab 1.1 是 Scilab 的第一个发布版本，于 1994 年 1 月 2 日在匿名 ftp 站

点上发布。自 2008 年以来，Scilab 在 CeCILL 许可证下分发，CeCILL 许可证是一种开源 GPL 兼容许可证。Scilab Enterprises 公司于 2010 年 6 月成立，以保证 Scilab 的后续发展。自 2012 年 7 月起，Scilab Enterprises 全面负责 Scilab 的版本和开发。基于经典的开源商业模式，Scilab Enterprises 继续为 Scilab 提供专业的服务和支持[33]。Ubuntu 系统下 Scilab 控制台主界面如图 1-3 所示。

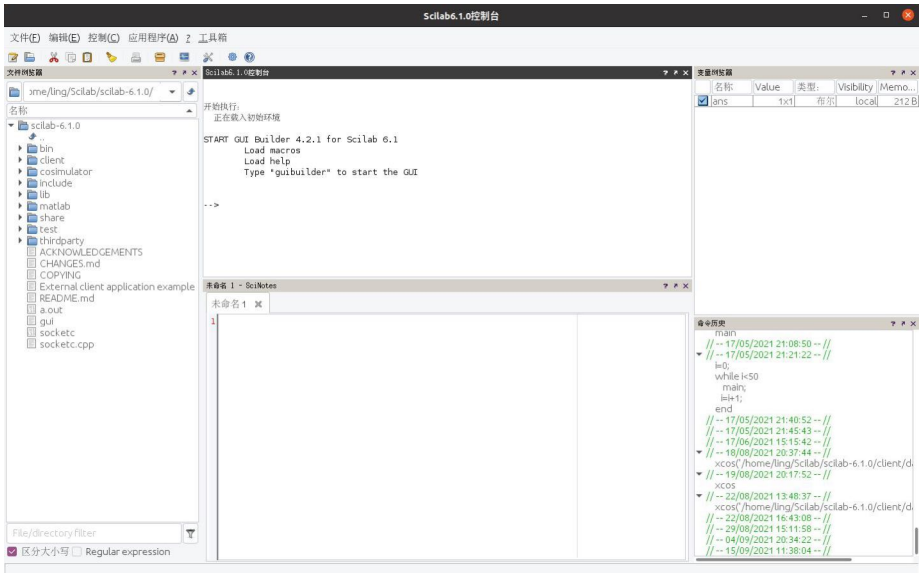


图1-3　Scilab控制台主界面

Figure 1-3.　Scilab console main interface

Scilab 被视为对标 MATLAB 的免费软件，开发团队是由来自高等教育学校或大学的工程师和科学家组成，他们大都是数学、自动化、电信、物理或计算机科学等专业，对 Scilab 软件及其环境有广泛的了解，并参与到 Scilab 的开发。在 2017 年，Scilab 被 ESI 集团收购，但是该集团仍然致力于开发用于数值计算的免费开源 Scilab 软件。凭借 OpenFOAM 的经验，ESI 集团已经证明了其对工程师和科学家开源软件的承诺。经过 50 年的发展，Scilab 功能越来越完善，又因其免费和开源的特性，维护和开发参与者众多，有着比较活跃的开发社区，在科学计算领域发挥着越来越重大的作用[34]。

Scilab 可在 GNU/Linux、Mac OS X 和 Windows XP/Vista/7/8/10 下使用。Scilab 包括数百个数学函数，含有高级编程语言，允许访问高级数据结构、具有 2-D 和 3-D 图形功能。

Scilab 中包含大量功能：

(1) 数值仿真：可用于通常的工程和科学应用，包括数学运算和数据分析。

(2) 2-D 和 3-D 可视化图形绘制：图形功能用于可视化、注释和导出数据，以及多种创建和自定义各种类型图和图表的方法。

(3) 优化：解决约束和无约束连续和离散优化问题的优化算法。

(4) 数据分析：执行数据分析和建模的统计工具。

(5) 控制系统：用于控制系统研究的标准算法和工具。

(6) 信号处理：在时域和频域中可视化、分析和过滤信号。

(7) 应用程序开发：增加 Scilab 本地功能并管理与外部工具的数据交换。

(8) Xcos 可视化仿真工具箱：对动态系统进行数学建模，如机械系统、液压回路、控制系统的建模等。

(9) 作为平台与其他软件协作：由于具有与第三方技术和应用程序互连的能力，Scilab 还可以作为一个独特的平台，将用不同编程语言编写的代码以单一、统一的语言汇集在一起，从而促进它们的分发、备份和使用。Scilab 因其开源可以在修改后重新发放，能实现自主可控，符合国内开源生态发展的趋势。

1.3.2 ns-3 软件介绍

ns-3 是一个用于通信网络系统的仿真软件，主要用于研究和教育领域。ns-3 是由 C++语言编写，专门用于模拟通信网络的开源软件，ns-3 没有图形用户界面，而是被构建为一个 C++程序库，提供各种用于网络模拟的应用程序接口[35]。Ubuntu 系统下 ns-3 终端运行界面如图 1-4 所示。

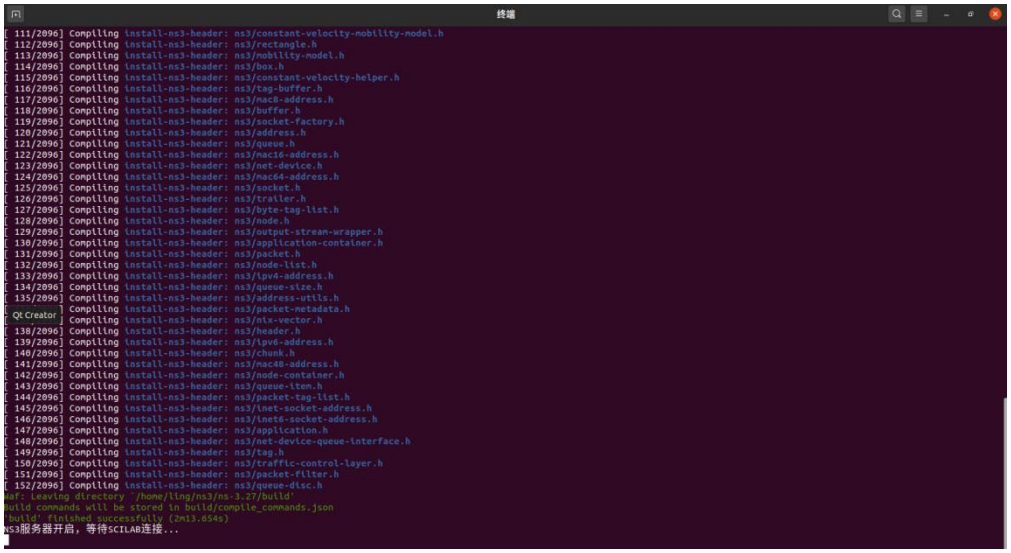


图1-4　ns-3终端运行界面

Figure 1-4.　ns-3 terminal running interface

ns-3 是开源免费的离散事件网络仿真软件，ns-3 并不是 ns-2 的版本更迭[36]，而是一个重新编写的软件[37]。ns-3 在 GNU GPLv2 许可下获得许可，可公开用于研究、开发和使用。ns-3 项目的目标是为网络研究开发一个首选的开放模拟环境，它与现代网络研究的模拟需求保持一致，ns-3 项目鼓励社区贡献、同行评审和软件验证。ns-3 项目致力于构建一个有据可查、易于使用和调试的实体仿真核心，满足从仿真配置到跟踪收集和分析的整个仿真工作流程的需求[38]。此外，ns-3 软

件基础框架鼓励开发足够逼真的仿真模型，允许将 ns-3 用作实时网络仿真器，与现实世界互连，并允许重复使用。许多现有的现实世界协议可以在 ns-3 内实现[39]。

ns-3 仿真核心支持基于 IP 和非 IP 的网络的研究。但是，其绝大多数用户专注于无线/IP 仿真，其中涉及用于第 1 层和第 2 层的 Wi-Fi、WiMAX 或 LTE 模型以及各种静态或动态路由协议，例如用于基于 IP 的应用程序的 OLSR 和 AODV。ns-3 还支持实时调度程序，可促进许多与真实系统交互的“在环仿真”用例[40]。例如，用户可以在真实的网络设备上发送和接收 ns-3 生成的数据包，而 ns-3 可以作为互连框架在虚拟机之间添加链接效果。每三个月，ns-3 社区都会发布一个新的 ns-3 稳定版本，其中包含由研究人员开发、记录、验证和维护的新模型。社区鼓励第三方在邮件列表上公开验证这些模型，以确保运送的模型尽可能保持最高质量。

对于网络协议的模拟，ns-3 使用了离散事件的模拟技术，这种技术把物理世界中一个连续的过程抽象成了虚拟世界中的一系列的离散事件。这种技术使得 ns-3 可以非常逼真地模拟物理时间中各种网络协议，如应用层的各种分组产生器、传输层的 TCP 和 UDP、网络层的 IPv4 和 IPv6 协议、链路层和物理层的 PPP、IEEE802.11a/b/g/n 和 IEEE 协议等[41]，ns-3 支持的协议如表 1-2 所示。

表1-2 ns-3模拟的网络分层及支持协议

Table 1-2. ns-3 simulated network layering and supporting protocol

网络层	支持的网络协议
应用层	分组产生器、应用层协议（ping 等）
传输层	TCP、UDP
网络层	IPv4、IPv6、ICMP、ICMPv6、路由协议
链路层和物理层	LTE、WiFi、车载网、个域网、水下通信网、点对点网络、总线CSMA网络、ARP

1.4 本文研究内容和章节安排

本文根据网络化控制系统的特点并结合近年来网络化控制系统的仿真研究，设计并实现了一种开源免费的网络化控制系统联合仿真平台，本文的框架分为以下几个部分：

第一章绪论中介绍了网络化控制系统和网络化控制系统仿真工具的研究现状，分析了网络化控制系统现今仿真方式的优缺点，并确定了搭建基于 Scilab 与 ns-3 的开源联合仿真平台，之后对 Scilab 与 ns-3 进行了介绍。

第二章介绍了开源联合仿真平台的总体架构，结合现有的进程间通信技术和进程间时间同步方案，分析了联合仿真平台所需要设计的模块和需要使用的技术和工具，提出了一种联合仿真接口设计。

第三章介绍了联合仿真平台的同步机制设计以及现有的仿真软件的时间同步方案，并设计了一种主从式周期同步机制作为本文联合仿真平台的同步机制。

第四章介绍了联合仿真接口的程序实现，详细介绍了主从式周期同步机制在 Scilab 与 ns-3 软件内的程序实现。

第五章选取了实验实例对联合仿真平台进行功能测试，并使用其他仿真工具做了对比试验用于验证联合仿真平台的可用性和可靠性。

第六章对联合仿真平台搭建工作做了总结，并对联合仿真平台的设计和实现进行了分析。对联合仿真平台未来的研究方向做了展望。

附录为 Scilab 与 ns-3 的使用及其运行原理介绍，以及 Xcos 控制系统模型的驱动程序。

第二章 基于 Scilab 和 ns-3 联合仿真平台总体架构和通信接口设计

基于 Scilab 和 ns-3 的联合仿真平台的搭建，首先需要将整个仿真平台按功能分成模块，另外还要实现各个功能模块之间的通信。基于以上两点，本章提出了基于 Scilab 和 ns-3 联合仿真平台总体架构和通信接口设计。

2.1 基于 Scilab 和 ns-3 联合仿真平台总体架构设计

为了实现 Scilab 和 ns-3 联合仿真，需要设计一种联合仿真平台架构，联合仿真平台就依据此架构进行搭建和实施，设计架构首先需要了解网络化控制系统的仿真过程。网络化控制系统的仿真过程可描述为：在k时刻，传感器采样获得采样数据并通过网络将传感量发送给控制器，在$k+d_1$时刻控制器接收到传感量后对传感量进行计算并生成控制量（d_1是前向传输网络延时），在$k+n+d_1$时刻控制器向执行器发送控制量（n是计算延时），在$k+n+d_1+d_2$时刻执行器接收控制量计算并输出给被控对象（d_2是后向传输网络延时）。

要实现上述网络化控制系统的动态仿真，需要将网络化控制系统中的仿真建模为四大部分：传感器节点、控制器节点、执行器节点和通信网络，其中传感器节点、控制器节点、执行器节点由数学计算软件 Scilab 中的 Xcos 工具箱来仿真，因此需要在 Xcos 中建立控制系统模型，控制系统模型中包含控制器、执行器和传感器三部分，通信网络由网络仿真软件 ns-3 来仿真，需要分别在 Scilab 和 ns-3 软件内建立网络化控制系统仿真模型。

联合仿真过程中，Xcos 控制系统模型的控制器节点发送控制量给执行器节点的过程可分解为：Xcos 模型通过软件间通信工具将控制量传送给 ns-3，ns-3 获取控制量后启动 ns-3 网络拓扑的控制器发送节点，从控制器发送节点将控制量发送给 ns-3 网络拓扑的执行器接受节点，仿真完成后 ns-3 网络拓扑的执行器接受节点接受到控制量并输出 ns-3 网络拓扑发送节点到接收节点的网络仿真结果（延时和丢包信息等），之后 ns-3 将执行器接收节点接受到的控制量和 ns-3 网络仿真结果打包发送 Scilab，Scilab 解析 ns-3 仿真结果将网络仿真结果设置为 Xcos 的模型属性并将其中的控制量传递给 Xcos 内的执行器节点。Xcos 模型中传感器节点到控制器节点的传感量传递过程也类似模型中控制器节点发送控制量给执行器节点过程，这样 Xcos 模型和 ns-3 就完成了一轮联合仿真。具体的模型设置为在 Xcos 的控制

系统模型中，控制器与执行器之间和传感器与控制器之间嵌入接口用于连接 ns-3 仿真模块。网络化控制系统分模块仿真总体架构如图 2-1 所示。

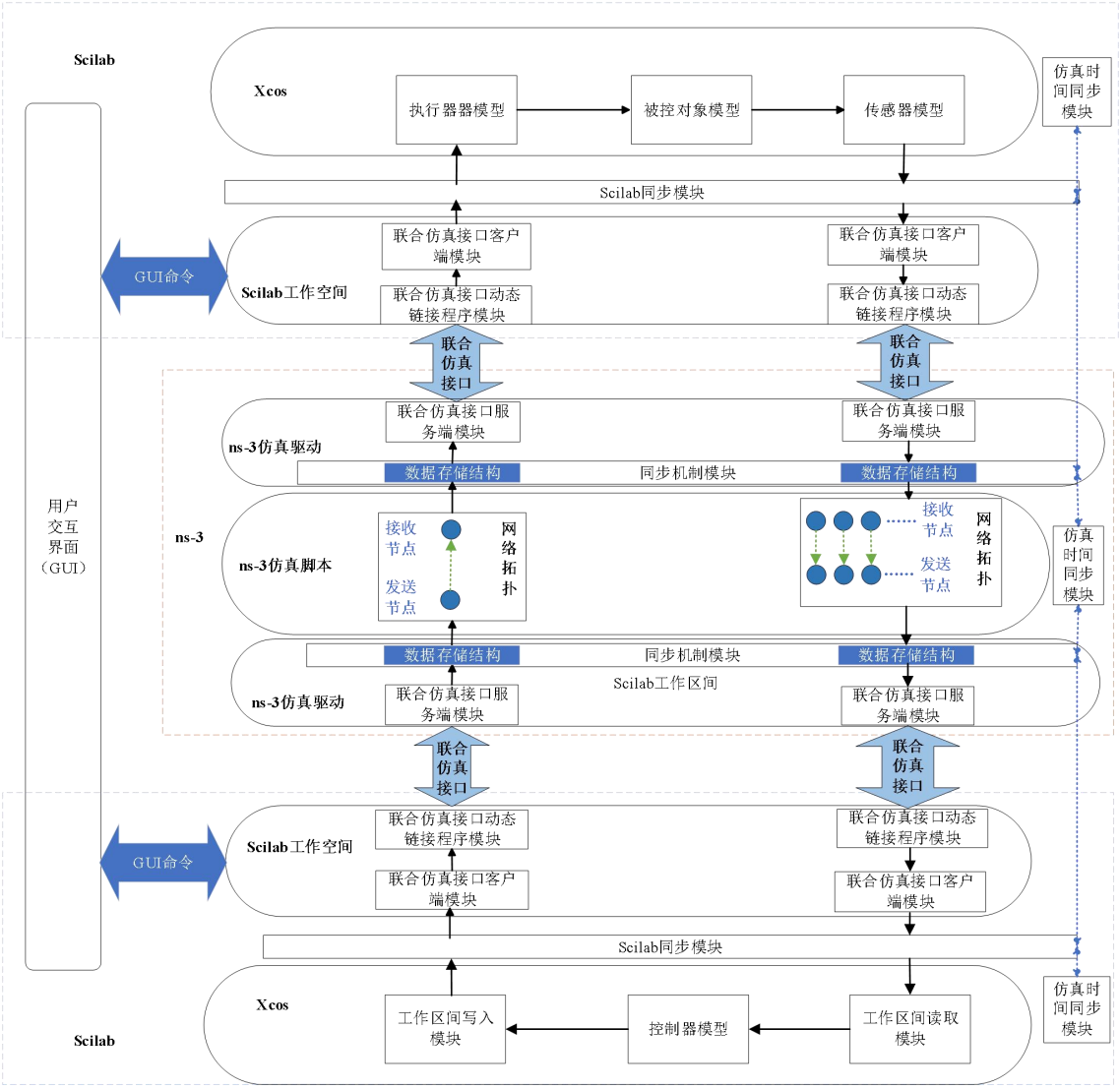


图 2-1 联合仿真平台总体架构图

Figure 2-1 NCS Cooperative Simulation Platform Architecture

2.2 基于 Scilab 和 ns-3 联合仿真平台通信接口设计

一般可以将软件在操作系统的运行过程称为进程[42]，和同一进程下不同线程之间数据易于共享不同，由于进程是操作系统分配资源的最小单位，不同进程间的数据很难共享。联合仿真平台将两个进程联合起来作为一个整体去仿真，需要解决不同进程间的数据共享问题，即软件间的通信问题。除此之外，同一进程的不同线程可以通过互斥器和加锁等方式实现有序运行从而保证进程的正常运行。

联合仿真平台也需要一种协调机制来协调两个软件的运行，保证联合仿真平台各模块时间一致以及仿真时序的正确，即软件间的时间同步问题。本节将设计一种通信接口解决软件间通信问题。

2.2.1 软件间的通信方案

为了实现在 Scilab 与 ns-3 的通信，需要设计一种联合仿真平台通信接口，借助这种接口，Scilab 能够实现向 ns-3 发送 Scilab 仿真结果和通知以及接收 ns-3 发送的网络仿真结果，同样的 ns-3 借助联合仿真接口能够实现向 Scilab 发送网络仿真结果以及接收 Scilab 仿真结果。同时接口程序还能设置客户端和服务端，通过客户端是否向服务端发送消息，服务端是否应答等方式控制整个仿真进程。

Scilab 与 ns-3 的通信属于进程间通信，根据 Scilab 和 ns-3 的软件特性确定联合仿真通信接口设计方案，由 1.2.2 节确定了软件联合仿真平台的方案，由 1.2.3 节确定了联合仿真平台通信接口以下的三种具体实现方案：

(1) 联合仿真平台通信接口框架中以 ns-3 为主导软件，在需要进行控制系统仿真的时候通信接口会调用 Scilab 进程，实现与 Scilab 的数据交换；

(2) 联合仿真平台通信接口框架中以 Scilab 为主导软件，在网络化控制系统仿真的过程需要进行网络仿真时调用 ns-3 仿真驱动程序执行仿真脚本，Scilab 控制整个仿真过程，根据需要启动和停止 ns-3 仿真；

(3) 基于 HLA（高级体系架构）的联合仿真平台[43]，HLA 为仿真软件提供接口，能够实现将每个独立的仿真软件在一个统一的时间和环境中协调运行和互相操作以第三方软件为主导[44]，使用高级体系架构的 RTI 协调控制多仿真软件实现协同仿真，高级体系架构模块(HLA)支持多仿真软件间在 HLA 平台上协同构建与运行，这样每个仿真软件仿真一个系统的一部分，再将几部分联合起来组成完成的仿真[45]。

两个软件间数据交换是两个仿真软件联合仿真的基础，而单个仿真软件内的数据传递是软件正常运行的基础，单仿真软件内的数据传递主要包括：

在 Scilab 仿真软件内数据的传递发生在以下场景：

① 仿真平台交互界面（GUI）中的参数与 Scilab 工作空间的信息交换。

② Scilab 工作空间与 Scilab 驱动程序和执行程序间的数据传递。

③ Xcos 与 Scilab 工作空间的信息交换。

在 ns-3 仿真软件内数据的传递：

ns-3 软件内的数据传递主要发生在仿真脚本与仿真驱动程序及 ns-3 仿真内核程序等程序文件之间。

2.2.2 基于 Scilab 和 ns-3 联合仿真平台连接结构设计

网络化控制系统包括控制系统和网络通信两个模块，根据网络化控制系统的

特点，将对应的联合仿真平台分为两部分分别仿真，之后将两部分仿真融合。需要在 Scilab 与 ns-3 设计连接结构，具体需要 Scilab 的 Xcos 工具箱和 ns-3 连接，连接结构如图 2-2 所示，Scilab 的 Xcos 工具箱负责控制系统的仿真，ns-3 负责通信网络的仿真，C++联合仿真程序作为连接 Scilab 与 ns-3 的桥梁工具，负责 Scilab 和 ns-3 的通信。

