



浙江工业大学

硕士学位论文

基于 NS3 和 MATLAB 的网络化控制系统协同

仿真平台设计

作者姓名	顾慧卿
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：2020 年 01 月

Design of Cooperative Simulation Platform for Networked Control System Based on NS3 and MATLAB

Dissertation Submitted to

Zhejiang University of Technology

in partial fulfillment of the requirement

for the degree of

Master of Engineering



by

Hui-qing GU

Dissertation Supervisor: Prof. Yun-bo ZHAO

Jan. 2020

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： 顾慧卿 日期：2019年 12月

学位论文授权使用授权书

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： 顾慧卿 日期：2019年 12月
导师签名： 赵云波 日期：2019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 NS3 和 MATLAB 的网络化控制系统协同仿真平台设计

Design of Cooperative Simulation Platform for Networked
Control System Based on NS3 and MATLAB

作者姓名 顾慧卿

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: 2019 年 11 月 26 日

基于 NS3 和 MATLAB 的网络化控制系统协同仿真平台设计

摘 要

伴随着控制和通信技术的不断革新,网络化控制系统已经成为控制领域的研究热点之一,通过仿真软件进行算法验证逐渐成为网络化控制系统研究的重要环节。然而网络化控制系统是结合网络通信和控制两大特征的系统,传统的网络化控制系统仿真工具对这类结合交叉领域的系统无法进行有效地精确仿真,这就需要不同领域的仿真软件之间建立纽带,使不同的工具一起协同工作,各自负责系统中各自专长的部分。因此,设计并开发网络化控制系统协同仿真平台具有相当重要的现实意义。

本文深入分析了通信网络与控制系统仿真软件的特性,基于网络化控制系统仿真平台的研究现状,选取了两种普遍使用的仿真软件,分别是能够胜任物理仿真器的 MATLAB,以及在网络通信领域近年来快速发展的 NS3 作为网络仿真器,构建了一种新型的网络化控制系统仿真平台。本文的主要工作如下:

(1) 设计了协同仿真平台的总体方案,确定了仿真平台依赖的各软件模块与协同仿真架构的主导方式。以此设计了一个合适的系统架构,在该架构下两个软件能够互相通信,同时也实现了单仿真器内部各模块之间的通信。

(2) 设计了仿真平台软件系统的时间同步方案。针对两种仿真软件间需要进行时间同步的问题,根据仿真器驱动方式的不同和仿真目标的不同对两种可行的同步方式予以实施,分别为控制器时间驱动采用的主从式时间同步方式,以及控制器事件驱动采用的基于控制器事件的全局驱动式同步方式。

(3) 设计了仿真平台软件系统的底层模块实现方案。针对部分模块予以改进并设计了新的模块,例如两种不同仿真器的时间控制模块。并使用 MATLAB GUI 套件开发了仿真平台交互界面以及将可视化功能加入系统。

(4) 通过使用该网络化控制系统协同仿真平台,对两种控制系统的主动补偿方案进行测试。仿真结果验证了主动补偿方案的有效性,以及仿真平台具有较好的可靠性与稳定性,符合仿真平台设计的基本要求。

最后,对本文的工作进行分析与总结,指出了仿真平台在设计方面的优点与不足,并展望了未来的研究方向。

关键词: 协同仿真平台, NS3 与 MATLAB, 网络化控制系统, 时间同步, 主动补偿

DESIGN OF COOPERATIVE SIMULATION PLATFORM FOR NETWORKED CONTROL SYSTEM BASED ON NS3 AND MATLAB

ABSTRACT

With the continuous innovation of control and communication technology, networked control system has become one of the research hotspots in the field of control. The algorithm verification by simulation software has gradually become an important part of networked control system research. However, the networked control system is a system that combines the two characteristics of network communication and control. The traditional networked control system simulation tool can not effectively and accurately simulate such a system in the cross-domain, which requires different fields of simulation software. Establish a bond that allows different tools to work together and each is responsible for the respective expertise of the system. Therefore, designing and developing a networked control system co-simulation platform has considerable practical significance.

This paper deeply analyzes the characteristics of communication network and control system simulation software. Based on the research status of networked control system simulation platform, two commonly used simulation softwares are selected, which are MATLAB capable of competent physical simulator and network communication field. In recent years, the fast-developing NS3 as a network simulator has built a new networked control system simulation platform. The main work of this paper is as follows:

(1) The overall scheme of the collaborative simulation platform is designed, and the leading methods of the software modules and the co-simulation architecture that the simulation platform depends on are determined. In this way, a suitable system architecture is designed, in which two softwares can communicate with each other, and also realize communication between modules within a single simulator.

(2) The time synchronization scheme of the simulation platform software system is designed. For the problem of time synchronization between the two simulation softwares, the two feasible synchronization modes are implemented according to the different driving modes of the simulator and the different simulation targets, which are the master-slave time synchronization mode adopted by the controller time driving.

And the controller-driven globally-driven synchronization method used by the controller event driver.

(3) The implementation scheme of the underlying module of the simulation platform software system is designed. Some modules have been improved and new modules have been designed, such as time control modules for two different simulators. The simulation platform interactive interface was developed using the MATLAB GUI suite and visualization was added to the system.

(4) By using the networked control system co-simulation platform, the active compensation schemes of the two control systems are tested. The simulation results verify the effectiveness of the active compensation scheme, and the simulation platform has good reliability and stability, and meets the basic requirements of the simulation platform design.

Finally, the paper analyzes and summarizes the work of this paper, points out the advantages and disadvantages of the simulation platform in design, and looks forward to the future research direction.