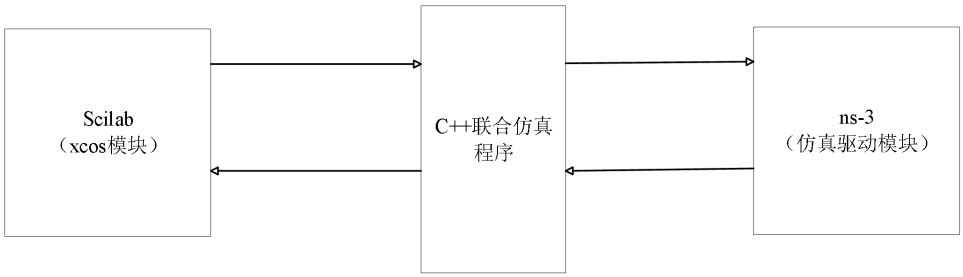


图2-2 Xcos 和 ns-3 之间的连接结构

Figure2-2．Connective structure between Xcos and ns-3

2.2.3 Linux 下的进程间通信工具 Socket 介绍

联合仿真通信接口还需要借助进程间通信的工具实现。进程间通信的方式包括管道、消息队列、共享存储、Socket 套接字等[46,47]，本文中联合仿真通信接口是基于 TCP 套接字实现的，如图 2-2 中的 C++联合仿真程序模块中包含 Socket 套接字。Socket 是计算机网络通信中的术语，又称套接字。是位于传输层的一组接口，把 TCP/IP 协议隐藏其中，这样对用户而言，在进行传输层的编程中只需要调用 Socket 编程 TCP/IP 协议，可以实现单个计算机的进程间通信或者多台计算机通过网络进行通信[48]。

Scoket 的创建：Socket 以“打开—write/read—关闭”的实现模式，Socket 连接如图 2-3 所示[49]。

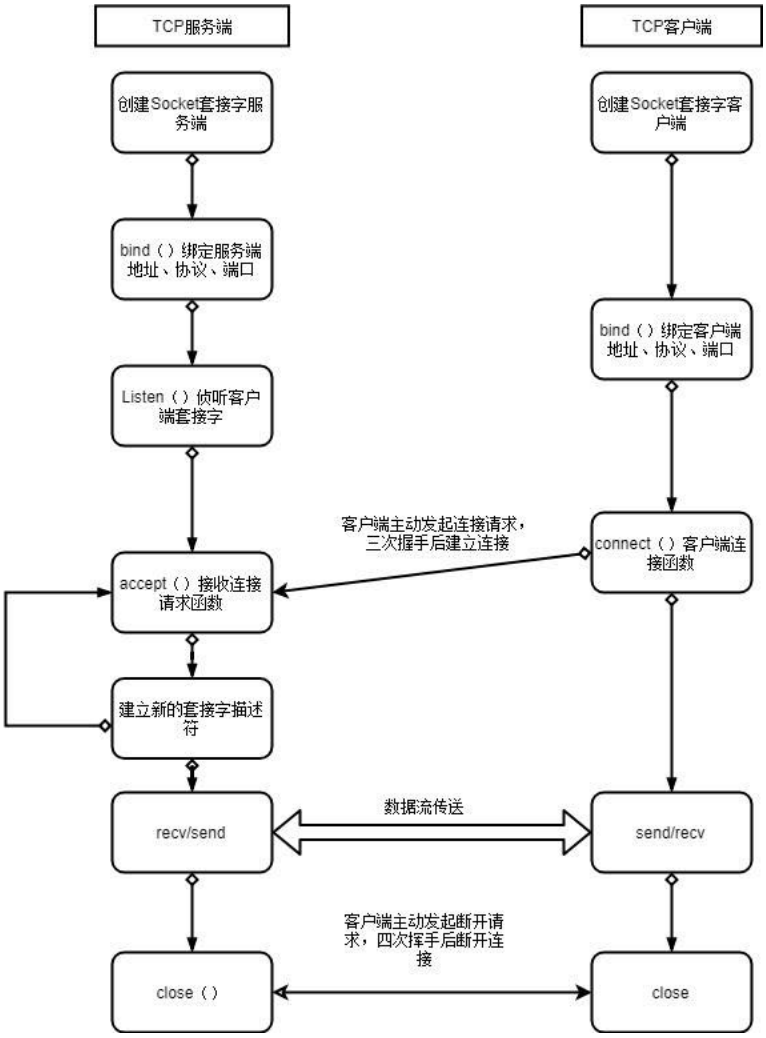


图2-3 Socket连接示意图

Figure 2-3 Scoket connect block diagram

2.2.4 软件间的通信接口架构设计

本文设计的联合仿真平台通信接口的实现方案将采用2.2.1节中的第(2)种方案，将 Scilab 作为主导软件，为 Scilab 安装通信接口客户端程序，ns-3 作为从属软件安装通信接口服务端程序，通信接口客户端控制整个仿真进程，另外通信接口程序可以实现两仿真软件联合仿真时的进程间通信。联合仿真通信接口主要的实现是基于 TCP 套接字，为了实现在 Scilab 可能接到空消息的判断，本文选用文件 I/O 作为进程间通信的辅助方式。在通信接口中实现数据在软件内和软件间的传递，并在软件与接口连接点设置数据大小和格式转换，保证软件间数据交换的正确和联合仿真平台的正常运行。联合仿真通信接口架构如图 2-4 所示。

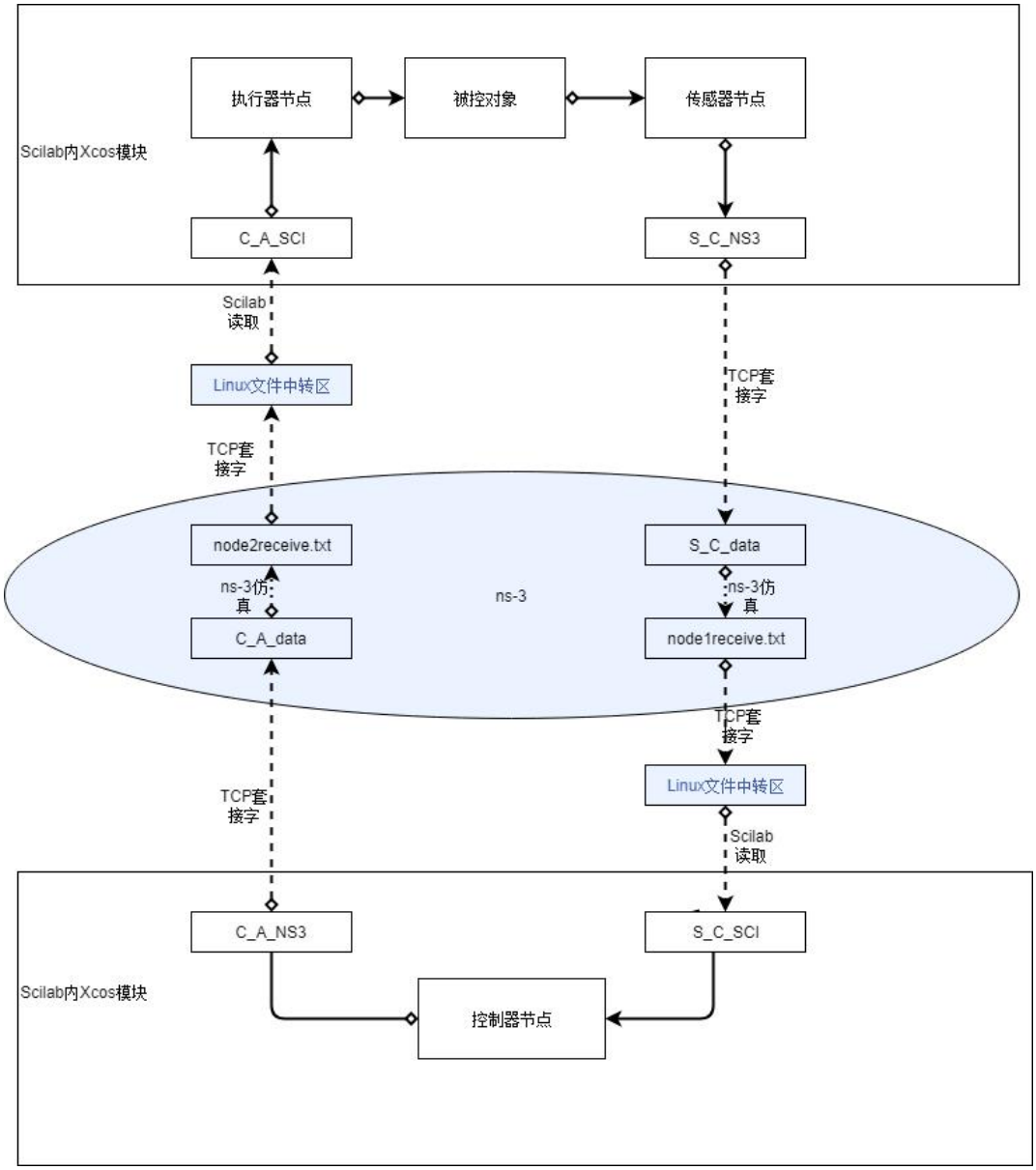


图2-4 联合仿真平台接口架构

Figure2-4. Schematic diagram of Cooperative simlation

2.3 本章小结

本章提出了一种联合仿真平台的架构，分析了搭建联合仿真平台需要解决的软件间通信问题和软件间同步的问题，介绍了软件间通信的工具 Socket 和联合仿真平台接口的架构。

第三章 联合仿真平台同步机制设计

网络化控制系统中联合仿真的软件间驱动方式各有不同，驱动方式的不同会造成软件间仿真时间不同步，从而引发系统状态不同步，导致联合仿真的失败。同步机制能够同步仿真软件的仿真时间。本章提出了一种联合仿真平台软件同步机制，用于保障联合仿真平台能够正常运行。

3.1 联合仿真平台时间同步机制总体设计

时间同步机制顾名思义是为了协调仿真平台不同软件的仿真时间，这样联合仿真平台才能保证各软件的仿真次序和实时性要求，使仿真平台正常运行。

3.1.1 软件驱动方式

如果要设计时间同步机制，必须了解软件的驱动方式，根据各个软件的驱动方式设计软件间的时间同步机制。现有软件的驱动方式主要分为事件驱动、信号驱动、状态驱动、事务（业务）驱动等。网络化控制系统中仿真软件的驱动方式主要分为时间驱动方式和事件驱动方式。

(1) Scilab 驱动方式

Scilab 虽然支持多种驱动方式，但是 Scilab 内部 Xcos 可视化仿真模块主要以时间驱动的方式运行。Xcos 的每个模块都包含有一个时钟模块，在 Xcos 运行前必须设置时钟模块。在时间驱动方式中仿真时间步长以相等的增量前进，系统状态会定期更新，直到仿真到达停止时间或系统达到某个状态。控制系统动态仿真以时间驱动方式为主，时间驱动方式如图 3-1 所示。

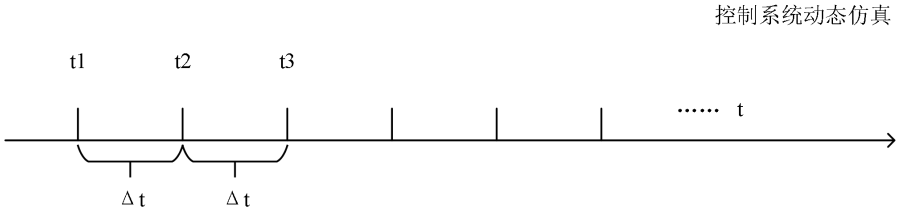


图3-1 时间驱动方式

Figure 3-1. Time-driven approach

(2) ns-3 驱动方式

ns-3 网络模拟器是事件驱动的。ns-3 是离散事件驱动的软件，内部维护一个事件列表，通过更新事件列表来驱动整个仿真过程的进行。事件驱动方式的系统

在模拟过程中响应事件而改变其状态，模拟时间从一个事件推进到下一个事件，事件之间的时间不断变化。当系统到达停止时间或系统到达某种状态时，例如事件列表中没有更多事件，整个模拟停止。事件驱动方式如图 3-2 所示。

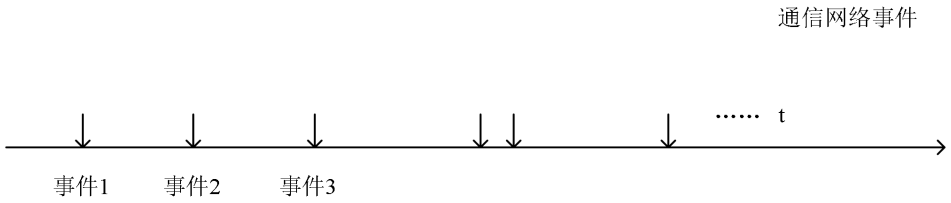


图3-2 事件驱动方式

Figure 3-2. Event-driven approach.

3.1.2 软件间的同步方式

由于参于搭建联合仿真平台的软件驱动方式不同，设计同步机制保证仿真时间同步是促使仿真成功最关键问题。对不同驱动方式的仿真器间的时间同步机制，根据仿真目标场景的不同，进程间同步的实现方式有三种不同的方案：时间步进同步方式、分步仿真同步方式和全局事件驱动同步方式[50]。

（1）时间步进同步方式：各个仿真软件独立运行它们的仿真，但在仿真软件间的交换信息的固定时间点停止并同步，如图 3-3 所示。然而，如果某些需要这些仿真软件之间交互的系统事件发生在同步点之间，则该事件必须被缓存在缓存区中并等待，直到下一个同步点被处理。因此，系统错误可能会累积并妨碍仿真的真实程度，特别是如果控制系统对时间的要求严格，并且需要控制系统和通信网络之间有大量通信接口时，错误的累积会更加严重。通常，需要为同步和数据交换设计一个通信中间件模块。

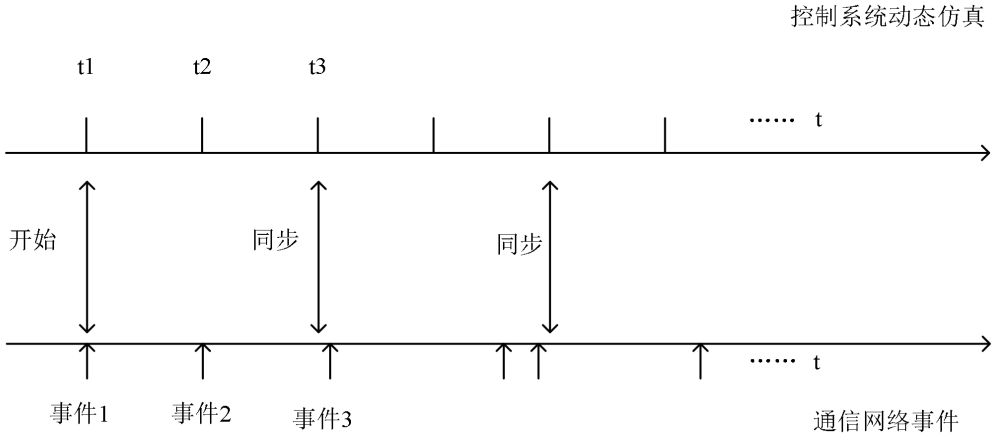


图3-3 时间步进同步方法

Figure 3-3. Time-stepped synchronization method.

（2）全局事件驱动同步方式：将传输数据的过程看成事件，按照控制系统迭代事件与其他通信网络事件的时间将这些事件安排进一个全局事件列表。按照事件的时间戳大小执行事件列表中的事件，通过这种方式维护和更新事件列表来推动仿真。该方式中一个事件的进程运行时，另一个事件进程将会停止。全局事件驱动同步方式如图 3-4 所示。这种方法可以避免时间步长方法中可能出现的潜在问题。 但是，由于使用了全局事件列表，整体的联合仿真速度非常有限。

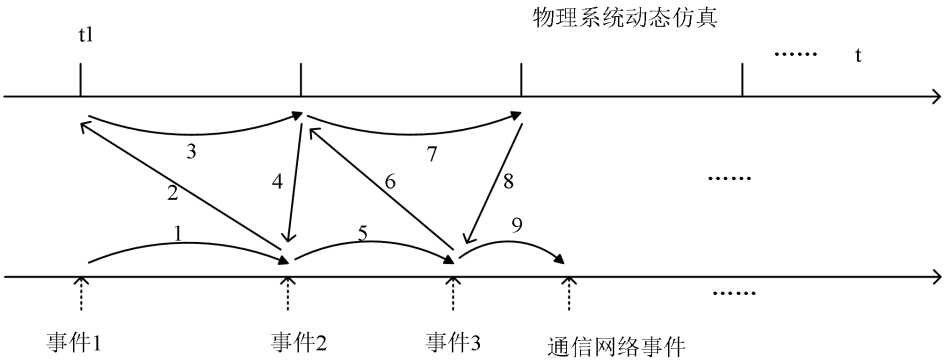


图 3-4 全局事件驱动同步方法。

Figure 3-4. Global event-driven synchronization method.

（3）分步式同步方式：也称主从式同步方式。在联合仿真过程中将参与方针的软件分为主仿真软件和从属仿真软件，主仿真软件具有更高的优先级，将协调联合仿真过程。与其它两种方法相比，它是最简单的[51,52]。

在仿真过程中，参与联合仿真的软件驱动方式的不同，仿真时间不同步的问题可能导致仿真失败，如何使两者仿真时间实现同步尤为关键。在联合仿真中，仿真时间由主仿真软件主导。如在网络化控制系统的仿真中选用 Scilab 作为主仿真软件，ns-3 作为从属软件。分别在 Scilab 与 ns-3 中建立控制节点对象和被控节点对象，t_0 时刻仿真开始，Scilab 的仿真在时间 T_0 暂停。同时，Xcos 节点模型调用相应的 ns-3 模型。初始化后，ns-3 控制对象模型在时间 T_0 暂停。Xcos 继续执行并在采样时间 T_1 暂停。它将命令传递给 ns-3 控制对象模型以激发直到时间 T_1。当 ns-3 在时间 T_1 暂停时，Xcos 节点模型从 ns-3 模型读取节点状态并生成样本数据包。总闭环延迟表示为 t_1，它是传感器到控制器的延迟加上控制器到执行器的延迟。控制对象节点会在时间 T_1+t_1 收到一个控制量，然后 Xcos 会向 ns-3 控制对象模型发出命令以使用先前的输入 u_0 执行直到时间 $T1+t_1$。ns-3 控制对象模型暂停，输入在时间 T_1+t_1 时从 u_0 更改为 u_1。当 ns-3 完成执行时，Xcos 继续以这种方式运行仿真。Xcos 和 ns-3 模型之间的同步机制如图 3-5 所示[53]。

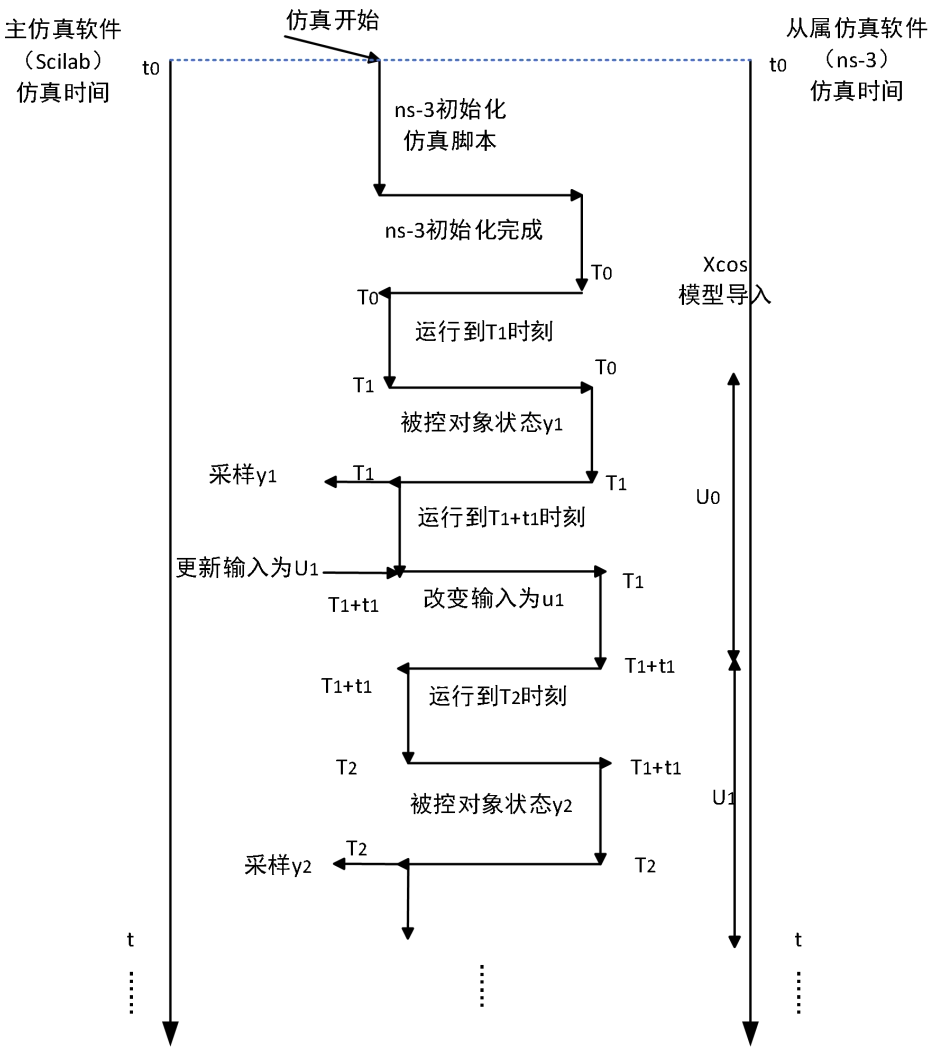


图3-5 分步式仿真同步运行机制示意图

Figure 3-5. Schematic diagram of time step synchronization operation mechanism

3.1.3 同步方式的选择

网络化控制系统对仿真实时性要求较高，通过分析上述各同步方式，本文选择结合 3.1.2 节中（1）、（3）两种时间同步方案，设计了一种主从式同步和时间步进同步结合的主从周期同步方案。主从式周期同步机制如图 3-6 所示，图中带箭头的线为执行次序。

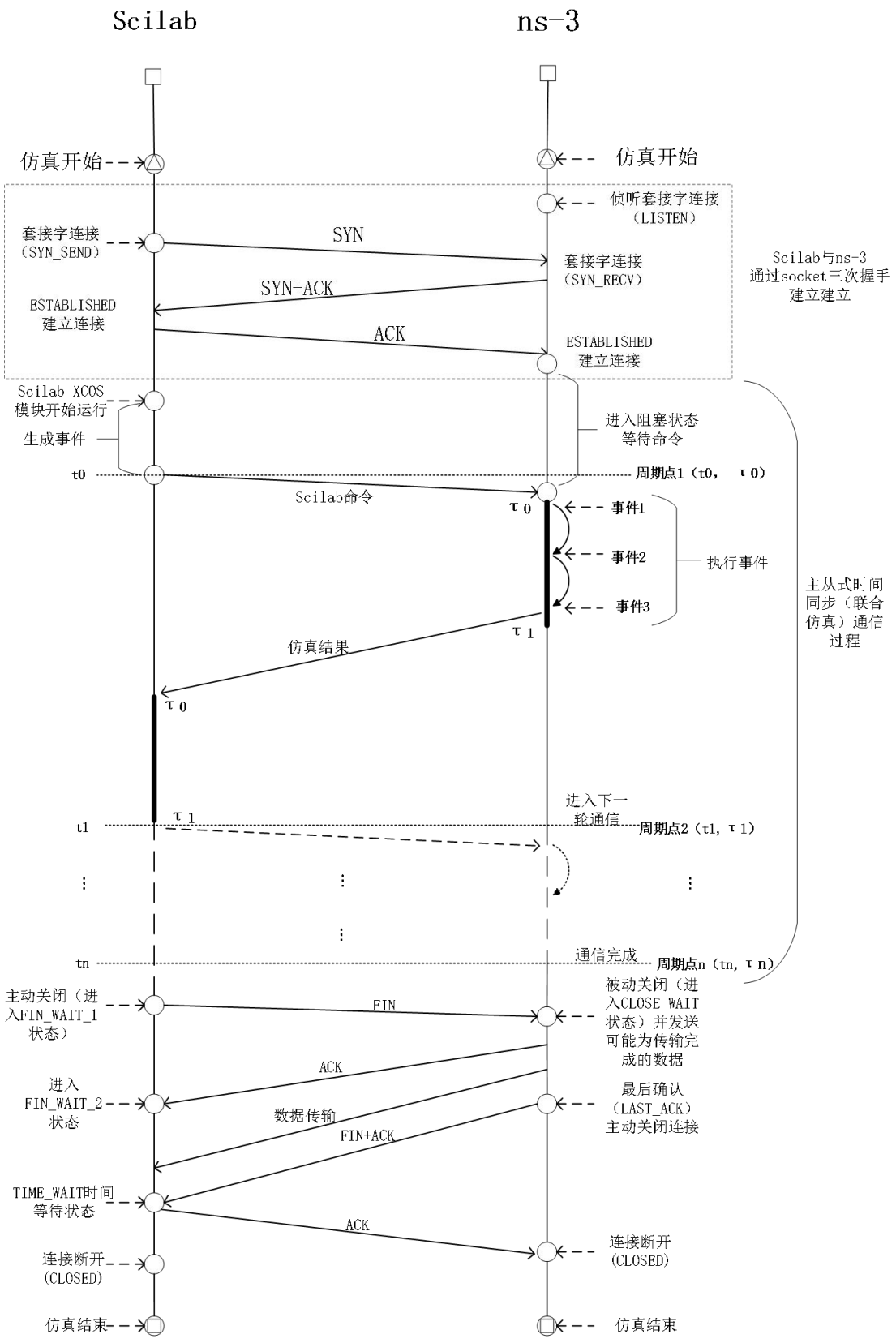


图3-6 主从式周期同步机制总体示意图

Figure3-6 Schematic diagram of master-slave cycle synchronization mechanism

主从式周期同步机制中，设置主仿真软件模式参考 2.2.1 节，借助于联合仿真接口，主从式周期同步机制将 Scilab 设置为主仿真软件，在联合仿真接口程序中为 Scilab 安装接口客户端，为 ns-3 安装接口服务端，这样 ns-3 作为服务端需要开启后一直运行，并被动接收 Scilab 发送的通知，实现了主从式周期同步仿真的主从方式。主仿真软件具有更高的优先级，它将会协调联合仿真的仿真步骤。主仿真软件设置两仿真软件的仿真开始以及结束，信息交换的时间点。

在建立连接后Scilab的Xcos模块运行，生成控制系统事件并将事件发送给ns-3后，Scilab 进入阻塞状态，ns-3 在接到通知后进行解析，安排事件并执行。ns-3 仿真时刻设置为τ_0，将τ_0到τ_1为设置的一个同步周期，在同步周期内 ns-3 执行 Scilab 安排的事件，并根据 ns-3 内同步机制判断是否在τ_0时刻之前发送。Scilab 内具体机制设计将在 3.2.1 节中介绍。ns-3 内具体机制设计将在 3.3.1 节中介绍。ns-3 执行同步将仿真结果发送给 Scilab，ns-3 进入阻塞状态，Scilab 仿真时刻进入τ_0，进入下一个周期的仿真，直到仿真达到结束条件。

3.2 Scilab 内同步机制设计

在 Scilab 编写接口程序，Xcos 建立控制系统模型，模型内包含控制器 C、执行器 A、传感器 S，在控制器与传感器间设置 S-C 的 ns-3 接口，执行器与控制器间设置 C-A 的 ns-3 接口，接口程序内包含 Socket 套接字与 ns-3 通信。

3.2.1 Xcos 控制系统模型设计

在 Scilab 中进行控制系统的建模，为了使建立的模型尽可能通用，减少仿真平台使用的难度。模型需要建立控制器模块、传感器模块、执行器模块，并在这些模块之间设置连接通道，这些连接通道充当 NCS 中通信网络，在其上搭建联合仿真平台接口，通过接口实现与 ns-3 的通信。控制器、执行器和传感器的模块都可以进行编辑，可应对不同仿真模型，被控对象包含 State，用来传送更新后的状态量。搭建的 Xcos 模型如图 3-7 所示，其中变量 C_A_SCI 为获取 ns-3 发送控制量，C_A_NS3 为 Scilab 发送给 ns-3 的控制量，S_C_NS3 为获取 ns-3 发送传感量，S_C_SCI 为 Scilab 发送给 ns-3 的传感量。

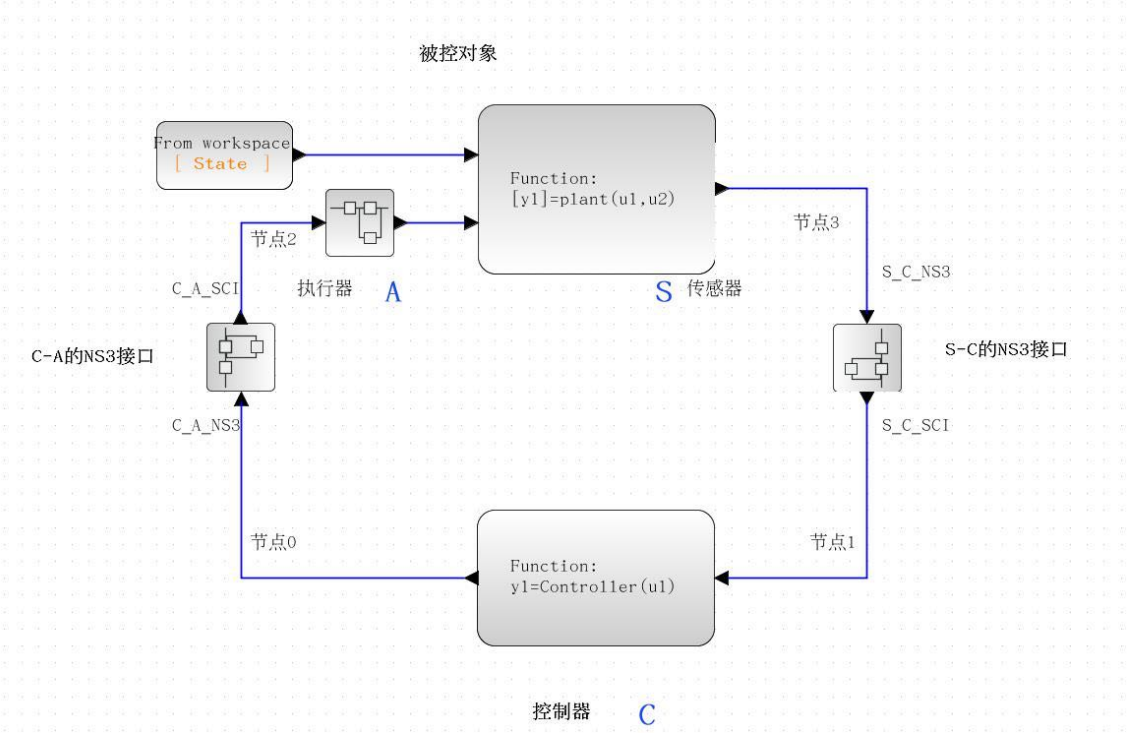


图3-7 Xcos控制系统模型

Figure 3-7 Control system model of Xcos

搭建 Xcos 的模型包含若干个从 Scilab 工作空间读数据模块（From workspace）和向 Scilab 工作空间写数据模块（To workspace）并嵌入到 Xcos 的控制器模块和被控对象模块中。

搭建 Xcos 模型包含若干个从 Scilab 工作空间读数据模块和向 Scilab 工作空间写数据模块，Xcos 模型可看为两大部分，控制模块和包含被控对象、执行器和传感器的被控对象模块，这两大模块在 ns-3 内被建模为两个节点；From workspace 和 To workspace 分别嵌入到控制器模块和被控对象模块中。控制器与执行器之间的接口模块包含一个 From workspace，其读取的变量名为 C_A_SCI，一个 To wprkspace 其写入的变量名为 C_A_NS3。传感器与控制器之间的接口模块包含一个 From workspace 变量名为 S_C_NS3，一个 To workspace 变量名为 S_C_SCI，负责 Xcos 模型与 Scilab 工作空间的数据传递，另外在执行器中设置根据数据包时间戳选择最新数据包的功能也需要一对读写模块，状态量模块需要设置一个读模块用于读取模型的状态量。

基于 Xcos 和 ns-3 的联合仿真时间同步方法是实现 Scilab 和 ns-3 联合仿真网络化控制系统的关键技术。软件间的联合仿真接口是实现联合仿真的基础。

创建被控对象模型需要在Xcos中使用用户自定义功能模块scifunc_block_m，并在工作区间中创建后缀为.sci的Scilab文件编写模块功能。定义被控对象模块的控

制量u_1，系统输入状态x_1，系统输出状态y_1，以及系统输出z_1。被控对象功能模块使用u_1和x_1，得到控制系统的仿真结果输出y_1和z_1。模块函数形式为 $[y_1，z_1] = plant（x_1，u_1）$。以线性系统为例，函数实现如下：

$$y_1 = Ax_1 + Bu_1，\ z_1 = Cx_1 + Du_1 \quad （3\text{-}1）$$

其中A、$C \in R^{n*n}$，B、$D \in R^{n*m}$，为给定矩阵。

搭建控制器模型：使用用户自定义功能模块scifunc_block_m新建控制器模块，模块接受ns-3仿真结果数据作为模块的输入，仿真结果的数据内容除了包括用于记录数据发送时刻的时间戳time_stamp，还包括发送的数据传感量，记为u_2，控制器模块的输出为y_2，模块的函数形式设为为[y_1]=controller(u_2)。模型可根据仿真实际需要的控制算法编写，如以线性状态反馈的数学模型如（3-2）所示。

$$y_2 = [\text{time\_stamp}，\text{K}u_2] \quad （3\text{-}2）$$

其中K为给定控制增益。

搭建执行器模型：执行器模型设计具有选择最新的控制量执行的功能，因此设计如下机制，首先定义变量ut_memory,同时定义用于更新ut_memory的变量ut_memory_update，ut_memory_update存入的是上个仿真周期使用过的包含时间戳的控制量（时间戳最大的控制量），ut_memory_update存入Scilab的工作区间。仿真开始后Xcos读取存入Scilab工作区间ut_memory_update的值并赋给ut_memory，之后将本仿真周期到达执行器的控制量数据包与ut_memory比较时间戳（传感器节点发送数据包的时刻）依此来选择最新的控制量，选择时间戳大的控制量数据包执行，也就是最新的数据包，并将此数据包保存到ut_memory_update变量中。ut_memory_update存入Scilab工作区间用于下个仿真周期更新ut_memory的值。

3.2.2 Scilab 时间同步机制设计

在Scilab工作空间中初始化Xcos模型，给定传感器节点经过采样后获得的传感量，记为C_A_data，C_A_data需要经过通信网络发送给控制器节点，给定控制器节点输出的控制量，记为S_C_data，控制量S_C_data经过通信网络发送给执行器节点，启动接口程序将控制量C_A_data和传感量S_C_data的值传递给ns-3，ns-3会根据接收到的数据包内容安排仿真事件并执行仿真。

ns-3创建两个发送节点和两个接收节点用于执行发送和接受事件，完成这些事件后，ns-3会启动接口程序传送仿真结果给Scilab，Scilab接受ns-3的仿真结果，获取仿真结果中当前的仿真时刻，并将仿真时刻赋值给Scilab的当前仿真时间变量CurrentTime_SCI；Scilab定义状态变量State，将Xcos运行后生成的传感量S_C_NS3的值传递给State，完成State的初始化，之后获取ns-3仿真传感器节点发送传感量到控制器节点的仿真结果，将数据包的内容赋值给传感量S_C_SCI，获取ns-3仿真控制器节点到执行器节点的结果，将数据包的内容传给控制量C_A_SCI。

运行Xcos控制系统模型，获得运行结果，将结果安排为需要ns-3下次仿真的事件，即控制器节点发送给执行器节点的数据包的内容传给控制量C_A_NS3，传感器节点发送给控制器节点的数据包内容传给传感量S_C_NS3。

设置循环体的次数变量t，表示传感器采样的次数，次数t乘以采样周期为仿真总时长。进入循环体后，将上个采样周期内Xcos运行的仿真结果控制量C_A_NS3和传感量S_C_NS3传给协同仿真接口程序的接口变量C_A_data和S_C_data，运行联合仿真接口程序，将Scilab需要仿真的事件发送给ns-3，并接受ns-3仿真的结果，将延迟信息存储如延时文件中，数据包的内容用于更新传感量S_C_SCI和控制量C_A_SCI，之后运行Xcos控制系统模型，并在Scilab工作区间中更新当前仿真时刻CurrentTime_SCI增加一个SimulationStep，之后将次数变量t加1，进入下次循环，直到达到仿真总的时间上限，定义并初始化控制系统闭环延时close_loop_delay，close_loop_delay的计算公式如（3-3）所示。

$$close\_loop\_delay=CurrentTime\_SCI-time\_stamp \quad (3\text{-}3)$$

同步伪代码示例如下所示。

Scilab同步机制伪代码
1. 如果当前时间小于结束时间
2. 当如果接受到信息进入循环体
3. Xcos 运行的控制量结果赋值接口程序变量
4. Xcos 运行的传感量结果赋值接口程序变量
5. 给状态变量赋值
6. 运行接口程序传送给 ns-3，并接受 ns-3 的仿真结果
7. 从文件中读取数据到 Scilab 工作空间
8. 从文件中读取数据到 Scilab 工作空间
9. 读取的传感量数据给 Xcos 模块赋值
10. 读取的控制量数据给 Xcos 模块赋值
11. 运行 Xcos 模型
12. 更新仿真时间

3.3 ns-3 内同步机制设计

在整个仿真过程中，主从式同步机制的执行次序如图 3-8 所示。根据主从式周期同步机制的执行次序，本节设计了 ns-3 内的同步机制。

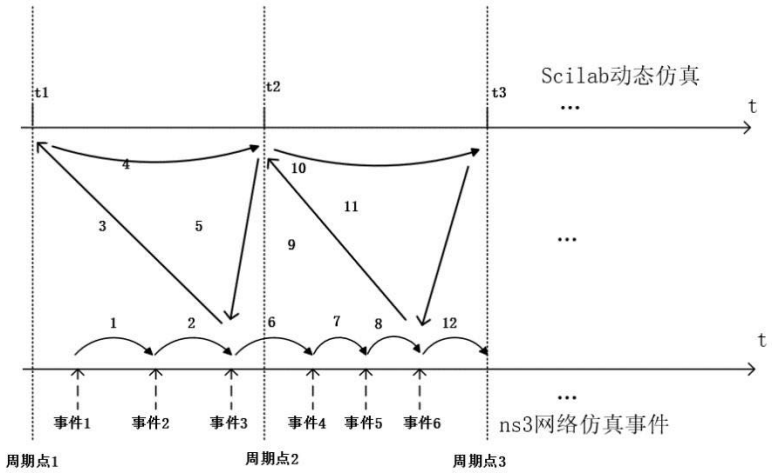


图3-8 主从式周期同步机制仿真过程同步图

Figure 3-8. Master-slave synchronization method.

3.3.1 同步状态

本文综合考虑联合仿真平台几种同步方案，选用的主从式周期同步机制，其中在 ns-3 内的同步机制如图 3-9 所示。

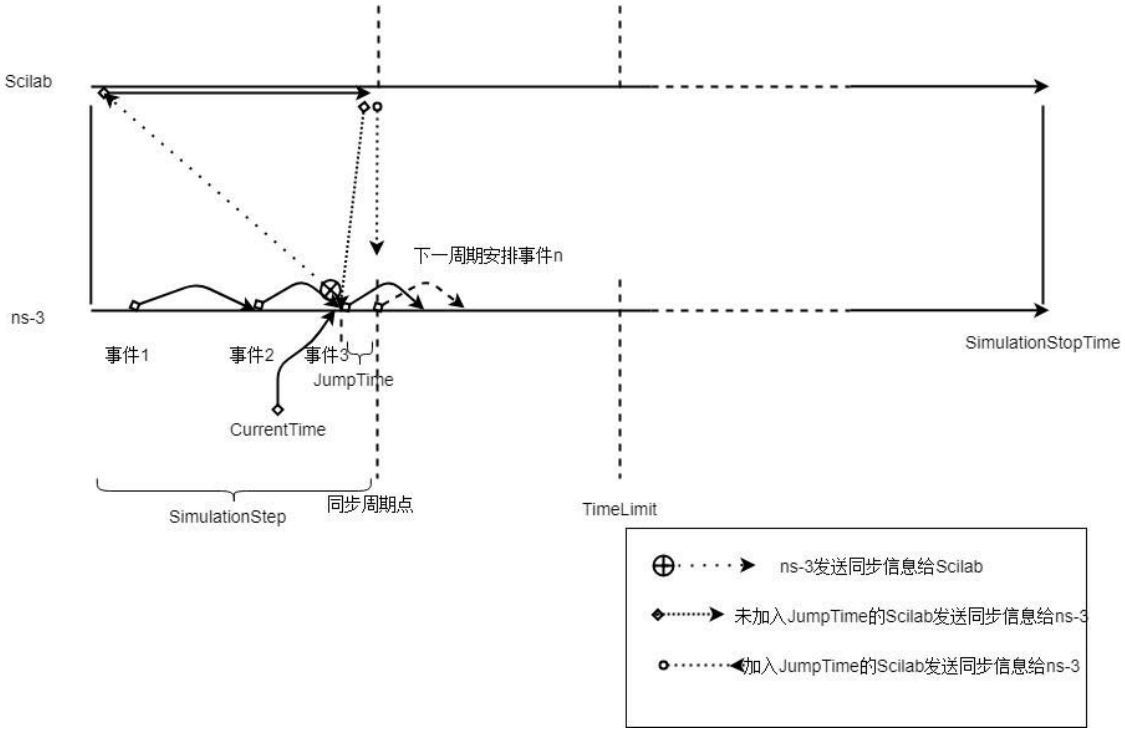


图3-9 ns-3周期同步周期

Figure 3-9. ns-3 synchronization mechanism

本仿真平台采用的主从式周期同步机制，由 Scilab 作为主导软件，ns-3 作为从属软件，在周期点发送仿真结果并传送给 Scilab，之后 ns-3 进入阻塞状态暂停仿真，等待接受 Scilab 的信息，在接受到 Scilab 的信息后对信息进行解析，安排需要 ns-3 执行的事件。

时间同步机制为在同步时间点 m_TimeLimit 完成事件同步，即 ns-3 将仿真结果发送给 Scilab、接收 Scilab 仿真结果、安排下一周期的节点发送/接收事件，如图 3-9 所示。用 ScheduleTransmit()函数安排事件，事件用 SendTo()函数表示。为了实现上述过程，需要先由当前 ns-3 仿真时间 m_currentTs 与 m_TimeLimit 比较，当 m_currentTs < m_TimeLimit 时，获取下一个事件的执行时间 f_next.key.m_ts，如果 f_next.key.m_ts≥m_TimeLimit,意味着下次事件执行时间在下一周期内，本周期内已经没有事件需要执行，所以运行 ns-3 发送仿真结果给 Scilab 的函数 TransmitNotices()与 Scilab 进行同步，并更新 m_TimeLimit，使其增加一个步长 g_SimulationStep。

$$m\_TimeLimit=m\_TimeLimit+g\_SimulationStep \quad (3\text{-}4)$$

由于 m_TimeLimit≠m_currentTs，ns-3 运行 Listen()函数接收 Scilab 仿真结果后，需要延时 JumpTime 后使用 ScheduleTransmit(JumpTime)安排在下一周期，ScheduleTransmit()函数包含了附录 4 中提到的 Schedule()函数。跳跃时间 JumpTime 计算如下：

$$JumpTime=m\_TimeLimit-g\_SimulationStep-m\_currentTs \quad (3\text{-}5)$$

这样安排的事件就可以进入下一周期内执行。Scilab 同步代码示例如下所示。

同步机制伪代码

```
1. 如果当前时间小于同步点
2.     返回下个事件执行时间
3.     如果下个事件执行时间大于同步点
4.     向 Scilab 发送仿真结果
5. 更新同步点
6.     计算需要的跳跃时间
7.     接受 Scilab 的通知
8. 将下个事件安排到跳跃时间执行
```