KEY WORDS: Co-simulation platform, NS3 and MATLAB, Networked control system, Time synchronization, Active compensation

目 录

摘 要.....	I
ABSTRACT.....	II
目 录.....	IV
第一章 绪 论.....	1
1.1 研究背景及意义.....	1
1.2 网络化控制系统概述.....	2
1.3 网络仿真器 NS3 介绍.....	3
1.4 国内外研究现状.....	4
1.4.1 网络化控制系统的研究现状.....	4
1.4.2 网络化控制系统仿真平台研究现状.....	4
1.5 本文的主要研究内容与章节安排.....	6
第二章 网络化控制系统协同仿真平台总体方案设计.....	9
2.1 网络化控制系统协同仿真平台总体方案设计.....	9
2.1.1 仿真平台需求分析.....	9
2.1.2 仿真平台依赖环境配置.....	10
2.2 协同仿真平台架构与模块.....	11
2.2.1 网络化控制系统协同仿真平台架构.....	11
2.2.2 NS3 模块.....	12
2.2.3 MATLAB 模块.....	13
2.3 协同仿真平台数据交换与传递设计.....	13
2.3.1 仿真软件间的数据交换.....	13
2.3.2 单仿真器内的数据传递.....	13
2.4 本章小结.....	14
第三章 网络化控制系统协同仿真平台时间同步方案设计.....	15
3.1 时间同步策略.....	15
3.1.1 MATLAB/Simulink 与 NS3 的仿真驱动方式.....	15
3.1.2 协同仿真时间同步方案.....	16
3.1.3 同步方案的选择.....	18
3.2 Linux 下的 Socket 编程.....	19
3.3 同步信息交互模型.....	19

3.4 本章小结.....	20
第四章 网络化控制系统协同仿真平台底层模块设计.....	23
4.1 NS3 各模块程序设计.....	23
4.1.1 基本类的描述.....	23
4.1.2 外部驱动仿真器类（EDS 类）程序设计.....	23
4.1.3 辅助类的设计.....	26
4.1.4 NS3 仿真脚本的设计.....	30
4.2 MATLAB/Simulink 各模块程序设计.....	31
4.2.1 MATLAB 仿真驱动模块.....	31
4.2.2 控制系统 Simulink 模型.....	37
4.2.3 MATLAB 客户端组件.....	38
4.2.4 仿真平台交互界面.....	39
4.3 本章小结.....	41
第五章 仿真平台测试与结果分析.....	43
5.1 桥式吊车远程控制系统仿真测试.....	43
5.1.1 系统描述与控制方案.....	43
5.1.2 吊车远程控制系统仿真.....	44
5.2 通信资源受限网络控制系统主动补偿方法的仿真测试.....	50
5.2.1 系统描述与控制方案.....	50
5.2.2 基于 TrueTime 的仿真.....	53
5.2.3 基于 STM32 的测试.....	54
5.2.4 基于协同仿真平台的仿真.....	56
5.3 本章小结.....	61
第六章 结论与展望.....	63
6.1 结 论.....	63
6.2 展 望.....	63
致 谢.....	65
参考文献.....	66
作者简介.....	70
1 作者简介.....	70
2 参与的科研项目及获奖情况.....	70
3 发明专利.....	70
学位论文数据集.....	71

第一章 绪 论

1.1 研究背景及意义

随着通信网络的发展和对设备远程控制的需求，利用网络构成控制系统已成为一种发展趋势，网络化控制系统应运而生。不同于传统的控制网络，网络化控制系统（Networked Control Systems, NCS）是通过通信网络进行信息交换与传输的闭环反馈控制系统^[1]，其将网络技术、通信技术和控制技术有机的结合起来。网络化控制系统相比传统的点对点控制，具有布线简单、方便维护以及高扩展性等许多优点。网络化控制系统现已广泛应用于工业控制、自动驾驶、无人机等远程控制领域^[2-5]（图 1-1）。



图1-1 网络化控制系统的应用场景 (a) 工业控制 (b) 自动驾驶 (c) 无人机^[1]
Figure 1-1. Application scenario of networked control system (a) Industrial control (b) Autopilot (c) Drone

同时伴随计算机技术的快速发展，不同的科学领域利用各自领域的基础理论与工程知识，结合快速发展的仿真软件技术，都发展出了多样化的仿真软件及工具。由于不同的仿真软件的发展过程相对独立，又分别建立在各自领域的基础理论与工程知识上进行开发，因此不同的仿真软件往往仅能对某一特定领域进行较为全面而精确的仿真，而对其他领域无法建立有效的仿真，这就导致不同仿真软件之间的壁垒和隔阂。即使是优秀的综合仿真软件 MATLAB，能够展露在多领域的仿真能力，也未必能实现特定领域的高精度仿真。然而现实中的实际系统往往是多种领域结合的产物，因此如果在单个仿真软件中对于该系统的仿真，往往要对该系统部分特征进行一定程度的删减和简化，这就会大大影响仿真结果的精确度和可信度。

现有针对网络化控制系统的算法的验证一般是通过在 MATLAB 数值仿真完成的，为了让网络化控制系统的仿真更贴近系统的实际运行情况，就需要能够全面

仿真网络化控制系统特征的仿真工具，然而不幸的是网络化控制系统正是结合了网络通信和控制两大特征的系统，对于这类结合交叉领域系统的精确仿真，单仿真软件进行仿真会降低仿真可信度，因而不同领域仿真软件之间的协同仿真的策略被提出，该策略使不同的工具一起工作，各自负责系统中各自专长的部分。协同仿真在网络化控制系统的仿真中得到了有效的实施，结合不同的仿真软件、协同方式以及仿真目标，演化出了诸多的变种。通过协同仿真，网络化控制系统应用的整体行为被暴露出来，可以进行研究，例如，网络拓扑，介质访问，路由协议以及流量模式如何影响整体控制系统的性能等，这在单独仿真控制或通信系统时是难以实现的。此外通过协同仿真，还能分析涉及多个控制回路的复杂系统的行为，这在数学上是难以进行的^[6]。

本文在网络化控制系统的研究背景下，结合多仿真软件协同仿真策略，设计并开发了基于 NS3 和 MATLAB 的网络化控制系统仿真平台。对提高我国仿真软件设计水平，以及推进我国网络化控制技术发展有着积极的意义。