上述代码表示 ns-3 进行同步的条件和需要执行的操作。

3.3.2 阻塞状态

在 ns-3 同步消息发出后，ns-3 就会进入阻塞状态。例如在图 3-8 中，取

m_TimeLimit=周期点 2， m_currentTs=事件 3， m_currentTs < m_TimeLimit，f_next.key.m_ts=事件 4≥ m_TimeLimit，ns-3 发送仿真结果给 Scilab，如箭头 3 所示，并更新 m_TimeLimit =周期点 3，ns-3 进程阻塞。Scilab 执行一个周期，如箭头 4 所示，并发送仿真结果给 ns-3，如箭头 5 所示，Scilab 进程阻塞，ns-3 接受仿真结果。JumpTime 计算公式如式（3-6）。

JumpTime=周期点 3- g_SimulationStep-事件 3 （3-6）

阻塞状态结束后，ns-3 才会调用 Simulator::Schedule(JumpTime, ,)安排新事件的执行。

3.4 本章小结

本章首先分析了联合仿真平台软件的驱动方式，介绍了几种同步机制，分析了适合本文仿真平台的同步机制。之后介绍了本文联合仿真平台设计的主从式周期同步机制，以及同步机制在 Scilab 与 ns-3 内的实现方式，联合仿真平台采用主从式周期同步机制，能够符合联合仿真的实时性和通信要求。

第四章 联合仿真平台设计实现

在选择了联合仿真平台总体的通信接口和时间同步机制设计方案之后，本章是通信接口和时间同步机制在 Scilab 和 ns-3 软件中的具体程序实现。

4.1 Scilab 内程序实现

在 Scilab 内需要设计联合仿真平台通信接口客户端部分和时间同步机制程序，本节介绍了在 Scilab 联合仿真平台通信接口和时间同步机制程序的具体实现，以及 Xcos 模块的驱动程序和在 Scilab 中搭建联合仿真平台 GUI 的方案。

4.1.1 联合仿真接口程序实现

Scilab 作为主控程序，需要在软件内设计仿真进程控制模块、控制系统模型和通信接口。在 Scilab 内的控制系统模型安装接口才能实现与 ns-3 的通信，Scilab 能够使用命令“ilib_mex_build”创建 mex 库以及生成加载程序文件，用于动态加载 mex 共享库。

编写包含 Socket 的 C++文件 CLIENT.cpp，使用命令“ilib_mex_build”将 CLIENT.cpp 加载到 mex 库，之后在 Scilab 通过调用加载程序文件即可使用 CLIENT.cpp 文件，在 Scilab 内设计的程序文件如下：

（一）CLIENT.cpp 文件：

主函数：mexFunction()，作为接口函数，负责 Scilab 与 ns-3 通信。

变量 mexArray 类型指针：nodenum、packetct、packetcte、order_sys，从 Scilab 工作区间获取 nodenumber、C_A_data、S_C_data、order 的值。

通知结构体 struct：将需要传送和接受的数据装入结构体。

Socket 套接字函数：sock，只需绑定本机地址，发送和接受通知结构体；

文件转存功能块：将接受到的数据存入文件，用于 Scilab 读取。

Scilab 与 ns-3 通信的载体结构如表 4-1 所示。

表 4-1 Scilab 与 ns-3 通知结构体

Table4-1 Scilab and ns-3 notification structure

Scilab发送给ns-3通知结构体	ns-3发送给Scilab通知结构体
Data_Scilab	Data_ns-3
sig	sig
des	packetcontent
packetcontent	Timedelay
sys_send_node	Sim_time
pos	Send_time

结构体 Data_Scilab:发送标志 sig、目标节点 des、数据包内容：packetcontent、系统发送节点标志位 sys_send_node、位置 pos、移动速度 vel。

Data_ns-3：发送标志 sig、数据包内容：packetcontent、时间延时 Timedelay、数据包的接收时间 Sim_time、数据包的发送时间 Send_time。

（二）ns-3 启动程序

在 Scilab 中可以通过调用 unix 命令“terminal”启动终端并执行命令，首先在 Scilab 中打开 ns-3 脚本文件路径，之后使用终端命令启动 ns-3，通过在 Scilab 内系统 API 接口调用 Linux 命令启动并初始化 ns-3，初始化 ns-3 节点个数 nodenumber = 4，ns-3 的模拟脚本命名为 ExternallyTest，由于需要给已经编译过的 ns-3 重新赋节点数，所以每次启动协同仿真前需要使用命令 unix(‘gnome-terminal -- ./waf –run scratch/ExternallyTest’)启动并重新编译 ns-3。

（三）加载 CLIENT.cpp 到 mex 动态库的程序，

Scilab 和 ns-3 的联合仿真中需要进行仿真结果数据的交换，编写了一种基于 TCP 套接字的联合仿真接口程序用于完成 Scilab 和 ns-3 的数据交换，Scilab 软件不包含 Socket 套接字功能，需要通过 Scilab 的官方 API 接口 api_scilab 实现将 C++的功能扩展程序写入 Scilab，api_scilab 允许扩展程序定义本机函数（gateway），获取 gateway 函数的输入参数，设置 gateway 函数的返回参数并访问局部变量，使用通用的辅助函数访问错误或警告信息，通过扩展的 C 或 C++程序，可以实现 Scilab 内部变量与外部应用程序的数据交换。

加载 CLIENT.cpp 程序到 mex 动态库，并命名为 notify()，之后需要用到 CLIENT.cpp 程序。执行命令 ilib_mex_build([“接口名” “C++程序文件名”]，编译接口程序。之后执行命令 exec loader.sce，链接 Scilab 动态库文件，在 Scilab 中通过“接口名”即可调用接口程序，即只需在 Scilab 工作区间键入命令“notify()”。

4.1.2 Xcos 驱动程序实现

在 SciNote 内编辑 sci 文件，用于驱动 Xcos 模型，其中的变量定义如下：

order：控制系统阶数；nodenumber：节点总数；C_A_data：加载程序从 Scilab 读取的需要发送给 ns-的控制量的变量；S_C_data：加载程序从 Scilab 读取的需要发送给 ns-的传感量的变量； current_time：时间变量；State：状态量；S_C_SCI：ns-3 仿真结果解析的传感量；C_A_SCI：ns-3 仿真结果解析的控制量；ut_memory：执行器模块输入值；ut_memory_update：执行器模块更新值；S_C_NS3：控制器需要发送给 ns-3 的传感量；C_A_NS3：Scilab 需要发送给 ns-3 的控制量；outtemp：中间量，用于保存过程值。node1recv_time：节点 1 接收时间；node2recv_time：节点 2 接收时间；临时变量 temp1 和 temp2：用于临时存放 ns-3 接受的延迟数据信息。

初始化 ns-3 命令：在 Scilab 工作区间定义用于 ns-3 的变量并赋值，包括 nodenumber、C_A_data、S_C_data、order，启动加载程序文件 notify()。

编写 Scilab 联合仿真循环程序：交互一次会驱动 Scilab 和 ns-3 仿真一轮，即 Scilab 和 ns-3 把接收到的一次数据进行模拟，并到得一次结果输出。要驱动协同仿真运行需要的次数，就需要 ns-3 与 Scilab 工作区间以及 Xcos 仿真模块持续交换数据，进行多轮仿真。

定义时间变量 current_time，给定初始值为 0。定义状态量 State 并给定初值。定义步数变量 t，初始值为 0，用于设定总的仿真时长，仿真时长=t*仿真步长。

设置 while 主循环函数，给定 t 设定值，Xcos 输出数据变量 C_A_NS3 和 S_C_NS3 赋给 C_A_data 和 S_C_data，运行一次在 Scilab 编译过的扩展程序，运行命令 notify()，启动加载程序读入 C_A_data 和 S_C_data 并通过 Scoket 发送给 ns-3，同时接受 ns-3 上次仿真的结果，存放到 node1receive.txt 和 node2receive.txt 文件中，命令 loadmatfile 从 node1receive.txt 和 node2receive.txt 文件中读取的数据，存入 S_C_SCI 和 C_A_SCI 中，作为 Xcos 的输入数据值，命令[result]=importXcosDiagram 和 Xcos_simulate(scs_m, 4)启动 Xcos，Xcos 开始运行，从 Scilab 工作空间读取 S_C_SCI 和 C_A_SCI 的值并开始仿真，命令 loadmatfile 从 node1receive_delay.txt 和 node2receive_delay.txt 读取延迟信息赋给中间变量 midnumber1 和 midnumber2，运行命令 temp1=[temp1,midnumber1]和 temp2=[temp2,midnumber2]将每次运行得到的延迟信息保留下来，命令 fprintfMat 将 temp1 和 temp2 放入名字为 delay1.txt 和 delay2.txt 的文件中，命令 variable=[variable,in2],将每次运行的 node2receive 信息保存下来，命令 fprintfMat 将 variable 放入名字为 state.txt 的文件中，用 end 结束循环。Xcos 驱动程序流程示意如图 4-1 所示。

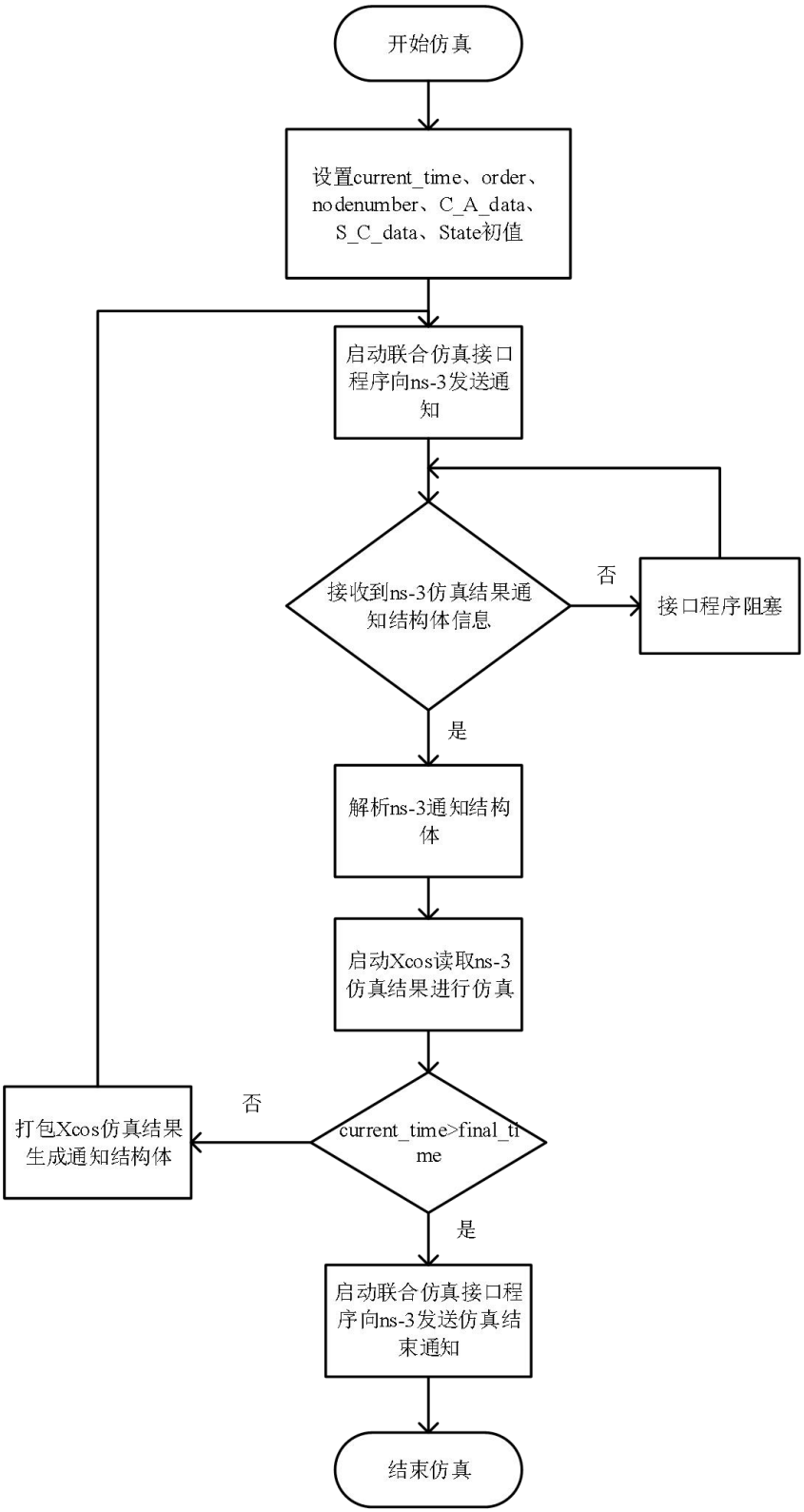


图4-1 Xcos驱动程序流程图

Figure 4-1. Driver flowchart of Xcos

4.1.3 Xcos 模型各节点模块实现

Xcos 模型如图 3-7 所示，Xcos 模型控制器节点数学模型示例如表 4-2 所示。

表 4-2 控制器节点数学模型示例

Table 4-2 Example of controller node mathematical module

控制器模型	
1.	function [y1]=Controller(u1)
2.	A=[0;0]
3.	y1=[u1(1,1),[u1(2,1),u1(3,1)]*A]
4.	endfunction

其中 u1 是控制器的输入，u1(1,1)是时间戳，用来记录传感器的发送时刻。

被控对象节点数学模型示例如表 4-3 所示。其中被控对象输出为 y 和 z。

表 4-3 被控对象节点数学模型示例

Table 4-3 Example of plant object node mathematical module

被控对象模型	
1.	function [y,z]=plant(x,u)
2.	A=[0.9,0;0,0.8];
3.	B=[1;0.5];
4.	C=[1,0;0,1];
5.	y=A*x+B*u;
6.	z=C*x;
7.	endfunction

执行器节点数学模型示例如表 4-4 所示。其中输入为 u1 和 u2，u1(2,1)和 u2(2,1)是数据包的时间戳，判断输入的时间戳的大小用于选取最近的输入数据执行运算。Xcos 驱动代码总结见附录 5。

表 4-4 执行器节点数学模型示例

Table 4-4 Example of actuator node mathematical module

被控对象数学模型	
1.	function [y1,y2]=Actuator(u1,u2)
2.	if u1(2,1)<u2(2,1) then
3.	y1=[u2(3,1)];
4.	y2=u2;
5.	else
6.	y1=[u1(3,1)];
7.	y2=u1;
8.	end
9.	endfunction

4.1.4 图形用户界面实现

图形用户界面 GUI 是为了方便用户使用，在 GUI 界面上，用户可以设置 ns-3 的属性，可以进行 Scilab 内 Xcos 控制模型的编辑，联合仿真结果的输出，仿真的停止等功能。在 Scilab 编写 GUI 界面，需要先在 Scilab 的安装 GUI Builder 工具箱，在 Scilab 工作区间输入命令“guibuilder”即可启动 GUI 编辑器。编写联合仿真平台 GUI 界面，可以实现在 GUI 上进行 ns-3 的启动和编译以及通过命令行设置 ns-3 仿真脚本中的自定义变量和属性变量。

本文联合仿真平台 GUI 界面的模块设置如下：

(1) ns-3_initial：ns-3 初始化设置。

(2) Scilab_initial：Scilab 初始化设置。

(3) Module_set：Xcos 模型设置。

(4) Controller_set:编辑控制器模块。

(5) Plant_set:编辑被控对象模块。

(6) Actuator_set:编辑执行器模块。

(7) START:仿真开始按钮。

(8) Delay_plot:绘制延迟图。

(9) Out_plot:绘制输出图。

(10)Clear_plot:清空绘图按钮。

(11)Axe:绘图区域。

(12)text:界面标题命名为“SCILAB_NS3_COSIMULATION_GUI”。

在使用联合仿真平台时，首先运行后缀.sci 的 GUI 程序，在 GUI 程序中通过代

码调用 Xcos 控制系统模型的 4.1.2 节中的各节点模型，GUI 调用节点模型代码示例如表 4-5 所示，调用后可直接编辑节点模型。

表 4-5 GUI 调用节点模型代码示例

Table4-5 Example of GUI call node module code

GUI调用节点模块代码	
1.	function “模块名称”(handles)
2.	exec('模块程序目录', -1)
3.	endfunction

程序启动后，GUI 界面如图 4-2 所示，通过 GUI 进行联合仿真平台的各种模块设置，仿真控制等。

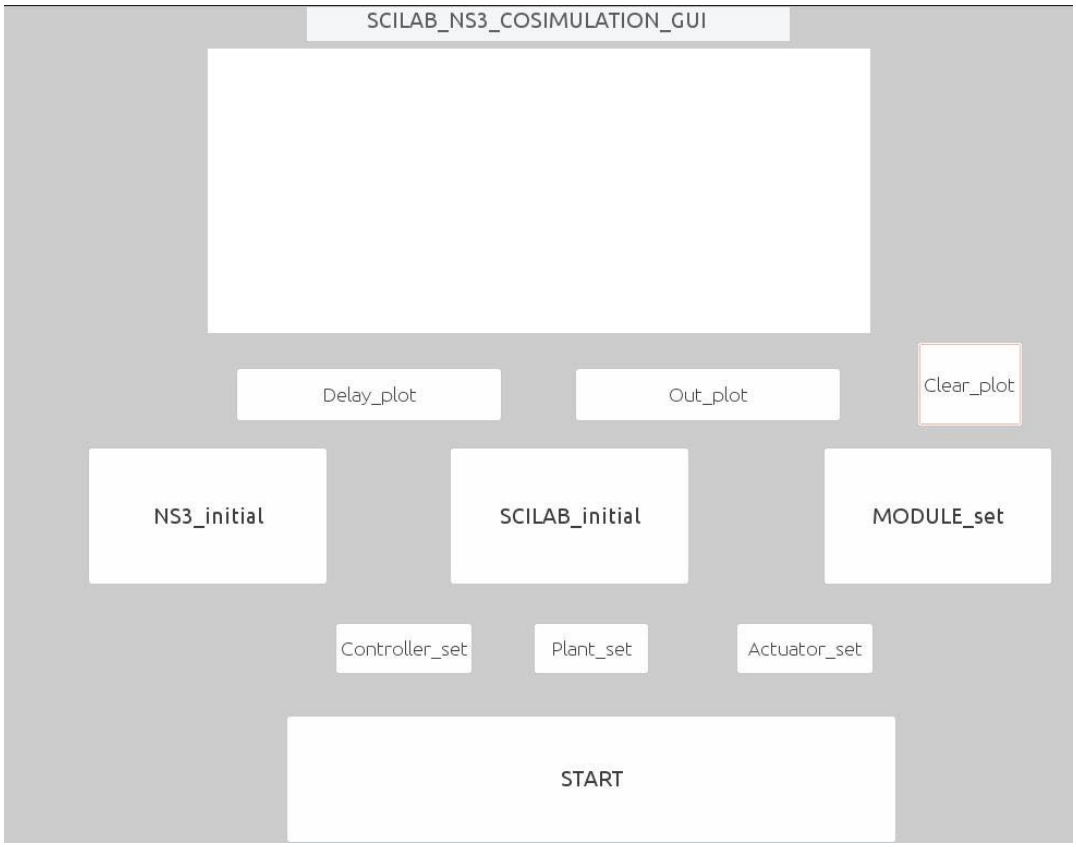


图4-2 联合仿真平台GUI界面

Figure4-2. Co-simulation platform GUI interface

4.2 ns-3 内程序实现

本节将主要介绍 ns-3 仿真脚本的具体程序实现以及搭建联合仿真平台需要修改的 ns-3 程序文件结构，主要包括表 4-6 中的程序文件。

表 4-6 ns-3 主要程序文件

Table 4-6 ns-3 main program files

文件名	作用
ExterallyTest.cc	ns-3 仿真脚本程序
externally-driven-sim.h	驱动内核程序，联合仿真接口程序
udp-echo-new-helper.h	设置属性，安装应用程序
udp-echo-client-new.h	启动客户端应用程序
udp-echo-server-new.h	启动服务端应用程序
tcp-server.h	启动 tcpserver 应用
tcp-client.h	启动 tcpclient 应用

4.2.1 ns-3 仿真脚本程序

仿真脚本程序是 ns-3 仿真时需要直接运行的程序，脚本程序包含 ns-3 必要的头文件，core 模块、network 模块、internet 模块、applications 模块以及内核驱动模块，包含网络拓扑的建立、安装网络协议、安装应用程序、启动与结束的 Run 和 Stop 等，详见 1.4.2 节。设计 ExterallyTest.cc 的 main()函数是入口函数，包含 ns-3 初始化和属性配置函数 configure()和仿真启动函数 Run()。

（1）属性配置函数 configure()

configure()函数用于配置 ns-3 仿真属性，首先从 cmd 命令行中读取命令，将命令进行赋值相应的变量，ns-3 获取这些变量的值可用于仿真环境的搭建，即初始化 ns-3 环境属性。其中包括一些基本仿真属性配置，如仿真总时长 totaltime，跟踪文件生成标志位 pcap，仿真步长 simulationstep，服务端口 servport，仿真中需要安装的协议 protocol，节点接收的能量 RxGain，设置延时标志位 Setdelay，丢包率 lossrate 等。

（2）网络搭建与仿真开始函数 Run()

调用函数 configure()进行 ns-3 属性配置后，需要使用这些属性进行网络的搭建，Run()函数负责获取这些属性搭建网络并启动仿真，在 Run()函数中，完成网络拓扑的创建、网络传输层 TCP/IP 协议族的安装、安装应用层程序、设置路由、设置数据跟踪、仿真的启动和停止。Run()调用的函数如表 4-7 所示：

表 4-7 仿真脚本 Run()函数内调用函数

Table 4-7 Calling functions in the simulation script Run() function

函数名	作用
SeedManage()	设置随机种子
Configure-phyMode-Defaults()	设置物理层属性
createNodesphymacmobility()	创建节点移动模型
Set_ExternallyDrivenSim_config()	驱动内核设置仿真步长、协议、节点个数
InstallInternetStack()	安装 internet 协议栈
InstallApplication()	安装应用层
SetupWaveMessages()	设置信道接入模型
ConfigureLogTracing()	跟踪文件设置
Runsimulation()	设置仿真开始与结束