1.2 网络化控制系统概述

网络化控制系统的发展经历了三个阶段，分别是基于现场总线的控制系统、基于工业以太网的控制系统和基于无线通信网络的控制系统。早期提出的现场总线控制系统通过共享的串行通信网络把所有的传感器、控制器与执行器连接起来，使得传输信息变得数字化、串行化以及控制功能分散化。之后伴随着以太网技术的逐渐成熟，人们开始研究将以太网作为总线应用到控制系统。相比传统的现场总线而言，以太网具有高传输速度、高带宽、低成本等优点，为了消除传统以太网中 CSMA/CD 机制引起的不确定性，交换式以太网成为实时工业应用的一种非常有前景的替代方案。最近随着多种无线通信技术的成熟以及移动互联网的普及，采用无线网络作为数据传输媒介的控制系统成为研究的热点，相比于前两种网络化控制系统，顾名思义无线网络化控制系统脱离了对线缆的依赖，这最大程度的降低了系统的搭建成本，同时提升了系统的可靠性与安全性。通过使用无线设备，系统获得更高的灵活性和移动性，传感器能以插即用的方式放置在设备上，网络拓扑也能更自由地变化，设备能移动到不同地点而仍然保持通信畅通。同时，网络通信协议的标准化也将极大的简化了连接和实施成本。

如图 1-2 所示，一个典型的网络化控制系统主要由控制器节点、传感器节点、执行器节点、被控对象和通信网络构成。而网络化控制系统的结构按照主从节点的功能划分，可以分为功能分散式与功能集中式两类网络化控制系统^[7,8]。

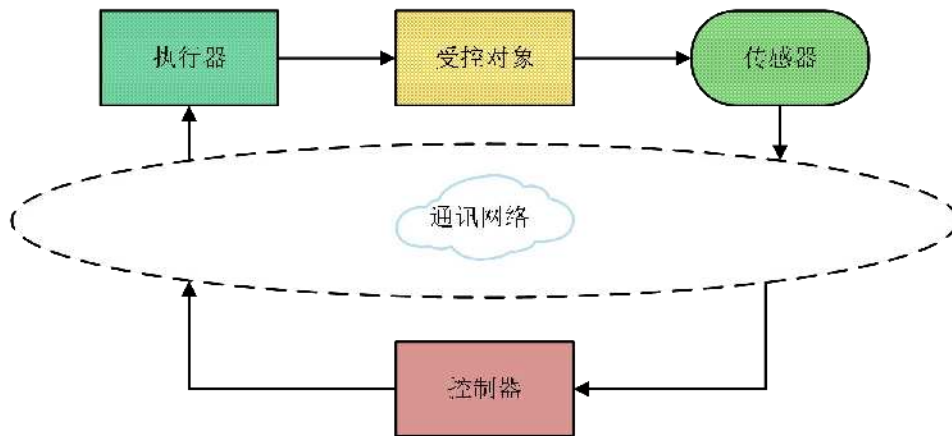


图1-2 典型的网络化控制系统结构框图

Figure 1-2. Simple networked control system block diagram

1.3 网络仿真器 NS3 介绍

网络仿真是用于评估网络拓扑或协议最常用的方法，到目前为止已经开发出了众多的网络仿真器，例如 NS2, NS3, OMNET ++Riverbed, GloMoSiM 和 Jist 等^[9]，其中仍然被较为广泛使用的仿真器为 NS3、OMNET ++与 Riverbed（旧称 OPNET）。

NS 起源于 1989 年，在 S. Keshav 的 REAL 仿真器的基础上，Steve McCanne 等人在劳伦斯伯克利国家实验室(LBNL)开发了 NS 的第一个版本，NS 是一个可扩展、易于配置和编程的事件驱动仿真引擎，并且支持多种类型的 TCP 和路由调度算法。到 1995 年，NS 的第二个版本 NS2 获得了 DARPA(美国国防高级研究计划署)、Xerox PARC(施乐帕克研究中心), UCB(加州大学伯克利分校)和 USC/ISI(南加州大学/信息科学研究所)的支持^[10]，NS2 被广泛应用于学术研究，许多不同的非营利性组织也提供了很多的研究成果。NS3 是 2006 年 7 月 1 日起开始开发的第三代 NS，它由华盛顿大学、佐治亚理工学院和 ICSI 互联网研究中心等机构资助，由法国国家信息与自动化研究所的 Planete 研究组提供合作支持。NS3 旨在取代目前流行的 NS2，然而 NS3 并不是 NS2 的更新版本，NS3 作为一个新的仿真器，它并不后向兼容 NS2，它相比于 NS2 拥有单一的语言、良好的架构、基于底层的开发、广泛支持与良好测试等“先天优势”，NS3 主要特点如下:1.采用合理的 C++设计模式编写，仿真速度快 2.重新审视网络、节点模型与仿真内核的软件架构，各类对象模型更加接近真实网络中的实体 3.与其他软件更好的整合与自身的扩展性 4.支持纯仿真、半仿真和实际网络之间迁移 5.拥有跟踪和统计框架与可视化工具。

总而言之，NS3 是一个不断发展的优秀开源项目，已被广泛应用在第五代移动通信(5G)、物联网、软件定义网络、数据中心等计算机网络的前沿研究领域^[11]。

1.4 国内外研究现状

1.4.1 网络化控制系统的研究现状

如前所述，网络化控制系统通过引入网络为控制系统带来众多的优势，然而也不可避免的引发一些新的问题，其中传感器信息和控制器产生的控制量都需要利用通信信道进行数据的传输，因而由于网络的运行受到自身通信协议、网络负载和网络拓扑等多方面因素带来的影响，会引发网络诱导时延、乱序、丢包、时钟同步等问题^[12]。

网络诱导时延作为最为常见的问题，由于需要传输的信息在网络中会经过网络协议栈自上而下的封包、数据包在收发端之间的传播时延与自下而上的解包三个过程，同时通常通讯网络的带宽是有限的，因而网络节点存在竞争关系，这些都会引发信息传输出现滞后或乱序的情况。丢包的现象可分为被动与主动，当网络出现流量过载，传输超时和传输错误等情况都会导致数据包的被动丢失，而且节点也会根据数据包的时延等特性进行一定策略的主动丢包，这些情况都会使得控制性能降低甚至引发系统不稳定。针对这些问题往往将各类现象建立为相应数学模型，例如线性系统或随机过程以设计控制器。

另外网络节点之间是相互独立的，因而时钟同步问题对网络化控制系统是首要的问题。时钟同步系统的优点在于能根据系统的前向通道与反馈通道的时延设计可靠控制器，而不需考虑节点间时钟异步而导致对于某节点从其他节点接收的信息不可靠的情况。