网络搭建完成后，调用 Runsimulation()函数启动仿真，Runsimulation()内包括 Simulator::Run，外部驱动程序类 ExternallyDrivenSim 公有继承 DefaultSimulatorImpl，并重写了 Simulator::Run()。Simulator::Run()用于启动仿真。

4.2.2 仿真驱动内核程序

仿真脚本程序的启动函数 Simulator::Run ()，为了驱动设计的 ns-3 仿真脚本，设计了外部的仿真驱动程序驱动仿真脚本，外部驱动程序类 ExternallyDrivenSim 公有继承 DefaultSimulatorImpl 类，并在 ExternallyDrivenSim 类重写客户端和服务端应用程序函数，实现向客户端应用内写入内容并运行 Socket 传送至服务端应用，ExternallyDrivenSim 类内函数 Run()重写 Simulator::Run ()并作为仿真的入口，ExternallyDrivenSim 类的 UML 图如图 4-3 所示。

图4-3 外部驱动ExternallyDrivenSim类UML图

Figure 4-3. UML diagram of ExternallyDrivenSim class

类 ExternallyDrivenSim 内入口函数为 Run()。

Run() 函数：由默认 Simulator::Run() 重写，能够调用函数 Listen()、TransimitNotices()等，用于控制仿真流程，其流程如图 4-4 所示。

在整个联合仿真平台中，ns-3 作为从属软件，负责接收 Scilab 的命令和仿真结果通知，返回仿真结果。因此，ns-3 设置了负责接收外部应用通知的函数 Listen() 和向外部发送仿真通知函数 TransmitNotices()。

首先需要建立一块存储区域并进行初始化，用于存储接收到的 Scilab 命令通知。首先当判断事件列表 m_events 不为空与仿真停止标志位 m_stop 为假与同步周期点小于仿真结束时刻 m_TimeOfEnd，m_TimeLimit 含义详见 2.1.1 节。即：

$$m\_TimeLimit < m\_TimeOfEnd + 2*g\_SimulationStep() \quad (4\text{-}1)$$

上述条件成立才会进入循环体，否则输出无可执行事件，仿真程序终止运行。

事件列表执行循环体设计用于不断取出并执行事件列表中的事件，直到运行到仿真时间上限或要执行的事件在下一周期，具体的事件列表执行循环体设计为取出下一个事件 f_next，当判断 f_next 的执行时间 f_next.key.m_ts 大于仿真时间上限 m_TimeOfEnd 与同步周期点大于仿真时间上限同时成立时，即：

$$f\_next.key.m\_ts \geq m\_TimeOfEnd + g\_SimulationStep \quad (4\text{-}2)$$

$$m\_TimeLimit \geq m\_TimeOfEnd + g\_SimulationStep \quad (4\text{-}3)$$

式（4-2）（4-3）成立，判断仿真应该结束，运行 TransmitNotices()向 Scilab 传送仿真结果和 Stop()结束仿真。否则判断判断 f_next 的执行时间大于 m_TimeLimit，即：

$$f\_next.key.m\_ts >= m\_TimeLimit \quad (4\text{-}4)$$

式（4-4）成立，f_next 应在下一个仿真周期执行，此时根据同步周期机制设计，即 2.1.1 节，应该执行与 Scilab 的同步，这一步操作需要首先判断仿真暂停标志位 m_quit 为假，执行 TransmitNotices()发送通知，Listen()接收 Scilab 在此同步时刻的通知，如果 m_quit 为真，意味着仿真结束，则将仿真结果执行 TransmitNotices()发送，运行 Stop()结束仿真。

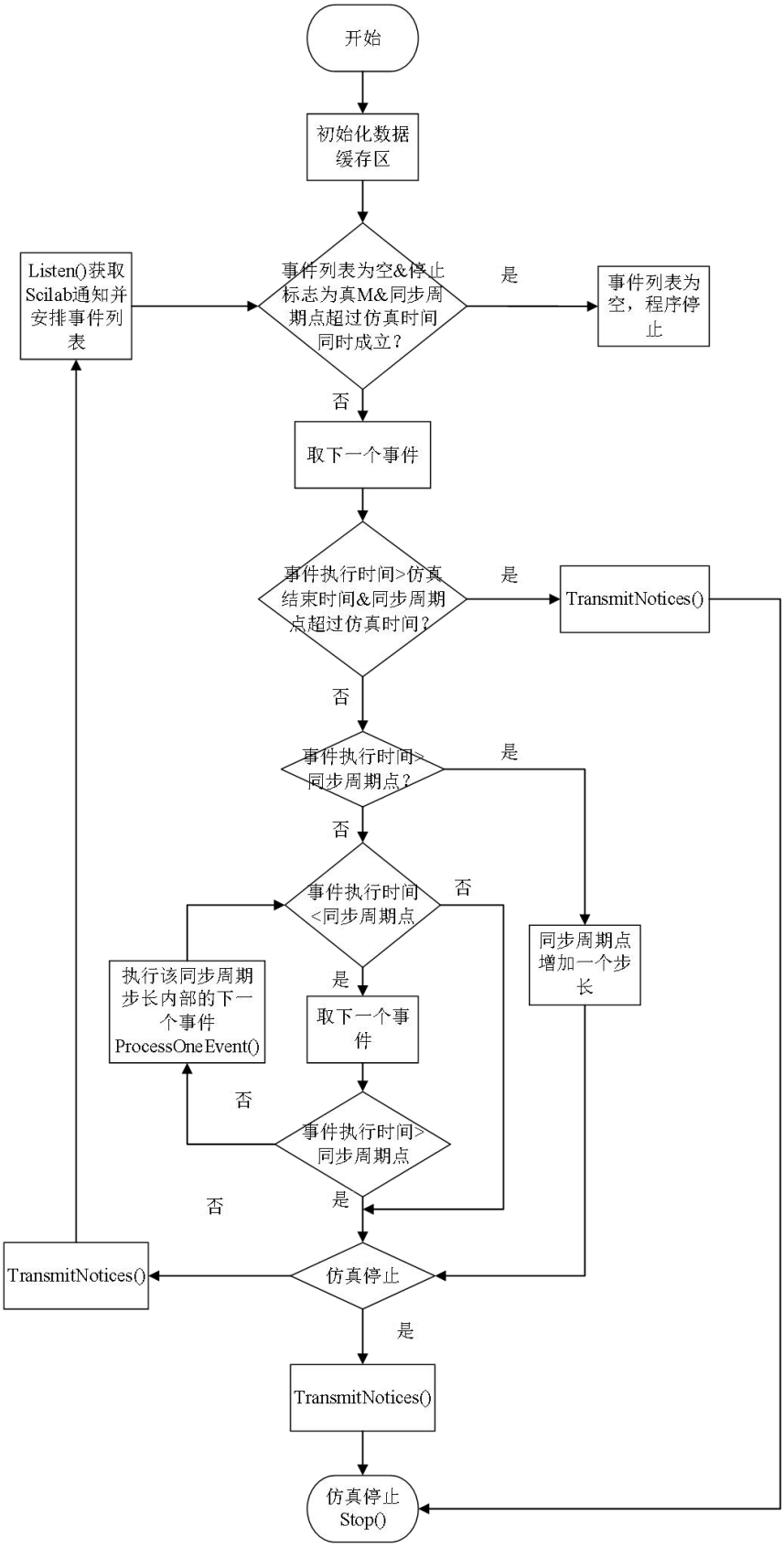


图4-4 ExternallyDrivenSim类中Run()流程图

Figure 4-4. Flow chart of Run() in Class ExternallyDrivenSim

当判断 f_next 的执行时间小于 m_TimeLimit，即：

$$f\_next.key.m\_ts < m\_TimeLimit \quad (4\text{-}5)$$

式（4-5）成立，意味着下一个事件还在当前仿真周期内，进入取事件运行循环体，进入循环体后首先设置跳出循环体的判断条件，即：

$$f\_next.key.m\_ts >= m\_TimeLimit \quad (4\text{-}6)$$

式（4-6）成立，表示下一事件 f_next 在下一周期，跳出循环体，否则继续执行循环体，执行 ProcessOneEvent()函数，ProcessOneEvent()函数用于从事件列表 m_events 取出执行事件，并将此事件在进行 m_events 移除，将该事件执行。

当跳出循环结构体时，说明 f_next 的执行时间大于 m_TimeLimit，执行同步操作。

4.2.3 ns-3 仿真应用程序实现

ns-3 网络框架中应用层的安装主要分为 TCP 应用和 UDP 应用，重新设计这些应用类以实现接收外部指令，如设置发送节点和目标节点，发送的内容，获取发送节点的发送时间 SendTime 和接收节点的接收时间 RecvTime，这些重写的应用类包括 UdpClient 客户端应用类如图 4-5 所示，Udp 服务端应用类如图 4-6 所示，TcpClient 客户端应用类如图 4-7 所示，TcpServer 服务端应用类如图 4-8 所示以及对应的安装助手类，根据仿真的内容需求对发送节点和接收节点安装选择对应协议的客户端应用程序类和服务端应用程序类。

图4-5 Udp客户端应用类UML图

Figure 4-5. UML diagram of UDP client application class

图4-6 Udp服务端应用类

Figure 4-6. UML diagram of UDP server application class

图4-7 Tcp客户端应用类UML图

Figure 4-7. UML diagram of TCP client application class

图4-8 Tcp服务端应用类UML图

Figure 4-8. UML diagram of TCP server application class

应用安装助手类包括 UDP 分组产生器和 TCP 分组产生器，应用安装助手类设计是为了屏蔽应用安装复杂的操作细节，方便用户使用，应用安装助手类 UML 图如图 4-9 所示。

图4-9 应用安装助手类UML图

Figure 4-9. UML diagram of application helper class

4.2.4 ns-3 中 C++类的调用关系

新建 ns-3 仿真脚本，在脚本中定义 ns-3 仿真模块，定义初始化函数：节点设置函数、网络拓扑建立函数、信道属性配置函数、网络设备创建函数、网络协议栈安装函数、仿真开始及结束函数，通过 ns-3 协同仿真接口向以上初始化函数传值完成 ns-3 配置；启动脚本程序 ExternallyTest，ns-3 内 C++类的调用关系如图 4-10 所示。

本仿真平台编写的 ns-3 仿真脚本内包含仿真脚本类，在仿真脚本类 ExternallyTest 实例化一个对象，并通过对象调用 configure()进行网络属性的配置，之后对象调用 Run()函数搭建网络，在网络搭建过程中会创建节点 node 并为该节点同时安装客户端和应用端程序，在 ns-3 仿真运行时可以通过设置节点的索引指向服务端程序和应用端程序，控制该节点是服务器节点还是客户端节点。

为节点安装完应用程序后，脚本类函数 Run()调用启动 ns-3 默认仿真函数 Simulator::Run()，因为 Simulator::Run()是仿真驱动类继承并重写过的，所以实际运行的是仿真驱动类 ExternallyDrivenSim 类的类内函数 Run()，启动运行后会启动 ns-3 联合仿真接口程序，接收 Scilab 的事件通知，并将相应的节点设置为发送节点或接收节点。

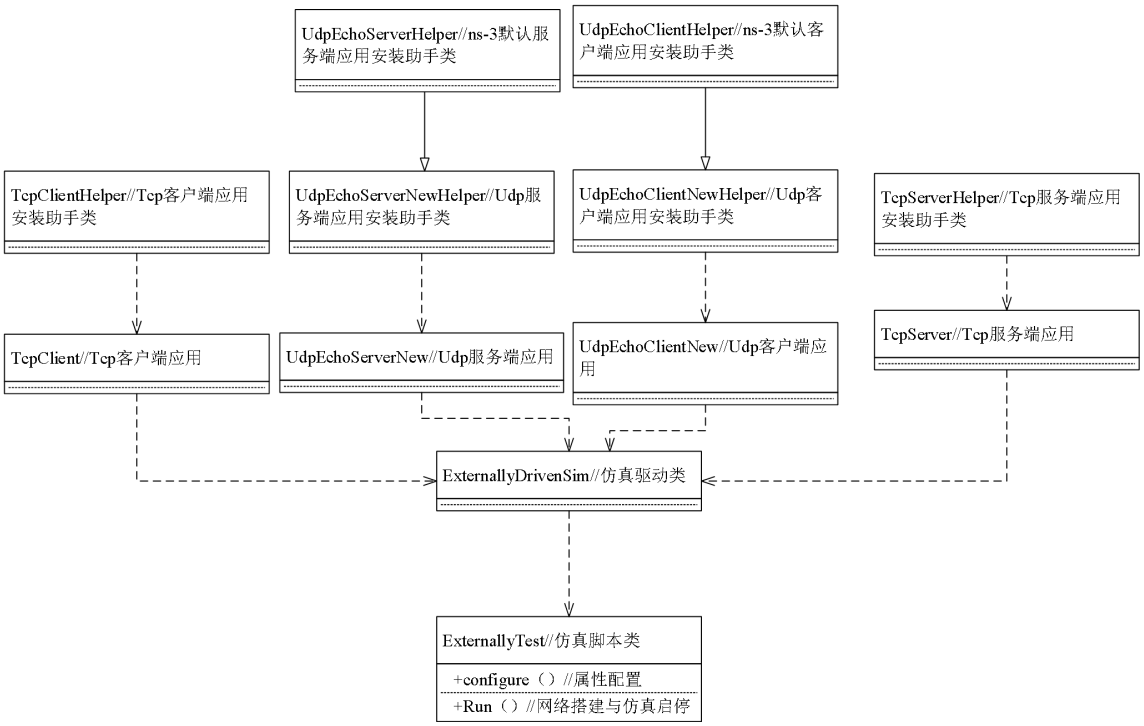


图4-10 ns-3内C++类的调用关系图

Figure 4-10. Calling diagram of C++ classes in ns-3

ns-3 内仿真网络是通过为节点安装客户端程序使该节点成为发送节点，为节点安装客户端程序使该节点成为服务器节点，如果事件列表中包含一个发送事件send()；会找到该事件执行的节点，激活该节点的客户端属性，填充需要发送的内容，记录节点发送的时间 SendTime。为目标节点激活服务端属性，记录接受的时间 RecvTime，检测丢包并根据公式（4-7）求得延时 Delay。

$$Delay=RecvTime-SendTime \quad (4\text{-}7)$$

求得延时之后仿真驱动类 ExternallyDrivenSim 类内函数 GetNotices()获取接收节点的丢包和延时结果，并将结果存储在结构体中，根据联合仿真同步机制是否运行 ExternallyDrivenSim 类内函数 TransmitNotice()选择是否传递给 Scilab，函数 TransmitNotice 的调用可参考 4.2.2 节及图 4-4。

4.3 本章小结

本章中主要介绍了 Scilab 内部程序实现，包括 Xcos 中控制系统模型的搭建，Xcos 驱动程序的实现流程以及 Scilab 中联合仿真平台用户交互 GUI 界面的设计，ns-3 软件仿真脚本的设计和驱动内核程序的设计以及部分类的实现，应用类及助手类的 UML 图，给出了类的关系和类内函数的调用次序，画出了联合仿真平台在仿真时 Scilab 和 ns-3 的内部仿真流程图。

第五章 仿真平台实验验证

联合仿真平台搭建完成后需要通过实验验证仿真平台的有效性和可靠性，本章将通过两个仿真实例验证仿真平台的性能。

5.1 联合仿真平台有效性验证实验设计

为了检验本文联合仿真平台的基本功能，本节设计了一个稳定的控制系统仿真实例，通过观察仿真平台的输出判断仿真平台是否正常运行。

5.1.1 仿真平台基本功能验证

本节设计了一个稳定控制系统实验方案，搭建如图 3-7 所示的 Xcos 控制系统模型，选用一个稳定的控制系统，测试联合仿真平台是否能够达到稳定以及输出延时信息，控制器选用模型为 y=Au，其中控制器模型选用 A=[0,0]，u 为输入，y 为输出。进行功能性验证，执行器会根据数据包的时间戳判断数据包的发送时间并选择最新的数据包，被控对象选用模型为：

$$A = \begin{bmatrix} 0.9 & 0 \\ 0 & 0.8 \end{bmatrix} \tag{5-1}$$

$$B = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} \tag{5-2}$$

二阶系统模型如下：

$$x(k+1) = Ax(k) + Bu(k) \tag{5-3}$$

通过联合仿真 GUI 界面打开并编辑被控对象模块和控制器模块，将上述模型输入，之后开始仿真，设置 ns-3 通信网络标准为 802.11b 无线局域网。联合仿真运行如图 5-1 所示，仿真完成后得到传感输出如图 5-2 所示，ns-3 仿真过程网络拓扑和节点间数据分组流动画演示截图如图 5-3 所示，ns-3 仿真过程中控制器节点接收数据的打印信息如图 5-4 所示。

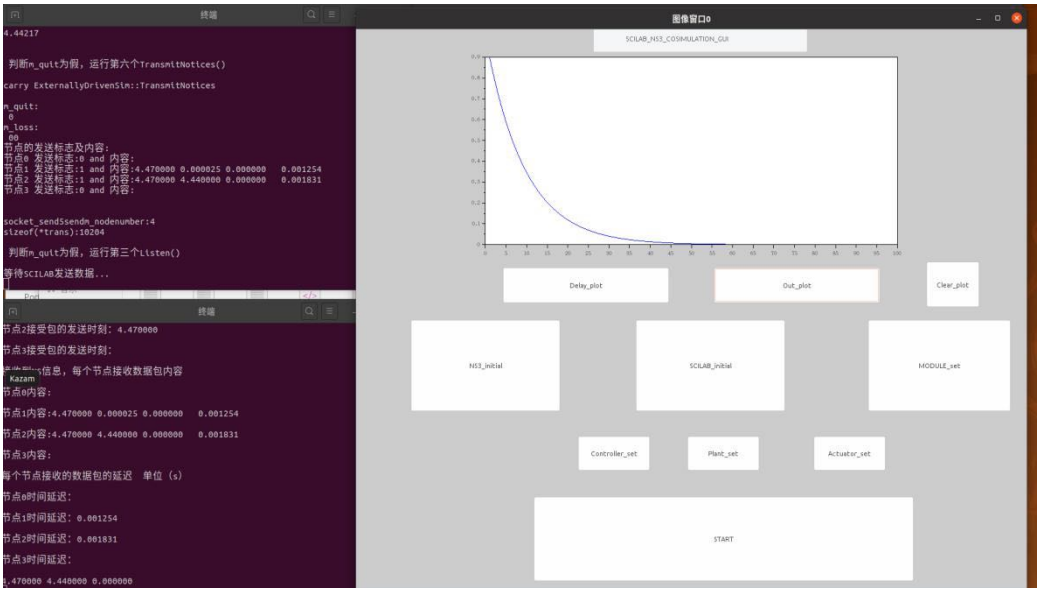


图5-1 联合仿真平台运行界面图

Figure 5-1. Co-simulation platform operation diagram

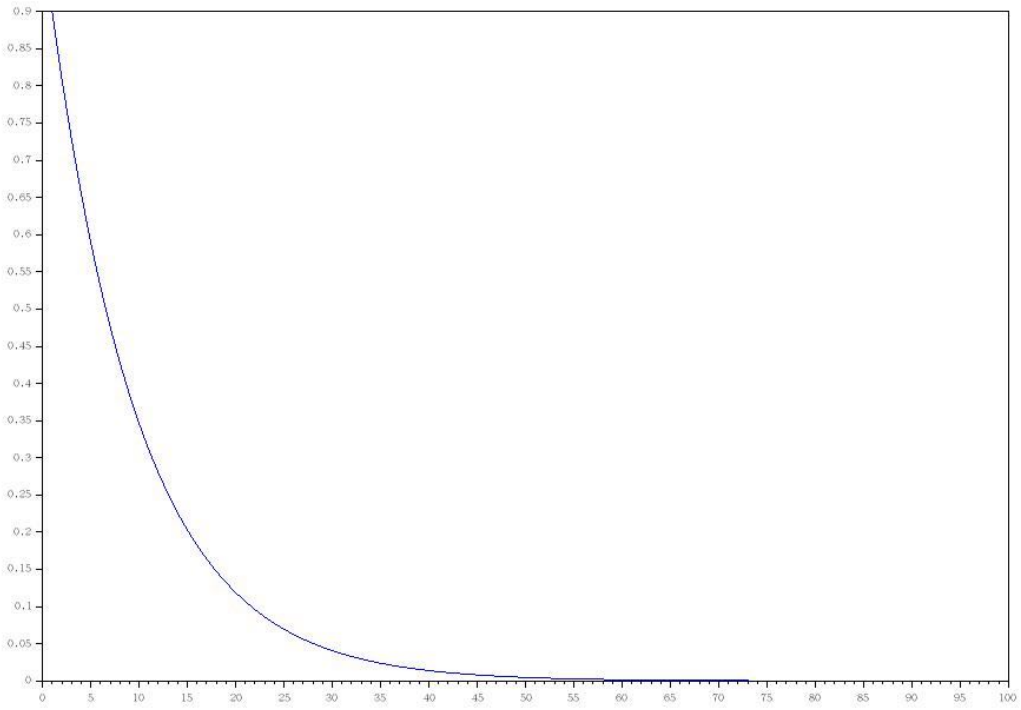


图5-2 仿真结果图

Figure 5-2. Simulation result graph

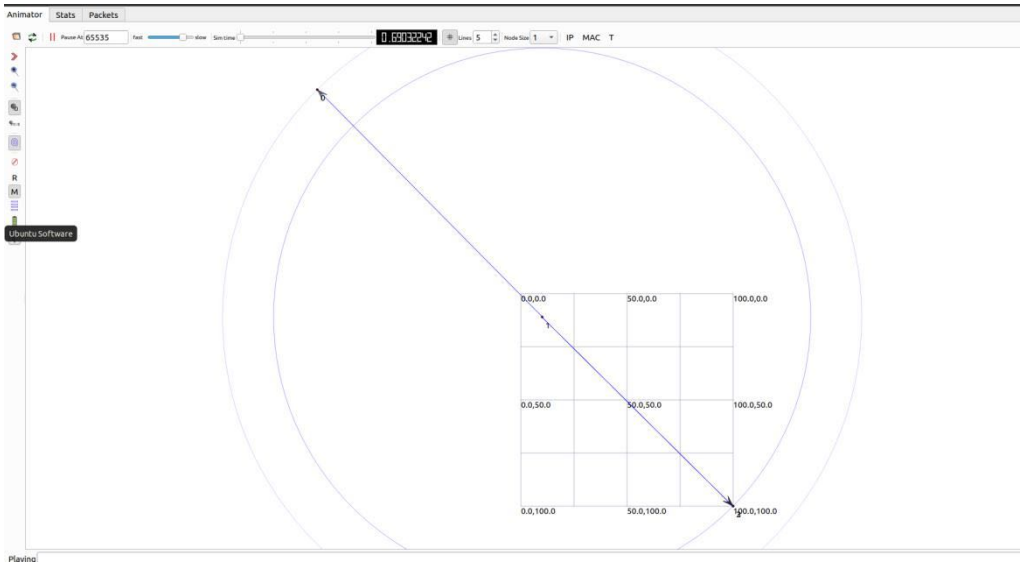


图5-3 ns-3仿真过程网络拓扑和节点间数据分组流动画演示截图

Figure 5-3. Screenshot of animation demonstration of network topology and data packet flow between nodes during ns-3 simulation

```
4.081124 Acknowledgment RA:00:00:00:00:00:04
4.082230 IP 6.6.6.1.49153 > 6.6.6.3.9: UDP, length 18
4.082444 Acknowledgment RA:00:00:00:00:00:01
4.111194 IP 6.6.6.4.49153 > 6.6.6.2.9: UDP, length 18
4.111204 Acknowledgment RA:00:00:00:00:00:04
4.111830 IP 6.6.6.1.49153 > 6.6.6.3.9: UDP, length 18
4.112044 Acknowledgment RA:00:00:00:00:00:01
4.141014 IP 6.6.6.4.49153 > 6.6.6.2.9: UDP, length 18
4.141024 Acknowledgment RA:00:00:00:00:00:04
4.142310 IP 6.6.6.1.49153 > 6.6.6.3.9: UDP, length 18
4.142524 Acknowledgment RA:00:00:00:00:00:01
4.171334 IP 6.6.6.4.49153 > 6.6.6.2.9: UDP, length 18
4.171344 Acknowledgment RA:00:00:00:00:00:04
4.172250 IP 6.6.6.1.49153 > 6.6.6.3.9: UDP, length 18
4.172464 Acknowledgment RA:00:00:00:00:00:01
4.201634 IP 6.6.6.4.49153 > 6.6.6.2.9: UDP, length 18
4.201644 Acknowledgment RA:00:00:00:00:00:04
4.202670 IP 6.6.6.1.49153 > 6.6.6.3.9: UDP, length 18
4.202884 Acknowledgment RA:00:00:00:00:00:01
4.231554 IP 6.6.6.1.49153 > 6.6.6.3.9: UDP, length 18
4.231768 Acknowledgment RA:00:00:00:00:00:01
4.232390 IP 6.6.6.4.49153 > 6.6.6.2.9: UDP, length 18
4.232400 Acknowledgment RA:00:00:00:00:00:04
4.261054 IP 6.6.6.1.49153 > 6.6.6.3.9: UDP, length 18
4.261268 Acknowledgment RA:00:00:00:00:00:01
4.262050 IP 6.6.6.4.49153 > 6.6.6.2.9: UDP, length 18
4.262060 Acknowledgment RA:00:00:00:00:00:04
4.291074 IP 6.6.6.1.49153 > 6.6.6.3.9: UDP, length 18
4.291288 Acknowledgment RA:00:00:00:00:00:01
4.291750 IP 6.6.6.4.49153 > 6.6.6.2.9: UDP, length 18
4.291760 Acknowledgment RA:00:00:00:00:00:04
4.321174 IP 6.6.6.1.49153 > 6.6.6.3.9: UDP, length 18
4.321388 Acknowledgment RA:00:00:00:00:00:01
4.321810 IP 6.6.6.4.49153 > 6.6.6.2.9: UDP, length 18
4.321820 Acknowledgment RA:00:00:00:00:00:04
4.351314 IP 6.6.6.4.49153 > 6.6.6.2.9: UDP, length 18
4.351324 Acknowledgment RA:00:00:00:00:00:04
4.352490 IP 6.6.6.1.49153 > 6.6.6.3.9: UDP, length 18
4.352704 Acknowledgment RA:00:00:00:00:00:01
4.381614 IP 6.6.6.1.49153 > 6.6.6.3.9: UDP, length 18
4.381828 Acknowledgment RA:00:00:00:00:00:01
4.382250 IP 6.6.6.4.49153 > 6.6.6.2.9: UDP, length 18
4.382260 Acknowledgment RA:00:00:00:00:00:04
4.411794 IP 6.6.6.1.49153 > 6.6.6.3.9: UDP, length 18
4.412008 Acknowledgment RA:00:00:00:00:00:01
4.412350 IP 6.6.6.4.49153 > 6.6.6.2.9: UDP, length 18
4.412360 Acknowledgment RA:00:00:00:00:00:04
4.441194 IP 6.6.6.4.49153 > 6.6.6.2.9: UDP, length 18
4.441204 Acknowledgment RA:00:00:00:00:00:04
4.442430 IP 6.6.6.1.49153 > 6.6.6.3.9: UDP, length 18
4.442644 Acknowledgment RA:00:00:00:00:00:01
4.471254 IP 6.6.6.4.49153 > 6.6.6.2.9: UDP, length 18
4.471264 Acknowledgment RA:00:00:00:00:00:04
4.471830 IP 6.6.6.1.49153 > 6.6.6.3.9: UDP, length 18
4.472044 Acknowledgment RA:00:00:00:00:00:01
```

图5-4 ns-3仿真中控制器节点接收数据信息

Figure 5-4. The controller node receives data information in ns-3 simulation

5.1.2 仿真结果分析

由于输入系统是一个稳定系统，在系统比较简单的情况下没有丢包信息，系统输出最后会达到稳定状态。实验结果由上图可知系统的输出逐渐接近为 0，控制系统的输出达到稳定状态，与预期相同，可以初步判定联合仿真平台的基本功能正常。

5.2 基于分类控制的具有有损多包传输的无线网络控制系统仿真

在验证联合仿真平台能够正常工作后，需要验证仿真平台面对复杂仿真环境的可靠性，本节使用一个复杂的控制系统实例来验证仿真平台的可靠性。

5.2.1 系统描述

针对具有有损多包传输的无线网络控制系统，有学者提出了一种基于分类的控制方法[54]。这种方法利用状态重建过程来处理数据部分故障的显着特征由多包传输引起的传输，然后对最新接收到的系统状态的差异进行分类和重建的那些来设计一个基于分类的控制器。系统的闭环稳定性被证明使用切换系统论。通过考虑多包传输的更多通信特性。

考虑多包传输的网络化控制系统如图 5-5 所示，其中 p 个独立传感器对被控对象进行采样，并通过具有竞争性和不完善的通信网络发送传感数据，可能会丢失数据，而从控制器到执行器的通信通道被假定为无损。这样的系统设置实际上是合理的，因为传输传感数据的“不完善”主要是由于通道的接入竞争造成的，而没有传输控制数据。 关于这个系统设置的更多讨论，可以参考文献[55]。

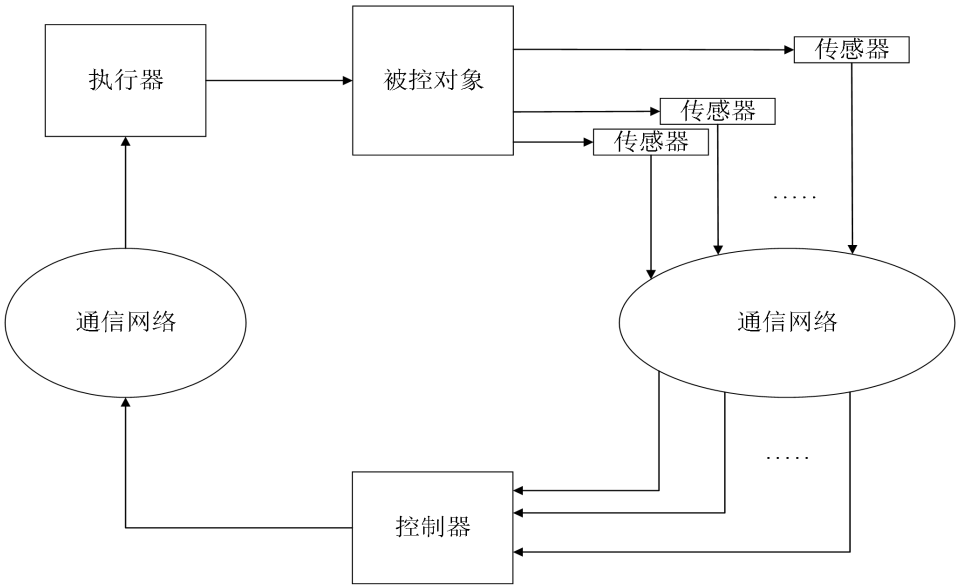


图 5-5 多包传输的网络化控制系统框图

Figure 5-5. Block diagram of networked control system for multi-packet transmission

被控对象模型可由以下线性离散时间不变系统描述为式（5-4）。

$$x(k+1)=Ax(k)+Bu(k) \tag{5-4}$$

其中，$x\in R^n, u\in R^n, A\in R^{n\times n}, B\in R^{n\times m}$。

我们可以假设系统状态 x(k)是通过顺序放置感知的数据，即 $x(k)=[(x^1(k))^T,(x^2(k))^T,\ldots,(x^p(k))^T]^T$，其中$x^{\mathrm{i}}(k)\in R^{c_i}\ ,i=1,2,\ldots,p$，是$k$时刻传感器 i 的传感量，$\sum_{i=1}^{p}c_i$=n。在传输中$x^{\mathrm{i}}(k)$可能发生丢失，可以建模为$\alpha_k^i$，如式（5-5）所示。

$$\alpha_k^i=\begin{cases}0 & x^i(k)\text{发生丢失}\\ 1 & \text{其他情况}\end{cases} \tag{5-5}$$

将指标向量定义为$\left[\alpha_k^1,\ \alpha_k^2,\ldots,\alpha_k^p\right]^T\in I$ 其中$I=\{[\sigma_1,\ldots,\sigma_i,\ldots,\sigma_p]^T:\sigma_1=0,1,i=1,\ldots,p\}$，该指标向量明确指定数据是否来自传感器$i$，$i=1,\ldots,$p 传输是否成功。

多包传输对系统设计的影响对于在网络化控制系统中具有有损单包传输的控制系统，常规的状态反馈规律通常如下获得：

$$u(k)=K\tilde{x}(k) \tag{5-6}$$

其中反馈增K是时不变的。当$x(k)$由于数据包丢失或延迟而不可用时，我们可以使用上一步的值[56]：

$$\tilde{x}(k)=\begin{cases}x(k) & x(k)\text{可用}\\ \tilde{x}(k-1) & \text{其他情况}\end{cases} \tag{5-7}$$

或式（5-8）值为 0 的情况。

$$\tilde{x}(k)=\begin{cases}x(k) & x(k)\text{可用}\\ 0 & \text{其他情况}\end{cases} \tag{5-8}$$

上述两种策略通常适用于传统的单包传输场景。然而在存在有损多包传输的情况下，面临着完全不同的情况，即$x(k)$可能在时间k对控制器不完全可用，但由于部分成功传输，它的某些部分可以使用。这样的事实意味着，上述对数据包丢失的传统处理可能过于保守，遗憾的是忽略了控制器可用的许多信息。

基于上述问题提出了具有多包传输的网络化控制系统的基于分类的控制方法，基于分类的控制方法包括三个模块：网络状态检测器、传感数据重建模块和基于分类的控制器，如图 5-6 所示。数据包丢失首先由网络检测，状态检测器在（5-6）中指定，然后重建系统状态。

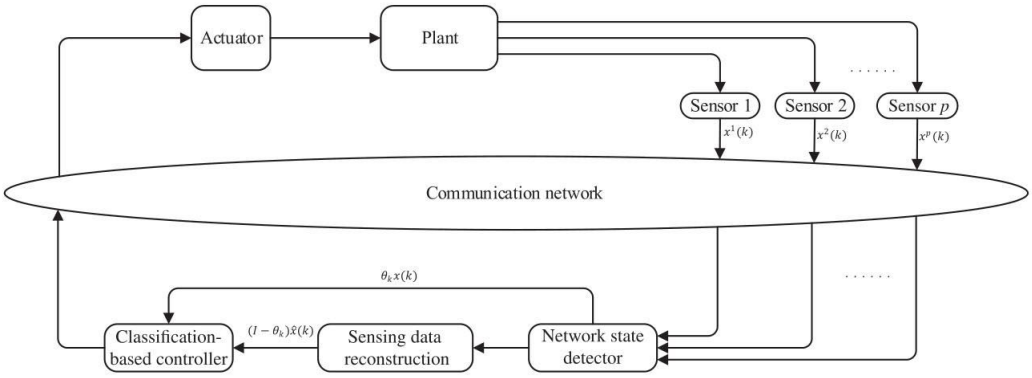


图5-6 多包传输的NCS分类控制律示意图

Figure 5-6. Schematic diagram of NCS classification control law for multi-packet transmission

通过系统状态重建模块[57]，最后设计了基于分类的控制器。对于系统状态重建过程，将（5-4）中的系统矩阵A和输入矩阵B拆分为 p×p 和 p×1 块矩阵。

$$A=\begin{bmatrix} A^{11} & A^{12} & \cdots & A^{1p} \\ A^{21} & A^{22} & & A^{2p} \\ \vdots & & \ddots & \vdots \\ A^{p1} & A^{p2} & \cdots & A^{pp} \end{bmatrix},\ B=\begin{bmatrix} B^1 \\ B^2 \\ \vdots \\ B^p \end{bmatrix} \tag{5-9}$$

其中$A^{ij}\in R^{c_i\times c_j}, i=1,2,\ldots,p, j=1,2,\ldots,p$,以及 $B^i\in R^{c_i\times m}$，则：

$$x^i(k+1)=\sum_{j=1}^{p}A^{ij}x^j(k)+B^iu(k) \tag{5-10}$$

其中$i=1,2,\ldots,p$。

假设$x^j(k)$对于某些 j，控制器可能不能使用。$x^j(k)$表示传感器 i 在时间 k+1 重建。则：

$$\hat{x}^i(k+1)=\sum_{j=1}^{p}A^{ij}\bar{x}^j(k)+B^iu(k) \tag{5-11}$$

其中$\bar{x}^j(k)=\alpha_k^j x^j(k)+$（$1-\alpha_k^j$）$\hat{x}^j(k)$。

定义$\bar{x}^T(k)=[\{\bar{x}^1(k)\}^T,\{\bar{x}^2(k)\}^T,\ldots\{\bar{x}^p(k)\}^T]$那么随之而来的是：

$$\bar{x}(k)=\Theta_{\sigma(k)}x(k)+(I-\Theta_{\sigma(k)})\hat{x}(k) \tag{5-12}$$

其中$\sigma(k)\in I$，以及

$$\Theta_{\sigma(k)}=\begin{bmatrix} \alpha_k^1 I^{c_1\times c_1} & & & \\ & \alpha_k^2 I^{c_2\times c_2} & & \\ & & \ddots & \\ & & & \alpha_k^p I^{c_p\times c_p} \end{bmatrix} \tag{5-13}$$

因此，(5-11)可以写成紧凑的形式：

$$\hat{x}(k+1)=A\bar{x}(k)+Bu(k) \tag{5-14}$$

由于实时性要求，控制系统通常只使用最新的系统状态和控制信号信息。上述

基于模型的补偿方案可以重建由多包传输引起的部分丢失信息，但是基于旧的感知数据，因此不如新到达的数据好。实时控制器需要澄清这些不同的信号，并且只使用最新的信息，从而产生所谓的基于澄清的控制律，如下所示：

$$u(k) = K_1\Theta_{\sigma(k)}x(k) + K_2(I - \Theta_{\sigma(k)})\hat{x}(k) \quad (5\text{-}15)$$

其中 K_1，K_2 $\in R^{n\times m}$ 是真实和重构系统状态的不同控制增益。

闭环系统通过定义$\eta^T(k) = [x^T(k), \hat{x}^T(k)]$，从（5-4）、（5-12）、（5-14）和（5-15），闭环网络化控制系统转换为：

$$\eta(k+1) = \Phi_{\sigma(k)}\eta(k) \quad (5\text{-}16)$$

其中$\Phi_{\sigma(k)}$如式（5-17）所示。

$$\Phi_{\sigma(k)} = \begin{bmatrix} A + BK_1\Theta_{\sigma(k)} & BK_2(I - \Theta_{\sigma(k)}) \\ (A + BK_1)\Theta_{\sigma(k)} & (A + BK_2)(I - \Theta_{\sigma(k)}) \end{bmatrix} \quad (5\text{-}17)$$

5.2.2 基于 Simulink 仿真

考虑以下有扰动的开环不稳定三阶系统：

$$x(k+1) = Ax(k) + Bu(k) + w(k) \quad (5\text{-}18)$$

其中初始状态为$x(0) = [-1, -1, 1]^T$，$w(k)$为方差为 0.05 的高斯白噪声，系统矩阵如下所示[58]。

$$A = \begin{bmatrix} -0.85 & 0.271 & -0.488 \\ 0.482 & 0.100 & 0.2400 \\ 0.002 & 0.3681 & 0.7070 \end{bmatrix} \quad (5\text{-}19)$$

$$B = \begin{bmatrix} 0.5 & 0.1 \\ 0.3 & -0.4 \\ 0.2 & 0.5 \end{bmatrix} \quad (5\text{-}20)$$

在仿真中，假设（5-18）中的 3 个系统状态由三个独立的传感器采样和发送，具有相同的数据丢失率为 0.35。使用本文提出的方法，可以获得以下增益矩阵：

$$K_1 = \begin{bmatrix} 0.8445 & -0.5877 & 0.1826 \\ 0.5410 & -0.4863 & -0.7082 \end{bmatrix} \quad (5\text{-}21)$$

$$K_2 = \begin{bmatrix} 0.1036 & -0.1547 & 0.0272 \\ 0.1154 & -0.0185 & -0.1766 \end{bmatrix} \quad (5\text{-}22)$$

Simulink 仿真结果如图 5-7 可以看出，方法保证了闭环系统的稳定性，最新接收的系统状态和重建的系统状态被明确地分类为不同的控制增益。

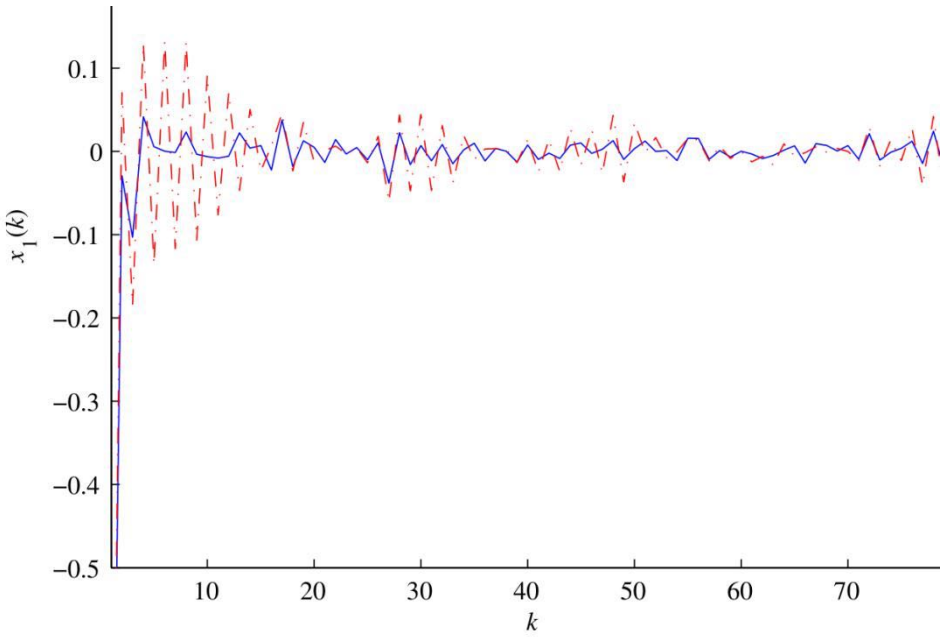


图5-7 基于分类控制统状态轨迹图

Figure 5-7. Trajectory diagram based on classification control system state

5.2.3 基于联合仿真平台仿真

将 5.2.2 节中控制系统的设计，使用本文联合仿真平台进行实验，同时使用了文献 5.2.2 节相同的参数。由于 5.2.2 节中是三阶系统，所以对联合仿真平台的 Xcos 模型进行了修改，修改后的控制系统 Xcos 建模如图 5-8 所示：