为提升控制系统性能，研究者们设计了多种控制与通讯相结合的算法，Liu G P^[13]和 Zhao Y B^[14-16]等人在使用通用的预测控制方法的基础上，采用主动补偿的方案有效的降低了通讯约束带给控制系统的影响。在控制和调度算法层面上的研究上，Walsh G C 等人^[17]提出了一种称为“Try-Once-Discard”的动态调度算法，该算法分配网络资源的方式是只允许上一个报告期间内出现最大错误的节点访问网络资源。Branicky M^[18]提出一种将控制器与执行器间的通道直接连接起来的方法，并采用单调速率调度算法对各子系统的传感数据进行调度。为了避免不必要的传输影响信道传输质量，事件触发的控制方案和通信策略被提出，Victor S. Dolk 针对车辆排系统在传感器端采用该策略以实现车辆排的弦稳定控制^[19]。

1.4.2 网络化控制系统仿真平台研究现状

考虑到通信网络的影响，传统控制仿真软件无法胜任网络化控制系统的仿真，需要开展应用于网络化控制系统的仿真技术与仿真平台的研究。网络化控制系统的仿真主要有以下 3 种方式：扩展控制仿真软件以实现通信仿真、扩展通信仿真软件以实现控制系统仿真以及控制、通信仿真软件之间的协同仿真^[6]。

(1) 三种网络化控制系统仿真工具的实现方式

MATLAB 上的 TrueTime 工具箱是通过扩展控制仿真软件来实现通信仿真的。TrueTime 是一个基于 MATLAB/ Simulink 的网络化控制系统仿真工具。TrueTime 有助于在实时内核，网络传输和连续被控对象动态中协同控制器进行仿真任务的执行，其模块库如图 1-3 所示。仿真器的功能包括：

- ①由于代码执行，任务调度和有线/无线网络通信而仿真复杂的控制器时序。
- ②可以将任务写为 M 文件或 C++函数。
- ③多种网络模块（CAN，以太网，TDMA，FDMA，轮询调度网，交换式以太网，PROFINET 和 FlexRay）。
- ④无线网络模块（802.11b WLAN 和 802.15.4 ZigBee）。
- ⑤本地时钟，电池供电设备与动态电压调节^[20]。

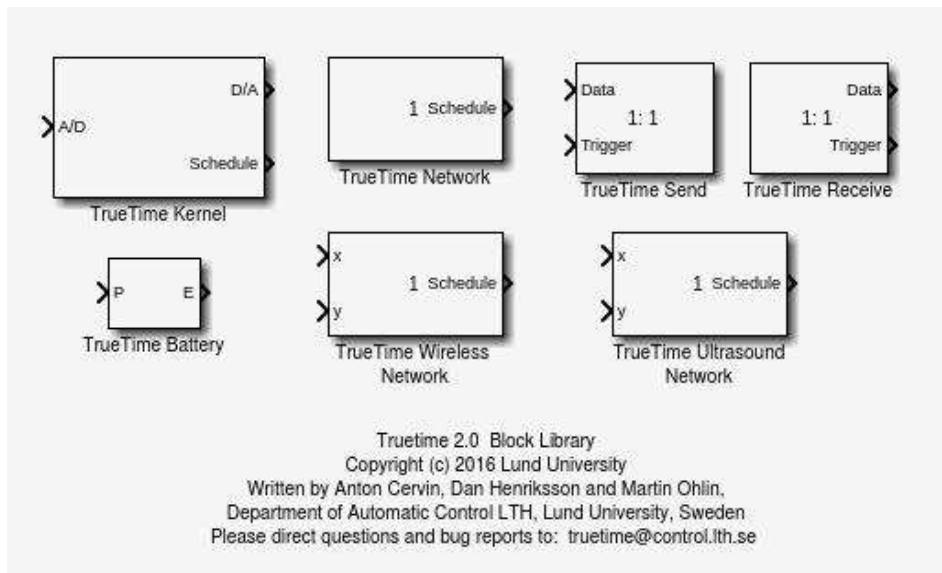


图1-3 TrueTime模块库 [20]
 Figure 1-3. TrueTime module library

可以看出 TrueTime 能完成针对大部分网络化控制系统的仿真，然而 TrueTime 工具箱的网络堆栈只允许有物理层和媒体访问层。这限制了其对其他网络参数的设置，包括高层网络协议（例如路由层，传输层和应用层协议）、地理分布式网络（即 WAN）的拓扑、底层信道、噪声模型以及节点设备的物理特性等更多广泛的网络参数无法得到有效的支持。

通过扩展控制仿真软件来实现通信仿真的实例还有 Prowler^[21]、Andre WSN^[22]、VisualSense^[23]等，然而大部分工具限于基于单仿真器的实现机制，使用局限性大。

扩展通信仿真软件以实现控制系统的仿真在实施时遇到较大的困难，通过在通信仿真软件中实现控制系统的微分代数方程模型来实现^[24]，并在仿真脚本中计算或通过调用外部仿真器（例如 MATLAB）计算来解决。然而除了相对简单的系统

外，大型控制系统通过通信仿真软件建模，特别是那些具有动态特性的系统是耗时、容易出错的任务。虽然该方式不是十分有效但仍然有诸如 Agent/plant^[25]、NSCSPlant^[25]、OPNET extension^[26]、SensorSim^[27]等尝试。

控制、通信仿真软件协同仿真方式由于其结合多种仿真器，克服了以上两种方式的劣势，近年来而受到了广泛的关注，只要在不同类型的仿真器之间建立数据交换机制，就能实现高效而全面的仿真。这些协同仿真平台主要基于离散事件系统规范（DEVS）来实现不同仿真器之间的结合。通过该方式的实施的仿真平台包括：GISOO^[28]、GECO^[29]、Morelli^[30,31]、NMLab^[32]、PiccSIM^[33]、MAPNET^[34]、RoboNetSim^[35]等等，通过结合不同的仿真器，协同仿真架构具体实现的不同，以及建立不同的具体仿真场景与应用，产生多样化的协同仿真工具,由此可见该方式广泛的通用性。

(2) 协同仿真方式下的通信与同步机制

协同仿真本质上是通过对多仿真器之间的数据交换实现的，而不同的仿真器在运行时在计算机操作系统下分别占有不同的进程或资源，因此有必要实现不同仿真器之间的通信机制或资源交换方式。

同时同步机制也是实现协同仿真的关键技术，在多道程序环境下，仿真器进程是并发执行的，不同进程之间存在着不同的相互制约关系。我们把异步环境下的一组并发进程因直接制约而相互发送消息、互相合作、互相等待，使各进程按一定速度运行的过程称为进程间的同步。在网络化控制系统协同仿真中进程同步的问题主要涉及不同仿真器间仿真时间的同步。

1.5 本文的主要研究内容与章节安排

本文结合协同仿真技术，将 NS3 网络仿真器应用到网络化控制系统仿真领域，开发了一种新型网络化控制协同仿真平台。详细的章节安排如下：

第一章是绪论，阐述了本文的研究背景与意义，介绍了网络化控制系统与网络仿真器 NS3 的基本概念，进而介绍了网络化控制系统与网络化控制协同仿真平台的研究现状，并阐明了本文的主要研究内容和章节安排。

第二章介绍了网络化控制系统协同仿真平台总体方案。首先分析了仿真平台的需求，进而介绍仿真平台的整体结构。接着介绍仿真平台中两仿真器需要设计的几大模块，最后，介绍了仿真平台各个子模块间的数据交换方式。

第三章介绍了网络化控制系统协同仿真平台时间同步方案。首先，深入介绍了仿真软件间同步的缘由并提出了 3 种同步方案，并分析 3 种方案的可实现性。其次，详细介绍了利用 TCP 套接字通信实现的同步信息交互模型。

第四章介绍了网络化控制系统协同仿真平台各底层模块的实现。首先，简要介绍了该平台中 NS3 有必要使用的基本类，其次，介绍了 NS3 新类的设计，接着介绍了 MATLAB/Simulink 中模块的实现。

第五章是仿真平台测试和结果分析。介绍了协同仿真平台中搭建的网络化控制系统与补偿方案的测试，以及在 STM32 上进行网络控制算法的测试，进而分析仿真与实验结果并与 TrueTime 仿真结果进行对比。结果表明，仿真平台基本满足了设计要求并具有较好的可靠性与稳定性，为下一步的算法研究和验证奠定基础。

第六章为总结与展望。对本文的工作进行分析与总结，指出了仿真平台在设计方面的优点与不足，并展望了未来的研究方向。

第二章 网络化控制系统协同仿真平台总体方案设计

针对协同仿真平台的总体方案设计，首先有必要考虑合适的系统架构联系各功能模块，该架构需要满足两个软件能够互相通信的要求，另外对于单仿真器内模块间通信和参数的传递也在考虑范围内。基于以上两点，本章提出了基于 NS3 和 MATLAB 的网络化控制系统协同仿真平台的总体方案。

2.1 网络化控制系统协同仿真平台总体方案设计

2.1.1 仿真平台需求分析

国内外学者对网络化控制系统开展了系统的研究，并取得了相当多的成果。然而大部分的研究是通过 MATLAB 数值仿真或者使用 MATLAB 的工具箱 TrueTime 来验证所提出的算法的控制效果。单一的 MATLAB 数值仿真无法仿真实际通信网络的情况，例如网络诱导时延仅由简单的随机数生成器生成，而工具箱 TrueTime 仅仅是被用来仿真网络化控制系统而开发的，由于其开发目的单一性以及依赖 MATLAB 环境的依赖，因而其与 NS3 仿真通信网络的能力是无法比拟的。NS3 作为专业通信网络仿真软件，能够仿真各类通信网络场景，因而将其引入到网络化控制系统仿真将是一个有希望的想法。

本文旨在设计一种基于 NS3 和 MATLAB 的网络化控制系统协同仿真平台，与通过 MATLAB 数值仿真或使用 TrueTime 仿真网络化控制系统相比，其能够验证不同网络化控制算法在更加贴近实际通信网络环境下的仿真效果。然而设计该协同仿真平台也需要解决如下几个问题：

(1) 仿真平台系统架构与数据传递的问题：作为一款仿真工具，应设计一种合理的系统架构将系统中需要使用的各部分联系起来，其次需要考虑系统中各部分数据传递的具体方式，分为在两仿真软件间的数据交互与单个仿真软件内的数据传递。

(2) 仿真平台底层模块实现与软件间时序同步的问题：仿真平台底层模块实现是平台实现的基础与运行的依赖，因此对底层各模块需要进行程序设计。而仿真软件间时间的不同步将会直接影响到协同仿真结果的可信度，需要根据仿真器驱动方式的不同和仿真目标的不同实施相应的同步方式。因此实现两仿真软件间的时序同步至关重要。

(3) 平台的灵活全面与使用便捷问题：网络化控制系统的是一个广泛而多样化

的问题，其涉及到实际系统中的众多应用，因此该平台需要满足绝大部分网络化控制系统的仿真需求，同时平台使用者能够通过相应界面较为容易地通过该平台搭建仿真场景。

总之，本文提出的基于 NS3 和 MATLAB 的网络化控制系统协同仿真平台总体解决方案需要实现以下三部分：仿真平台的底层逻辑、有效地仿真多种网络化控制系统与用户交互界面。

2.1.2 仿真平台依赖环境配置

NS3 的主要运行依赖操作系统为 Linux，Ubuntu 16.04LTS 被选为该平台运行的操作系统，因其容易安装必要的应用程序，编程环境，且拥有良好的可视化界面。

对于这两种仿真软件的版本，由于 MATLAB 对 Linux 操作系统的兼容性较差，为兼顾 MATLAB 在 Ubuntu 上的运行速度，选用了较稳定的 MATLAB 2011b UNIX 版本，而 NS3 为最新的 3.27 版本；另外需要安装 C++ 版本的 Eclipse 并对 NS3 进行配置；由于 NS3 本质为一个 C++ 库，所以为了能够方便地对 NS3 中的程序进行修改，安装了 C++ 版本的 Eclipse CDT，并对 NS3 进行相关配置，配置的具体步骤如下：