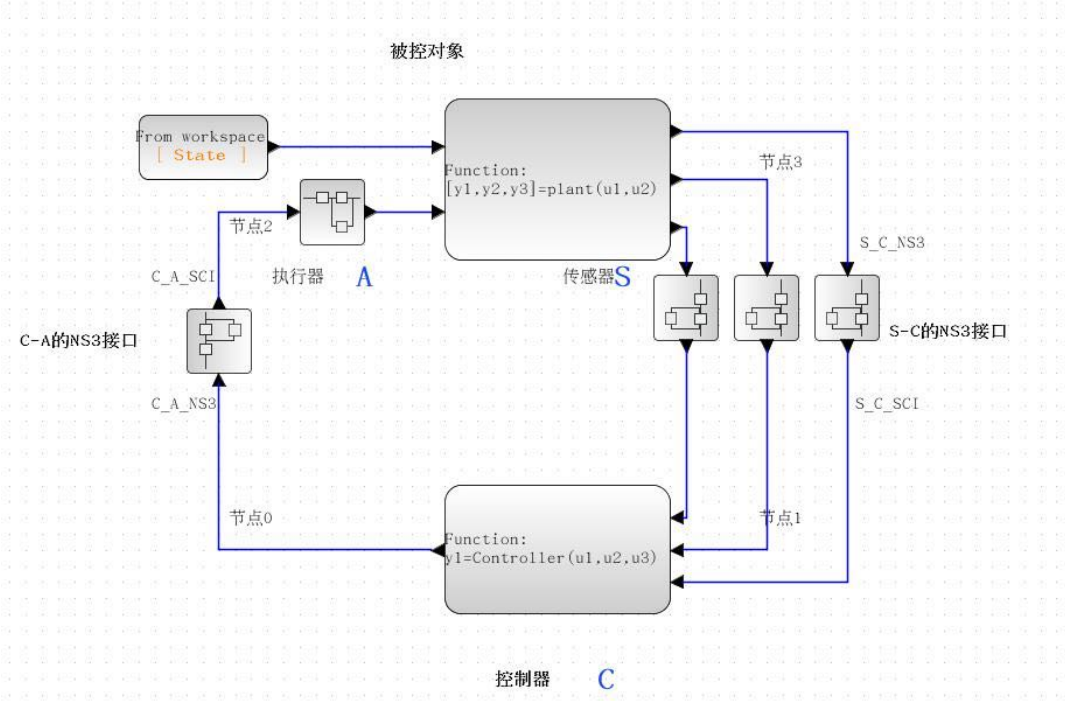


图5-8 多包传输Xcos控制系统模型

Figure 5-8. Multi-packet transmission Xcos control system model

得到实验结果如图 5-9 所示，仿真延时分布图如图 5-10 所示。

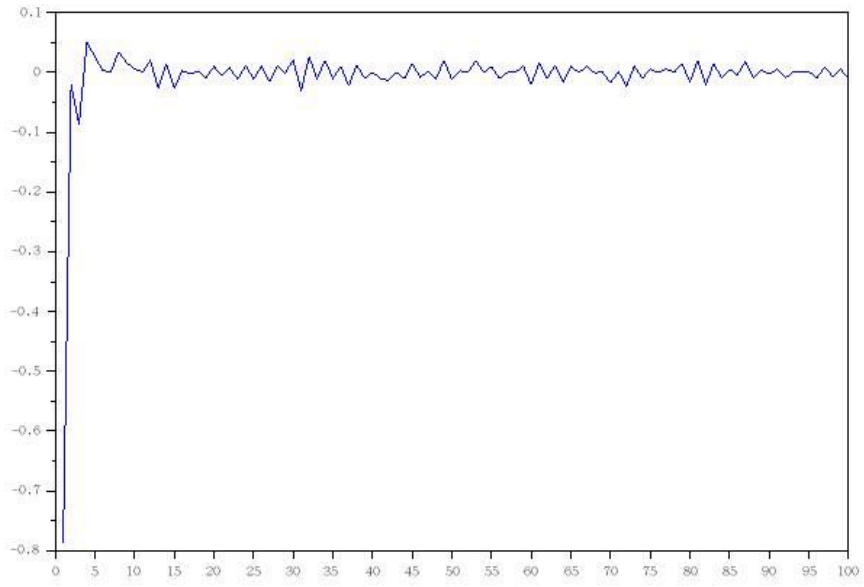


图5-9 基于联合仿真平台仿真系统状态轨迹图

Figure5-9．State trajectory diagram of simulation system based on co-simulation platform

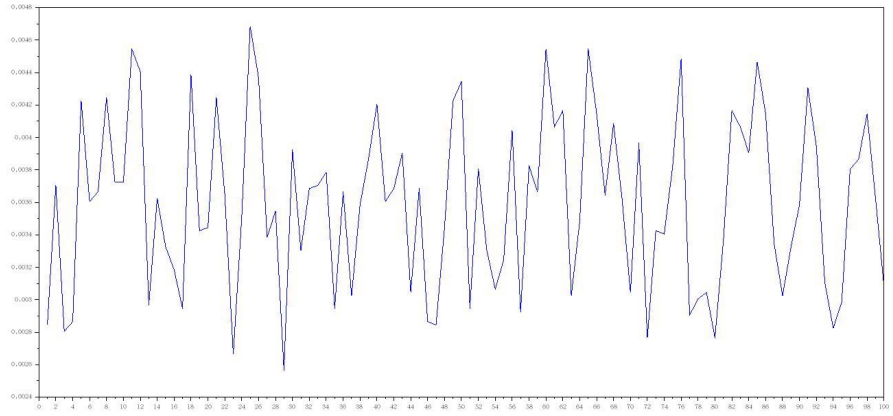


图5-10 基于联合仿真平台的传输时延分布图

Figure 5-10. Transmission delay distribution diagram based on the joint simulation platform

5.3 实验结果对比分析

根据图 5-7 中 Simulink 仿真和图 5-9 对比可看出在两个仿真实例中，采用该平台可有效的仿真各种网络化控制系统，仿真结果与使用 Simulink 工具箱仿真结果基本一致，相比于使用 Simulink 工具箱能更加真实的仿真多种网络环境下的控制

系统的运行情况，支持上层网络协议和节点移动模型，同时丰富了网络拓扑的搭建。本文联合仿真平台还支持根据打印的网络仿真辅助信息查看传输情况，结合辅助信息对网络化控制系统的性能进行分析和改善。

5.4 本章小结

本章首先通过一个稳定的控制系统进行仿真平台功能性验证，之后通过一个多传感器基于分类控制的具有有损多包传输的无线网络控制系统仿真实例验证了仿真平台的稳定性，和基于 Simulink 工具箱的实验相比，仿真结果基本一致，验证了本文联合仿真平台面对复杂系统的可靠性。

第六章 总结与展望

6.1 总结

本文通过分析网络化控制系统仿真工具的研究现状，设计并搭建了基于开源免费软件的网络化控制系统联合仿真平台，对推动我国网络化控制系统仿真技术的发展进步有着积极的作用。本文主要完成的工作如下：

(1) 设计了一种联合仿真平台总体架构和用于 Scilab 和 ns-3 软件间通信的联合仿真接口程序。总体架构确定了联合仿真平台各个模块功能、模块之间的通信方式，联合仿真平台架构还规定了各功能模块运行的次序。接口程序能够统一两个软件的数据格式，实现 Scilab 和 ns-3 仿真结果的交换与解析。

(2) 设计了用于同步 Scilab 和 ns-3 仿真时间的主从式周期同步机制。本文根据两软件驱动方式的不同以及联合仿真对时间同步的需求，设计了一种主从式周期同步机制用于协调 Scilab 和 ns-3 联合仿真的次序并同步两软件的仿真时间，能够保证联合仿真平台的正常运行。

(3) 通过使用联合仿真平台仿真给定稳定控制系统，验证联合仿真平台的可用性。之后通过有损多包传输的无线网络控制系统实验验证了联合仿真平台对多包传输的控制系统有效性。对比与其他仿真方式的仿真结果验证了联合仿真平台的可靠性。

最后通过两个仿真实例验证了仿真平台的可用性、稳定性和创新性，在对比试验中和基于 Simulink 工具箱的实验相比，仿真结果基本一致。本仿真平台还具有仿真模块组合灵活、开源免费、可二次分发、易于推广等优点。本文联合仿真平台代码已开源并托管至 gitee，项目代码地址：https://gitee.com/lu-shuailing/network-control-system-simulation-platform.git。

6.2 展望

（1）本文联合仿真平台的使用上有一定的门槛，模块的组合灵活性有一定的折扣，如果使用者修改 Xcos 模型，需要使用者有一定的 Scilab 模块编程基础。

（2）可以设计更好的用户软件使用界面，实现网络设置图形化和控制系统设计图形化，实现平台的 Web 端部署，使用户更容易使用此仿真平台。

参考文献

[1] Wang F Y , Liu D. Networked Control Systems [J]. Lecture Notes in Control & Information Sciences, 2008,52(9): 318–323.

[2] Zhang X M, Han Q L, Yu X. Survey on recent advances in networked control systems[J]. IEEE Transactions on Industrial Informatics, 2016, 12(5): 1740-1752.

[3] Farias C D, Soares H, Pirmez L, et al. A control and decision system for smart buildings using wireless sensor and actuator networks[J]. European Transactions on Telecommunications, 2014, 25(1): 120-135.

[4] Zhang D, Nguang S K, Yu L. Distributed control of large-scale networked control systems with communication constraints and topology switching[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017, 47(7): 1746-1757.

[5] Wu Y, Karimi H R, Lu R. Sampled-data control of network systems in industrial manufacturing[J]. IEEE Transactions on Industrial Electronics, 2018, 65(11): 9016-9024.

[6] Yang T C. Networked control system: a brief survey[J]. IEE Proceedings-Control Theory and Applications, 2006, 153(4): 403-412.

[7] Al-Sheikh H, Moubayed N. An overview of simulation tools for renewable applications in power systems[C]// Advances in Computational Tools for Engineering Applications (ACTEA). 2012 2nd International Conference. IEEE, 2012: 257-261.

[8] 王书廷. 卫星及气浮台质量特性的在线辨识算法研究[D]. 哈尔滨. 哈尔滨工业大学, 2006.

[9] 许剑, 杨庆俊, 包钢, 等. 多自由度气浮仿真试验台的研究与发展[J]. 航天控制, 2009 (6): 96-101.

[10] 李季苏, 牟小刚, 张锦江. 卫星控制系统全物理仿真[J]. 航天控制, 2004, 22(2): 37-41.

[11] 余波, 朱纪洪, 范勇, 等. 飞行控制系统大闭环半物理仿真研究[J]. 航天控制, 2010，28(2): 51-55.

[12] 谢素春. II 型车控制网络半实物仿真平台应用研究[D]. 成都. 西南交通大学, 2013.

[13] 冯江华, 王坚, 李江红. 高速列车牵引传动系统综合仿真平台的分析与设计[J]. 铁道学报, 2012, 34(2): 21-26.

[14] Al-Hammouri A T. Co-simulation tools [J]. Computer Communications, 2012, 36(1): 8-19.

[15] Tsang Y, Coates M, Nowak R D. Network delay tomography[J]. IEEE Transactions on Signal Processing, 2003, 51(8): 2125-2136.

[16] Zhao Y B, Kim J, Liu G P. Error bounded sensing for packet-based networked control systems[J]. IEEE Transactions on Industrial Electronics, 2010, 58(5): 1980-1989.

[17] Pang Z H , Liu G P . Design and Implementation of Secure Networked Predictive Control Systems Under Deception Attacks[J]. IEEE Transactions on Control Systems Technology, 2012, 20(5): 1334-1342.

[18] Rahmani B, Markazi A H D. Variable selective control method for networked control systems[J]. IEEE transactions on control systems technology, 2012, 21(3): 975-982.

[19] 游科友, 谢立华. 网络控制系统的最新研究综述[J]. 自动化学报，2013, 39(2): 101-118.

[20] Li W, Zhang X, Li H. Co-simulation platforms for co-design of networked control systems: An

overview[J]. Control Engineering Practice, 2014, 23(1): 44-56.
[21] Dan H. TRUETIME: Simulation of control loops under shared computer resources[J]. IFAC Proceedings Volumes, 2002, 35(1): 417-422.
[22] 何坚强, 张焕春. 基于 TrueTime 工具箱的网络控制系统仿真研究[J]. 微计算机信息, 2004, 20(1): 33-34.
[23] Cervin A, Henriksson D, Lincoln B, et al. How does control timing affect performance? Analysis and simulation of timing using Jitterbug and TrueTime[J]. IEEE control systems magazine, 2003, 23(3): 16-30.
[24] Peng C, Yang T C. Communication-delay-distribution-dependent networked control for a class of T–S fuzzy systems[J]. IEEE Transactions on Fuzzy Systems, 2010, 18(2): 326-335.
[25] Qiao Y, Liu G P, Zheng G, et al. NCSLab: A web-based global-scale control laboratory with rich interactive features[J]. IEEE Transactions on Industrial Electronics, 2009, 57(10): 3253-3265.
[26] Tong X, Liao C. Co-simulation of OPNET-based wide-area communication in power system[J]. Electric Power Automation Equipment, 2010, 30(8): 134-138.
[27] 陈寅, 宋杨, 费敏锐. 基于 Simulink OPNET 的 NCS 联合仿真平台的设计与开发[J]. 系统仿真学报, 2013, 25(7): 1518-1523.
[28] Hasan M S, Yu H, Carrington A, et al. Co-simulation of wireless networked control systems over mobile ad hoc network using SIMULINK and OPNET[J]. IET communications, 2009, 3(8): 1297-1310.
[29] Yin C, Yang S, Min-rui F E I. Design and development of cosimulation platform for NCS based on Simulink and OPNET[J]. Journal of System Simulation, 2013, 25(7): 1518-1523.
[30] 顾慧卿. 基于 NS3 和 MATLAB 的网络化控制系统协同仿真平台设计[D]. 杭州. 浙江工业大学,2020.
[31] Campbell S L, Chancelier J P, Nikoukhah R. Modeling and Simulation in SCILAB[M]. New York, Modeling and Simulation in Scilab/Scicos with ScicosLab 4.4, 2010: 73-106.
[32] Boardman, Bruce. Opnet Technologies' ITDG Gives Net Admins an ACE in the Hole.[J]. Network Computing, 2000, 11(13): 28-28.
[33] 胡包钢. 科学计算自由软件: SCILAB 教程[M]. 北京: 清华大学出版社有限公司, 2003.
[34] Nagar S. Introduction to Scilab[M]. CA: Introduction to Scilab, 2017: 1-14.
[35] Carneiro G. NS-3: Network simulator 3[C]//UTM Lab Meeting April. 2010, 20: 4-5.
[36] Saluja M A K, Darg M S A. A Detailed Analogy of Network Simulators NS1, NS2, NS3 and NS4[J]. International Journal on Future Revolution in Computer Science & Communication Engineering, 2017, 3(12): 291-295.
[37] Siraj S, Gupta A, Badgujar R. Network simulation tools survey[J]. International Journal of Advanced Research in Computer and Communication Engineering, 2012, 1(4): 199-206.
[38] Henderson T R, Lacage M, Riley G F, et al. Network simulations with the ns-3 simulator[J]. SIGCOMM demonstration, 2008, 14(14): 527.
[39] 茹新宇, 刘渊, 陈伟. 新网络仿真器 NS3 的研究综述[J]. 微型机与应用, 2017, 36(20): 14-16.
[40] 张登银, 张保峰. 新型网络模拟器 NS-3 研究[J]. 计算机技术与发展, 2009, 19(11): 80-84.
[41] 周迪之. 开源网络模拟器 ns-3：架构与实践[M]. 北京: 机械工业出版社，2018.11.
[42] 赵晓东, 李刚. 程序运行原理分析[J]. 科技信息, 2006, 1(12): 48-215.

[43] Johnson G D. Networked simulation with HLA and MODSIM III[C]// Conference on Winter Simulation: Simulation-a Bridge to the Future. ACM, 1999: 1065-1070.
[44] Booch G. The unified modeling language user guide[M]. India. Pearson Education India, 2005.
[45] Page E H. The rise of Web-based simulation: implications for the high level architecture[C]//1998 Winter Simulation Conference. Proceedings. IEEE, 1998: 1663-1668.
[46] Gaspar C, Dönszelmann M, Charpentier P. DIM, a portable, light weight package for information publishing, data transfer and inter-process communication[J]. Computer physics communications, 2001, 140(1-2): 102-109.
[47] Ouni B, Rekhissa H B, Belleudy C. Inter-process communication energy estimation through AADL modeling[C]// International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design. IEEE, 2012: 225-228.
[48] 周炎涛, 李立明. TCP/IP 协议下网络编程技术及其实现[J]. 航空计算技术, 2002, 32(3): 122-124.
[49] 宋泽瑞. 基于 SOCKET 编程接口的网络通信[J]. 中国新通信, 2017, 19(5): 29-29.
[50] Godoy E P, Porto A J V. Co-simulation tools for networked control systems: Revision and utilization[J]. Journal of Control, Automation and Electrical Systems, 2013, 24(6): 816-830.
[51] Al-Hammouri A T, Branicky M S, Liberatore V. Co-simulation tools for networked control systems[C]//International Workshop on Hybrid Systems: Computation and Control. Springer, Berlin, Heidelberg, 2008: 16-29.
[52] Albeseder D, Függer M, Breitenecker F, et al. Small PC-Network Simulation-a comprehensive performance case study[M]. na, 2005.
[53] Lin Z, Liu J, Xin W, et al. Design and Implementation of Co-Simulation Platform for Wireless Networked Control Systems Based on Simulink and OPNET[J]. Research Journal of Applied Sciences Engineering & Technology, 2013, 5(6): 2059-2066.
[54] Zhao Y B, He J T, Zhu Q H, et al. Classification‐Based Control for Wireless Networked Control Systems with Lossy Multipacket Transmission[J]. IEEJ Transactions on Electrical and Electronic Engineering, 2019, 14(11): 1667-1672.
[55] Zhao Y B, Huang T, Kang Y, et al. Stochastic stabilisation of wireless networked control systems with lossy multi-packet transmission[J]. IET Control Theory & Applications, 2019, 13(4): 594-601.
[56] Hu S, Yan W Y. Stability of networked control systems under a multiple-packet transmission policy[J]. IEEE transactions on Automatic Control, 2008, 53(7): 1706-1711.
[57] Gupta V, Hassibi B, Murray R M. Optimal LQG control across packet-dropping links[J]. Systems & Control Letters, 2007, 56(6): 439-446.
[58] Sun X M, Wu D, Wen C, et al. A novel stability analysis for networked predictive control systems[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2014, 61(6): 453-457.

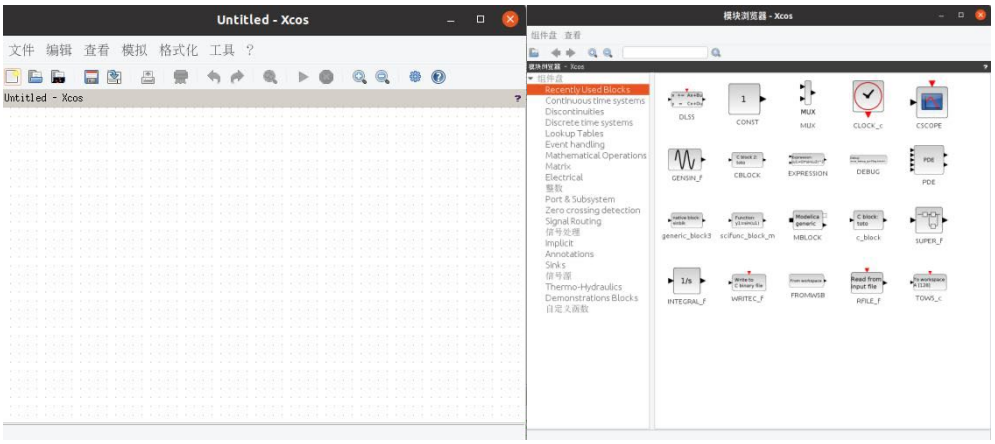
附录

附录 1 Scilab 安装使用介绍

前往 Scilab 官方网站 https://www.scilab.org/，选择操作系统版本即可下载对应版本的 Scilab 软件。本文选择下载的 Scilab 版本为 Scilab 6.1.0 Linux 64 bits 版。下载后在软件包目录下打开终端输入命令“tar xzvf scilab-6.1.0.bin.linux-x86_64.tar.gz”将安装包解压到本地，终端输入命令“cd <scilab-path>/scilab-6.1.0”打开解压后的 Scilab 文件，在终端输入命令“./bin/scilab”即可打开 Scilab。

Scilab 的使用：Scilab 官方使用教程网站为 https://help.scilab.org/，在 Scilab 的工作区间是程序运行的文件区间，如果需要运行某个文件，需要在文件浏览器进入该文件所在路径；后缀为“.sci”“.sce”的文件为 Scilab 的可执行文件。可以通过点击主页面 SciNotes 图标或在控制台输入 SciNotes 打开并用来编辑.sci 和.sce 文件。在 SciNotes 文件中可使用命令“cd 文件路径”进入文件路径，执行.sci 和.sce 文件必须在此文件所在路径中，也叫工作区间。编写好的可执行文件可在控制台通过命令“exec 文件名”执行。

Xcos 工具箱：Scilab 软件内置的可视化仿真工具箱 Xcos 具有强大的功能，其能够媲美 Simulink 工具箱，点击 Scilab 主页面上的 Xcos 图标或在控制台输入 Xcos 启动 Xcos，Xcos 工具箱界面和模块库如附图 1-1 所示。编写完成后的 Xcos 模型文件后缀为“.zcos”，运行 Xcos 模型可在控制台输入“[result]=importXcosDiagram('模型名')和“Xcos_simulate(scs_m, 4)”。



附图 1-1 Xcos模块

Extra figure 1-1. Xcos module

附录 2 Scilab 软件接口介绍

Scilab 的常用 API：Scilab 6.0 版本后预留了一些与外部软件和文件系统交互的 API 如 api_scilab，使用这种接口允许从 Scilab 内存读取/写入数据，此 API 用于扩展 Scilab 功能。Scilab 的许多库是在免费/开源或专有市场上开发的，这个库可以在像 Scilab 这样的高级语言中加载和使用。Scilab 的 API 提供了与此类库交互的功能。Scilab 也可以用作第三方软件的计算引擎。Scilab 用于 C/C++代码时此功能称为 call_scilab 时或用于 Java 时被称作 javasci。对变量的访问是通过它们的名称（命名变量）完成的。另外 Scilab 也能通过 mex 动态库命令“ilib_mex_build”编译 c++文件生成链接动态库的命令，通过命令运行 C++文件。

与操作系统交互：Scilab 也可在.sci 或控制台与操作系统交互，在 Ubuntu 系统下可使用命令“unix('gnome-terminal – 命令 –参数" ')”打开终端并执行终端命令，与文件系统的交互式通过命令“loadmatfile”和“fprintfMat”执行从文件向控制台中读和写数据。

附录 3 ns-3 的使用介绍

ns-3 的使用介绍主要分为：下载、编译和测试；仿真脚本；ns-3 文档系统；仿真脚本的输入输出和程序控制。

（一）下载、编译和测试

由于 ns-3 没有图形用户界面，要使用 ns-3 进行网络模拟，需要经过以下步骤：下载源代码、编译源代码、编写模拟脚本和运行模拟脚本，ns-3 的运行需要很多依赖环境，这些依赖环境包括 gcc g++ Python、mercurial、bzr 等，使用 Ubuntu 系统终端命令“apt-get install”安装。具体需要的依赖库安装参阅 https://www.nsnam.org/wiki/Installation。

下载：ns-3 源代码的下载需要登录 ns-3 的官方网站 https://www.nsnam.org/，下载 ns-3 源代码后运行解压缩命令将 ns-3 源代码压缩包解压到本地。

编译： 打开解压后的 ns-3 文件夹，终端运行命令“./build.py”“./waf clean”，之后使用 debug 编译命令“./waf configure --build-profile=debug --enable-examples --enable-tests”打开 debug 并开启例子及帮助，之后运行命令./waf build 进行构建，输入命令“./test.py -c core”进行测试 ns-3 是否安装完整。

测试：终端输入命令“./waf –run hello-simulator”运行模拟脚本例子来测试 ns-3 是否编译完成，输出“Hello simulator”时表示 ns-3 安装完成并能正常运行。

（二）仿真脚本

用户想要进行 ns-3 网络仿真实验时，需要编写仿真脚本程序，之后运行仿真脚本程序，得到仿真结果，仿真脚本程序的编写分为以下几个部分：

头文件：一般需要包含 ns-3 仿真核心 core-module.h、network-module.h 等头文件，如 core-module.h 文件定义了 ns-3 的模拟事件和事件调度等核心功能，network-module.h 文件定义了网络节点、网络分组和网络地址等基本网络组件。用户也可以根据自己仿真需要添加其他头文件，如经常使用的 internet-module.h 文件定义了 TCP/IP 协议，applications-module.h 文件定义了应用层的分组收发模型。

命名空间：ns-3 的源码被命名空间“namespace ns3”保护以区分 ns-3 项目和非 ns-3 项目。

宏定义辅助信息：打印调试信息的宏 NS_LOG_DEBUG 和打印错误信息的 NS_LOG_ERROR 宏等。

main()函数：ns-3 的模型搭建和仿真都在 main()函数中进行，在 main()函数中包含需要读取命令行的输入函数 cmd.Parse(参数)，仿真的时间格式 Time::SetResolution()以及用于仿真过程中的组件打印信息 LogComponentEnable()。创建网络拓扑：ns-3 创建网络拓扑是按照物理网络中的网络分层模型进行，网络

拓扑的建立需要经过网络分层模型的协议安装，分为以下几个步骤：

（1）创建结点：使用结点容器创建多个结点对象。

（2）物理层和数据链路层：网络拓扑是由结点和连接结点的信道组成。在 ns-3 中结点和信道被抽象为 Node 和 Channel 以及结点中用于建立信道连接的网络设备 NetDevice 类等三个类。NetDevice 主要负责实现链路层的协议，Channel 主要负责物理层协议的实现。

配置信道属性：使用助手类设置信道的属性，助手类和属性将在接下来介绍。创建信道并连接结点：使用信道助手类加 Install()函数进行信道的创 建 并 安 装 到结点上。

（3）网络层和传输层：安装 TCP/IP 协议族，ns-3 的 TCP/IP 协议族包括传输层的 TCP/UDP 协议，网络层的 IP（IPv4 与 Ipv6）、ICMP（ICMPv4 与 ICMPv6）和一系列路由协议。为结点安装 TCP/IP 协议栈的助手类是 InternetStackHelper。具体为：InternetStackHelper stack；stack.Install(nodes)。

之后需要为结点的网络设备分配 IP 地址。通过 IPv4AddressHelper（IPv4 地址）和 IPv6AddressHelper（IPv6 地址）完成。

设置路由：ns-3 的多个网络子网间存在通信行为时就需要设置路由，全局路由设置通过脚本中助手类 Ipv4GlobalRountingHelper 的 PopulateRountingTables()完成的。

（4）应用层：安装应用程序，ns-3 的应用是对物理世界中应用程序内部网络通 信 的 抽 象 ， 模 拟 分 组 的 发 送 和 接 收 行 为 。 通 过 ApplicationContainer clientApps=echoClient.Install(nodes)完成。

（5）仿真结果：数据生成与跟踪，产生数据是网络模拟的必须功能，是一个网络仿真实验的结果输出，EnablePcapAll()函数是收集信道上所有结点的链路层分组收发记录，记录将会被保存到后缀为.pcap 的文件中。

启动与结束：ns-3 脚本的最后一步 Simulator::Run()；Simulator::Destroy()；return()，Run()函数执行之前步骤定义的所有操作，完成之后 Destroy()函数执行销毁操作，最后，因为仿真脚本执行本质是一个 C++程序的 main()函数，所以用 return 返回 0 告诉操作系统程序执行完成。

编写要脚本程序后在终端输入命令“./waf －run 脚本名称”即可运行脚本。

（三）ns-3 文档系统

在进行 ns-3 仿真时遇到问题时可以查看 ns-3 的文档系统和邮件系统，是用户、开发者讨论问题，进行技术交流的专业平台。

ns-3 Doxygen：用于查看 ns-3 的源代码，具有 ns-3 源代码完整目录和框架，方便用户了解 ns-3 的设计架构。在 Doxygen 上，除了可以进行浏览源代码外，还可

以查询 ns-3 中类的设计、函数和变量的定义，了解类与类之间的关系等一些非常详细的信息。Doxygen 文档网址是 https://www.nsnam.org/docs/doxygen/index.html，作为一个源代码查询的工具网站，Doxygen 对用户和开发者的帮助很大。

ns-3 Wiki 主页：提供 ns-3 的安装教程，代码的提交和发布版本等辅助作用。ns-3 的 Wiki 中还存在没有进入 ns-3 发布版本代码的模块的补丁，这些未发布的代码可能对部分用户的网络模拟有作用，避免了用户重复工作。

ns-3 Model-library：ns-3 发布版本各个模块的开发者编写的使用文档。

ns-3 manual：ns-3 的架构、设计原理和 ns-3 核心模块的主要技术实现细节。

ns-3 tutorial：ns-3 的基本概念和基本使用方法，适合用户的入门级使用教程。

ns-3 邮件系统：可以通过邮件系统向开发者提问，与其他用户讨论问题。是用户、开发者和维护者的主要的交流工具。

（四）仿真脚本的输入输出和程序控制

使用仿真软件进行仿真实验的过程总体来说，给软件一个输入，仿真结果的输出。ns-3 将整个过程分为参数输入、数据输出、行为控制，此外还包括仿真的辅助信息打印和脚本助理等

参数输入：属性变量，Attribute（属性）是在 ns-3 网络模拟中用户可以设定的参数，如在网络设备中设置传输速率和信道传播延时等网络属性，ns-3 中的属性是通过 C++类内变量实现的。属性变量会影响网络性能和模拟的结果，用户可以通过属性系统，对 ns-3 脚本程序进行属性配置。属性系统的任务就是把内部的私有成员变量变成外部可改变的参数，这样仅通过一个脚本就可以实现模拟多个拓扑相同但是配置不同的网络场景。用户配置属性有三种方法，第一种方法：可使用助手类、命令行和 Config::SetDefault()配置属性；第二种方法：使用 ObjectBase::SetAttribute()函数；第三种方法：使用 Config::Set()。读取属性：配置完属性后，ns-3 需要通过 ObjectBase::GetAttribute()函数来获取一个对象的属性值。之后用户可以通过 GetTypeId()函数进行查找属性来了解 ns-3 已搭建的网络拓扑的属性。

数据输出：trace 变量是对网络模拟过程中一些重要的网络行为进行记录，并以用户可配置的格式输出记录的结果，网络行为包括关键性能指标的变化如各个网络协议层的分组发送、接收和丢失事件等。trace 变量本质上也是 C++类内成员变量，是一个函数指针。用户使用时需要在脚本中预先定义一个回调函数，通过 trace 系统将其与某个 C++对象内部的 trace 变量相关联。配置 trace 变量通过两种方法，第一种方法：Config::Connect()可以通过将用户定义的回调函数指针赋值给 trace 变量；第二种方法：通过助手类配置 trace 变量，助手类其实是对第一种方法的封装。包括 NetDevice 助手类和 InternetStackHelper 助手类。用户可通过 ns-3 自

带脚本和 ns-3model library 中查找，还可以通过 GetTypeId()函数或 Doxygen 主页中查找。

行为控制：除了执行脚本，ns-3 的 waf 命令行话可以设置脚本的自定义变量和属性变量，这种通过命令行设置变量的方法不需要再进行脚本的重新编译。通过命令行设置脚本的属性变量需要在脚本中添加下列信息：CommandLine cmd；cmd.AddValue（属性设置）；cmd.Parse(argc,argv)。

辅助信息：Log 系统时 ns-3 中除了 trace 变量外的另一个数据输出，trace 不能记录所有的网络，而且 trace 时存储在静态存储区，过多的 trace 变量不利于大规模模拟。Log 作为一种轻量级的输出系统作为对 trace 变量的补充。Log 系统主要包括一些打印信息的宏定义如 NS_LOG_ERRO、NS_LOG_WARN、NS_LOG_DEBUG、NS_LOG_INFO、NS_LOG_FUNCTION、NS_LOG_LOGIC、NS_LOG_UNCOND 等。

主要的函数 LogComponentEnable()用来按照 ns-3 的文件组织架构分层打印 Log 信息。

脚本助理：助手类，ns-3 内特殊的类，类名都是以 Helper 为后缀。助手类是设计用来屏蔽实现的细节来降低脚本的编写难度，助手类能够帮助完成从网络拓扑的构建、结点协议栈的安装、属性配置到数据生成等操作，在 ns-3 内大部分操作都可以通过助手类完成。ns-3 的每一个模块都有一个自己的助手类，如为节点安装 TCP/IP 协议栈的 InternetStackHelper 助手类是在文件 internet-stack-helpeer.h 中。

附录 4 ns-3 网络仿真原理介绍

（一）ns-3 事件介绍

ns-3 是离散事件仿真软件，贯穿 ns-3 运行的整个过程离不开事件，可以把事件简单的理解为执行了某种操作的 C++函数。在 ns-3 官方定义中凡是继承基类 ns3::EventImpl 的子类都是一个模拟事件，比如继承 ns3::EventImpl 类的应用启动事件 StartApplication()、应用停止事件 StopApplication()、节点发送事件类等。事件的执行调用是 EventImpl ::Invoke 类在关联到该事件的时间后通过仿真引擎调用。

附表4-1 ns3::Scheduler::Event事件结构

Extra table 4-1. ns3::Scheduler::Event Struct Reference

Public Attributes	公有成员函数	保护成员函数
EventImpl * impl	void Invoke() void Cancel() bool IsCancelled()	virtual void Notify (void)=0
EventKey key	uint64_t m_ts事件时间戳 uint32_t m_uid事件唯一id uint32_t m_context事件上下文信息	

在 ns-3 仿真过程中时间仿真时间与现实的时间不同，在 ns-3 内默认事件的完成是瞬时的，而时间是通过安排下一个事件的开始时间及该事件的执行这一循环过程来推进的。

总之，整个连续时间内的网络过程被分成若干个离散事件，ns-3 按照这一时间顺序依次执行所有的事件，而对于相同时间发生的事件，ns-3 按其在程序内被创建的先后顺序执行，避免随机顺序执行引起的仿真结果的不确定。网络的状态是只在事件被执行后才会发生改变，事件之间的网络状态是不会发生变化的，一个事件也可以按照需要具备取消和创建未来某一事件的功能。

（二）计划事件函数：schedule()

现实世界的网络通信系统的运行大多数时候都是一个随时间变化的连续过程，在 ns-3 程序中把这一连续变化的网络过程按照时间变化的顺序分成一系列的离散事件。ns-3 模拟网络的整个过程就是以时间顺序来执行这些分割好的离散事件。

ns-3 将这些分割好的离散事件安排到一个数据存储结构格式的事件列表，通过安排的事件压入事件列表和即将执行的事件弹出事件列表并返回该事件的 C++

指针的步骤来维护和更新这个事件列表。在 ns3 中事件是以变量 event 表示，通过 event_id 进行事件的区分，之后通过 Schedule()函数安排事件，通过 C++指针指向事件列表中需要执行事件地址，简单的说就是 Schedule()函数中包含了某个事件的函数，这一函数以回调函数的形式存在，某个事件函数指针通过 Invoke 函数驱动事件执行，之后调用 Remove 函数移除事件列表中执行的事件。

Schedule()中的回调函数指针用于安排事件时指向代表某个事件的 C++函数，将事件注册成回调函数，之后通过 Schedule()安排进事件列表。

ns-3 计划事件函数形式有如下两种：

Simulator::Schedule（tNext,&MyApp::SendPacket,sp.get()）；第一个参数为计划事件延迟，第二个参数是事件回调函数指针，第三个参数是回调函数参数。

Simulator::Schedule（tNext,&MyApp::SendPacket,this）；第一个参数为计划事件延迟，第二个参数是事件回调函数指针，第三个是回调函数所属的 C++对象指针。

（三）回调函数：Callback 类

在 ns-3 中经常使用回调函数，被广泛应用到 trace 变量配置、计划事件和不同协议层对象间的分组传递等场景。回调函数的使用分为三步：定义函数指针、赋值和调用。函数指针是由 Callback 类模板定义，可以定义函数的返回值和最多 9 个形参，Callback 对象的创建和赋值操作是由 MakeCallback()函数完成。

附录 5 Xcos 模型驱动程序

附表5-1 Xcos模型驱动程序

Extra table 5-1. Xcos model driver code

Xcos 模型驱动程序

```
exec('controller.sci', -1)
exec('actuator.sci', -1)
exec('Plant.sci', -1)
       order=2; //系统阶数
       nodenumber=4;//设置节点个数。
      C_A_data=[0,0];
      S_C_data=[1,1];
      main;//运行交互客户端接口程序，进行一次交互。
      current_time=1.5;
      State.values=[1,1];//状态量设置初始值
       loadmatfile("node1receive.txt");//ns3 仿真后 node 1 接受的数据包内容
       while    node1receive==9999
                  main
                  loadmatfile("node1receive.txt");
                  end
      S_C_SCI.values=[node1receive]';//数据传递到 scilab 运行空间
       loadmatfile("/home/ling/Scilab/scilab-6.1.0/client/node2receive.txt");
            while    node2receive==9999
                  main
                  loadmatfile("node2receive.txt");
                  end
      C_A_SCI.values=[node2receive]';//数据传递到 scilab 运行空间
      ut_memory.values=C_A_SCI.values;
      Xcos('daxingtest01.zcos');//打开 Xcos 模型
      [result]=importXcosDiagram('daxingtest01.zcos');
      Xcos_simulate(scs_m, 4);//从 scilab 数据传递到 Xcos 运行
      outtemp=[S_C_NS3.values];
      loadmatfile("/node1receive_delay.txt");//node2 的延迟数据
      loadmatfile("node2receive_delay.txt");
      midnumber=node1receive_delay+node2receive_delay;
      temp=[midnumber]; //建立中间缓存区
      temptest1=C_A_data;
      temptest2=C_A_SCI.values;
      ut_memory.values=[ut_memory_update.values];
      ut_memory.time=(5.01:5:10)';
   t=1;      //循环次数初值，即仿真时常=仿真次数*仿真步长
   while t<100//设置循环体运行次数（采样 sample），进行联合仿真
      C_A_data=[C_A_NS3.values];
      S_C_data=[S_C_NS3.values];
      State.values=S_C_NS3.values;
      main;
      temptest1=[temptest1,C_A_NS3.values];
      temptest2=[temptest2,C_A_SCI.values];
     loadmatfile("node2receive_delay.txt");
     loadmatfile("node1receive_delay.txt");
     midnumber=node1receive_delay+node2receive_delay;
     temp=[temp,midnumber];
```

```
    loadmatfile("node1receive.txt");
        while    node1receive==9999
         main
         loadmatfile("node1receive.txt");
   end
    loadmatfile("node2receive.txt");
      while    node2receive==9999
         main
         loadmatfile("node2receive.txt");
         end
  S_C_SCI.values=[node1receive]';
  C_A_SCI.values=[node2receive]';
  [result]=importXcosDiagram('daxingtest01.zcos');
  Xcos_simulate(scs_m, 4);
  outtemp=[outtemp;S_C_NS3.values]; //缓存系统输出值到文件 outrecord.txt

  ut_memory.values=[ut_memory_update.values]; //更新 Actutor 中存储的 ut 的值
  //获取节点 1 和节点 2 的接受数据包时间
  loadmatfile("node2receive_time.txt");
  loadmatfile("node1receive_time.txt");
  node1recv_time=node1receive_time;
  node2recv_time=node2receive_time;
  //获取闭环延时
  current_time=current_time+0.03;
  //获取闭环延时
  close_loop_delay(t)=current_time-ut_memory.values(1,2);
  t=t+1;
end//结束循环体
fprintfMat('outrecord.txt', outtemp, "%1f");
fprintfMat(' record.txt', temp, "%1f");
```

致　谢

两年多的时光飞逝而过，我的研究生的生涯也即将结束，回顾这两年多的时间，我付出了很多也收获了很多，不仅学到了知识，还开阔了眼界。

在此论文完成之际，首先要衷心感谢我的导师赵云波教授。在攻读硕士学位期间，导师不断地引导我发现问题和主动解决问题，并且教导我们要找到学习的方法，主动的去学习和提升自己，这些将会对我以后的工作和生活有很大的帮助。在毕业论文写作期间，导师对我论文的研究方案、实验思路和实验结果的分析提供了许多直接、深入、有益的指导、建议和帮助，让我能顺利完成毕业论文的撰写。

其次还要感谢顾慧卿师兄在此课题上取得的成果为我的研究做了铺垫。感谢我的师兄梁启鹏对我实验过程的帮助，同时还要感谢我的师兄唐敏、朱创，师姐王岭人、朱巧慧，同学吴芳、闫文晓、卢子轶、花婷婷、郝小梅、赵丽丽、王保佳等对我生活和学习上的照顾，也要感谢我的室友朱振强、周涛为我提供了一个和谐的住宿环境。感谢浙江工业大学为我提供一个良好的生活和科研环境，这些都使得我能够全身心的投入到学习和科研中去。

最后我要感谢我的父母和姐姐们对我生活和学业上的无条件支持，让我一直有动力在科研的道路上走下去。

最后，感谢各位专家，学者和老师们抽出宝贵的时间对本文的评阅。

作者简介

1 作者简历

1994 年 03 月出生于河南太康。

2019 年 9 月——2022 年 1 月，浙江工业大学信息工程学院控制工程专业学习，攻读工程硕士学位。

2 参与的科研项目及获奖情况

[1] 中国国家自然科学基金委面上项目 2017-2020：基于资源调度和预测控制的无线网络化控制系统的联合设计. （61673350）

[2] 中国国家自然科学基金委面上项目 2022-2025：基于数据驱动和联合设计的无线网络化控制系统的使能建模和设计. （62173317）

3 发明专利

[1] 赵云波，卢帅领，郝小梅，梁启鹏. 一种基于 XCOS 和 NS-3 的协同仿真时间同步方法. 中国, 2021-04-12. （除导师外第一作者, 202110398847.X, 公开）

[2] 赵云波，卢帅领，梁启鹏，闫文晓. 一种基于 TCP 套接字的 SCILAB 与 NS-3 协同仿真接口方法.中国，2021-01-06. （除导师外第一作者, 202110011849.9, 公开）

[3] 赵云波，卢帅领，梁启鹏. 一种基于有扰无线网络化控制系统延时估计的逼近控制策略方法.中国，2021-04-28.（除导师外第一作者, 202110488855.3, 公开）

学位论文数据集

<table>
<tr><td>密　级*</td><td>中图分类号*</td><td>UDC*</td><td>论文资助</td></tr>
<tr><td>公开</td><td>TP391.9</td><td>681.5</td><td></td></tr>
<tr><td>学位授予单位名称*</td><td>学位授予单位代码*</td><td>学位类型*</td><td>学位级别*</td></tr>
<tr><td>浙江工业大学</td><td>10337</td><td>工程硕士</td><td>全日制专业型研究生</td></tr>
<tr><td>论文题名*</td><td colspan="3">基于 Scilab 和 ns-3 的网络化控制系统联合仿真平台设计</td></tr>
<tr><td>关键词*</td><td colspan="2">网络化控制系统，网络通信，Scilab 和 ns-3，联合仿真平台，开源软件</td><td>论文语种*</td></tr>
<tr><td>并列题名</td><td colspan="2">无</td><td>中文</td></tr>
<tr><td>作者姓名*</td><td>卢帅领</td><td>学　号*</td><td>2111903284</td></tr>
<tr><td>培养单位名称*</td><td>培养单位代码*</td><td>培养单位地址*</td><td>邮政编码*</td></tr>
<tr><td>浙江工业大学环境学院</td><td>10337</td><td>杭州市潮王路 18 号</td><td>310032</td></tr>
<tr><td>学科专业*</td><td>研究方向*</td><td>学　制*</td><td>学位授予年*</td></tr>
<tr><td>控制工程</td><td>网络化控制</td><td>2.5 年</td><td>2022 年</td></tr>
<tr><td>论文提交日期*</td><td colspan="3">2022 年 01 月</td></tr>
<tr><td>导师姓名*</td><td>赵云波</td><td>职　称*</td><td>教授</td></tr>
<tr><td>评阅人</td><td>答辩委员会主席*</td><td colspan="2">答辩委员会成员</td></tr>
<tr><td></td><td>杨东勇</td><td colspan="2"></td></tr>
<tr><td colspan="4">电子版论文提交格式：文本（ √ ）图像（　）视频（　）音频（　）多媒体（　）其他（　）</td></tr>
<tr><td colspan="2">电子版论文出版（发布）者</td><td>电子版论文出版（发布）地</td><td>版权声明</td></tr>
<tr><td colspan="2"></td><td></td><td></td></tr>
<tr><td>论文总页数*</td><td colspan="3">74</td></tr>
<tr><td colspan="4">注：共 33 项，其中带*为必填数据，为 25 项。</td></tr>
</table>