(1) 在 Eclipse 中新建 C++ 的空工程项目，将 ns-3.27 文件夹拷贝到这个工程下面，刷新项目并需在 Eclipse 终端中运行 NS3 配置指令 `./waf configure`；

(2) 设置该工程，右键属性(properties)，在 build 中(C++ Build)进行修改，选择 builder 类型为 External builder，build 命令选择编译 NS3 程序的 waf 编译器所在的路径，比如：`${workspace_loc:/ns3/ns-3.27/waf}`，其中：`workspace_loc` 是指工程所在的路径；build 的目录就选择 NS3 对应的目录，比如：`${workspace_loc:/ns3/ns-3.27/build}`；

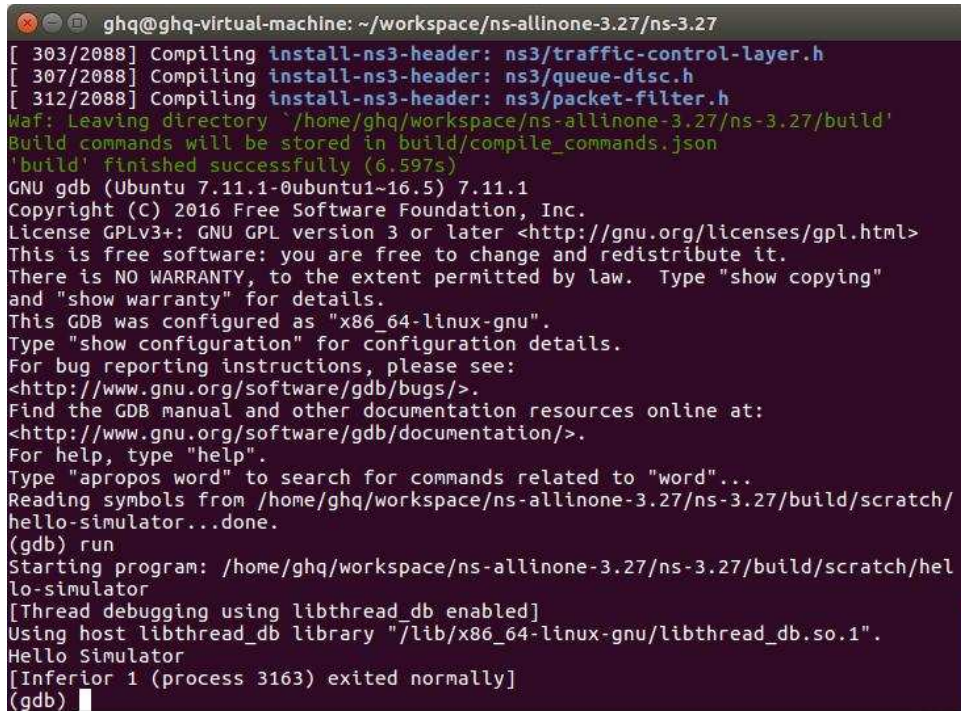
(3) 外部编译器的配置，就是 NS3 使用的 waf 编译器的配置，在运行--外部工具--外部工具配置(run-external tools-external tools configurations)中，位置(location)是 waf 的路径，比如：`${workspace_loc:/ns3/ns-3.27/waf}`，工作目录(working directory)是 NS3 所在的目录，比如：`${workspace_loc:/ns3/ns-3.27}`，自变量(arguments)，填：`--run "${string_prompt}"`，string_prompt 为输入的参数，即用户执行的 C++ 文件；

4. 调试：NS3 可以使用 Eclipse 调试，也可使用终端调用 gdb 进行调试，在 Eclipse 中调试步骤为：右键工程，选择属性(properties)，选择调试的配置(run/debug settings)，创建一个新的加载配置，选择你刚刚生成的文件 project，选择环境设置栏(environment)，新建一个环境变量，变量名(name)为：`LD_LIBRARY_PATH`，变量值(value)为 build 的路径：`${workspace_loc:/ns3/ns-3.27/build}`，之后选择 Main 栏，在 C/C++ Application 中填写 build 文件夹下需要调试的已通过编译器 waf 编译

的目标文件，比如：build/scratch/third。另外也可以通过 gdb 来调试 NS3 程序，在 ns-3.27 文件夹中打开终端，输入指令比如：

```
$ ./waf --run=hello-simulator --command-template="gdb %s --args <args>",
```

如图 2-1 所示，执行后将会以调试状态运行 hello-simulator 程序，使用 run 或者 r 命令开始程序的执行；



```
ghq@ghq-virtual-machine: ~/workspace/ns-allinone-3.27/ns-3.27
[ 303/2088] Compiling install-ns3-header: ns3/traffic-control-layer.h
[ 307/2088] Compiling install-ns3-header: ns3/queue-disc.h
[ 312/2088] Compiling install-ns3-header: ns3/packet-filter.h
Waf: Leaving directory `/home/ghq/workspace/ns-allinone-3.27/ns-3.27/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (6.597s)
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from /home/ghq/workspace/ns-allinone-3.27/ns-3.27/build/scratch/
hello-simulator...done.
(gdb) run
Starting program: /home/ghq/workspace/ns-allinone-3.27/ns-3.27/build/scratch/hel
lo-simulator
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Hello Simulator
[Inferior 1 (process 3163) exited normally]
(gdb) █
```

图2-1 调试状态运行hello-simulator

Figure 2-1. Debug state running hello-simulator

2.2 协同仿真平台架构与模块

2.2.1 网络化控制系统协同仿真平台架构

仿真平台系统从架构上可划分为控制层、通讯层和模型层三部分，其架构如图 2-2 所示。控制层为用户交互界面，负责显示平台的仿真结果，实现系统参数、网络参数的设置，下发仿真控制命令以及接收仿真模型中生成的数据信息；通讯层主要负责通过不同的手段实现 MATLAB 与 NS3 之间数据的交换；模型层则使用接收到的仿真模型控制指令与仿真模型运行所需的数据作为输入，此外两种仿真器分别调用各自的仿真驱动控制模块推进仿真过程，并建立两仿真器间的时间同步。

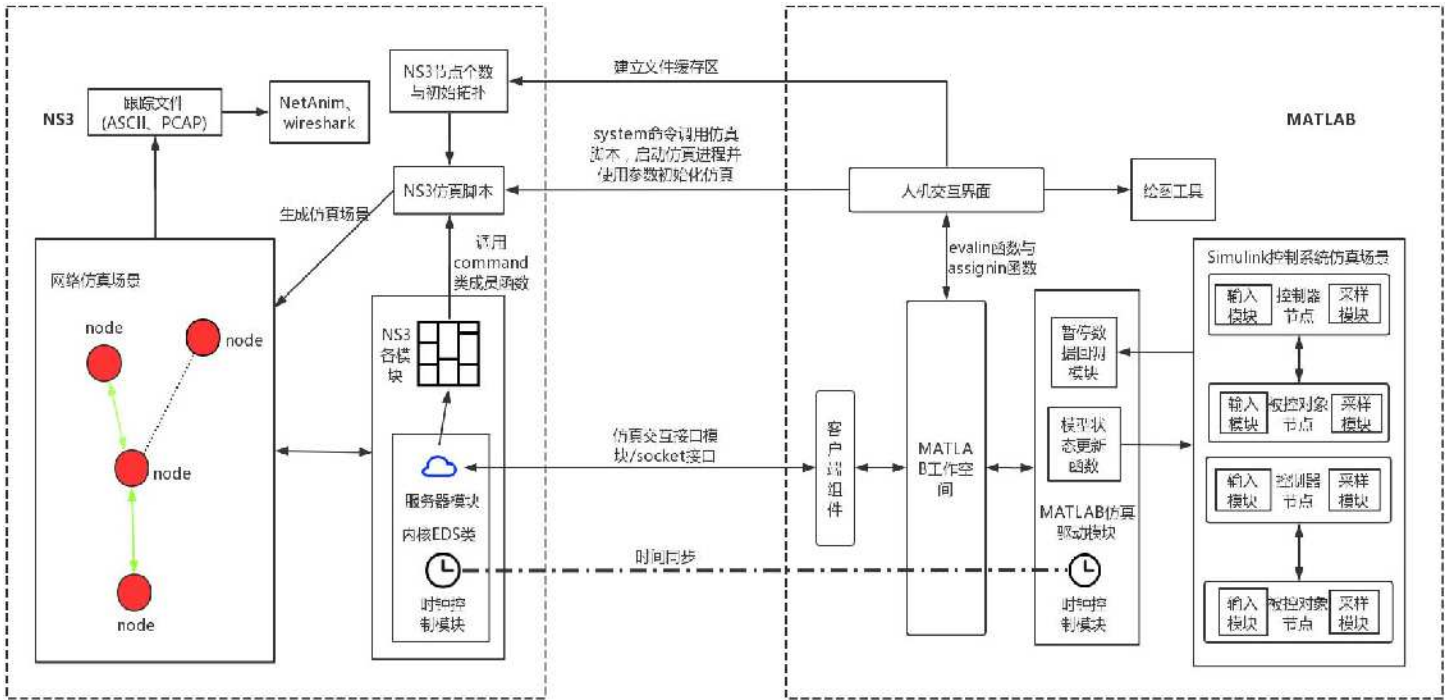


图2-2 基于NS3和MATLAB的网络化控制系统协同仿真平台架构

Figure 2-2. Networked Control System Co-simulation Platform Architecture Based on NS3 and MATLAB

2.2.2 NS3 模块

NS3 软件组织架构简图如图 2-3 所示，最下层的 core 模块定义了仿真软件的核心功能，第二层 network 模块定义了网络基本要素与模型的抽象基类，第三层 internet 模块和 mobility 模块定义了网络层模型和节点移动模型，第四层包括应用层模型的实现，第五层为助手类，它使用户能更方便地编写仿真场景程序，最上层就是用户仿真脚本与各功能模块的测试脚本。NS3 支持修改与扩展各模块的源代码。

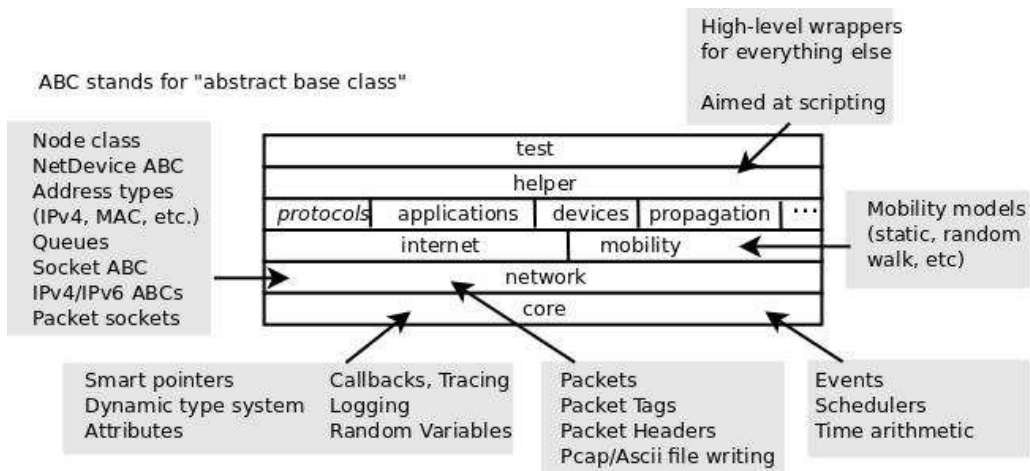


图 2-3 NS3 软件组织架构简图 [11]

Figure 2-3. NS3 software organization architecture diagram

本平台中调用并修改的 NS3 模块主要包括了 core 模块、NS3 仿真脚本、NS3 节点拓扑模块、application 模块、wifi 模块、LTE 模块、节点移动模型模块等，以满足仿真平台全面仿真网络化控制系统的目标。

2.2.3 MATLAB 模块

MATLAB 不像 NS3 由开源库构成的，用户虽无法修改 MATLAB 底层实现，但 MATLAB 支持用户自己编写函数与模块实现新的功能，以及开发出新的仿真工具。

本平台中 MATLAB 的主要模块包括 MATLAB 仿真驱动模块、Simulink 控制系统模型、MATLAB 客户端组件、仿真平台交互界面、MATLAB 基础工作空间、仿真跟踪统计与可视化模块。

2.3 协同仿真平台数据交换与传递设计

2.3.1 仿真软件间的数据交换

首先需要考虑仿真软件间的数据交换，根据 MATLAB 和 NS3 的对外交互特性确定仿真平台的主导软件，协同仿真的实现方案可分为 3 种：

(1) 以 NS3 为主导软件，通过启用一个新的 MATLAB 引擎进程在后台运行，实现与 MATLAB 间的交互；

(2) 以 MATLAB 为主导软件，并将 NS3 仿真模型封装为“函数”，MATLAB 端通过调用该“函数”，即可实现协同仿真；

(3) 以第三方软件为主导，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) 支持多仿真器间协同构建与运行，每个仿真器仿真一个复杂系统的某一方面。

本平台将采用方式(2)实现，即将 MATLAB 作主导软件，但具体交互方案的实施将不通过 NS3 仿真模型封装为“函数”实现，而是利用套接字实现两仿真器进程协同仿真时的进程间通信。

2.3.2 单仿真器内的数据传递

仿真软件间的数据交互是两个仿真软件协同仿真的基础，而单仿真器内的数据传递对于整个仿真平台的实现也具有重要的辅助作用，单仿真器内的数据传递主要包括：

(1) MATLAB 仿真软件内：

①仿真平台交互界面中的参数与 MATLAB 基础工作空间的信息交换。

②MATLAB 中客户端组件与 MATLAB 基础工作空间中信息的交换^[41]。

③Simulink 与 MATLAB 工作空间的信息交换。

(2) NS3 仿真软件内:

- ①NS3 提供的一些固有 API 函数用于数据传递。
- ②使用 C++编写新的 API 函数或者利用 C++的语言特性。

2.4 本章小结

本章首先分析了网络化控制系统协同仿真平台的需求, 并提出将 NS3 和 MATLAB 作为该仿真平台的依赖软件, 设计了基于 NS3 和 MATLAB 的网络化控制系统协同仿真平台总体解决方案, 接着介绍了仿真平台依赖环境的配置, 包括操作系统和仿真软件的版本及 NS3 在 Eclipse 中的配置方法。然后设计了仿真平台的软件系统架构并介绍了相关模块, 之后设计了仿真平台内数据传递的方案, 包括仿真软件间的数据交互与单仿真软件内的数据传递两个方面。

第三章 网络化控制系统协同仿真平台时间同步方案设计

本章将引入协同仿真中时间同步的概念，时间同步方案的设计是协同仿真平台实现的难点。由于 NS3 与 MATLAB 具有不同的驱动方式，因此设计了三种同步方案，之后将介绍使用套接字实现进程间通信的方法，以及利用进程间通信实现同步信息交换的具体模型。

3.1 时间同步策略

协同仿真系统是一种对实时性和时序严格要求的仿真系统，控制工程师定义控制功能和受控系统的模型，并根据与网络仿真器相互的同步反应（SR）（同步反应指仿真器间在指定时间点相互交换数据的过程）进行验证，其中所有的控制计算和通信仿真都假定在两个事件间的逻辑时间内完成，仿真进程根据同步假设所规定的时间计算，调度来推进协同仿真执行，执行完成后进行同步反应。

3.1.1 MATLAB/Simulink 与 NS3 的仿真驱动方式

首先需要注意到这两类仿真器的仿真驱动的不同：控制仿真器采用离散时间驱动，而网络仿真器采用离散事件驱动，本章中介绍的时间同步方案是指将这两种不同驱动方式的仿真器统一在一致的时间框架下，以避免出现时间不同步引发的系统状态不同步，进而导致仿真结果失效的情况。在不同仿真器中仿真同步事件的视角与定义将有所不同，因而会采用不同方式应对同步事件。

(1) MATLAB/Simulink 驱动方式

MATLAB 中的 Simulink 支持多种驱动方式，虽然其支持在仿真中插入离散事件但大部分仿真都采用离散时间驱动推进仿真，Simulink 在仿真开始到结束的每个时间点求解模型，如图 3-1 所示。在离散时间驱动方式下，仿真同步事件是平台运行时控制系统节点触发的一次周期发送事件或控制系统节点的某次数据包接收事件，控制系统执行仿真同步事件可以看做使用求解器求解在该事件发生时间点的控制系统模型，之后进行一次同步反应。

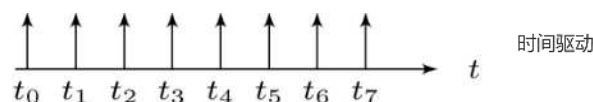


图 3-1 离散时间驱动方式

Figure 3-1. Discrete time driven

(2) NS3 驱动方式

NS3 则采用离散事件驱动推进仿真,如图 3-2 所示。它实现一个事件处理机制,使用一个队列保存一系列事件。事件可以异步或周期性地到达,并在事件队列中按照事件发生时间或因果顺序排序,调度器根据优先级调度算法从当前就绪队列中分派当前活动任务。在该种驱动方式下,仿真同步事件是控制系统节点触发的周期发送前的网络事件或控制系统节点的某次数据包接收事件。

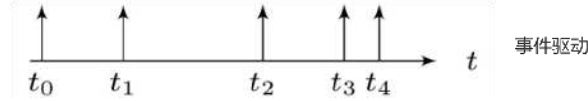


图3-2 离散事件驱动方式

Figure 3-2. Discrete event driven

3.1.2 协同仿真时间同步方案

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,有三种不同的同步方案:时间步进、主从式和全局事件驱动同步^[6]。

(1) 时间步进同步:该同步方式下,两仿真器都以控制系统节点的周期发送事件作为周期事件,并将相邻发送事件间的时间长度作为进行一次同步的周期步长,在每个周期步长内各仿真器独立运行仿真,仿真器在周期事件时间点暂停仿真,并相互交换信息的作为一次同步。然而,如果某些系统事件发生在同步点之间,则事件必须被缓存,直到下一个同步点被处理。因此如果应用程序是时间关键的,这可能会累积系统误差并妨碍仿真保真度,该同步方式通常使用在对仿真误差容错率较高的场景中,该方式运行机制如图 3-3 所示。

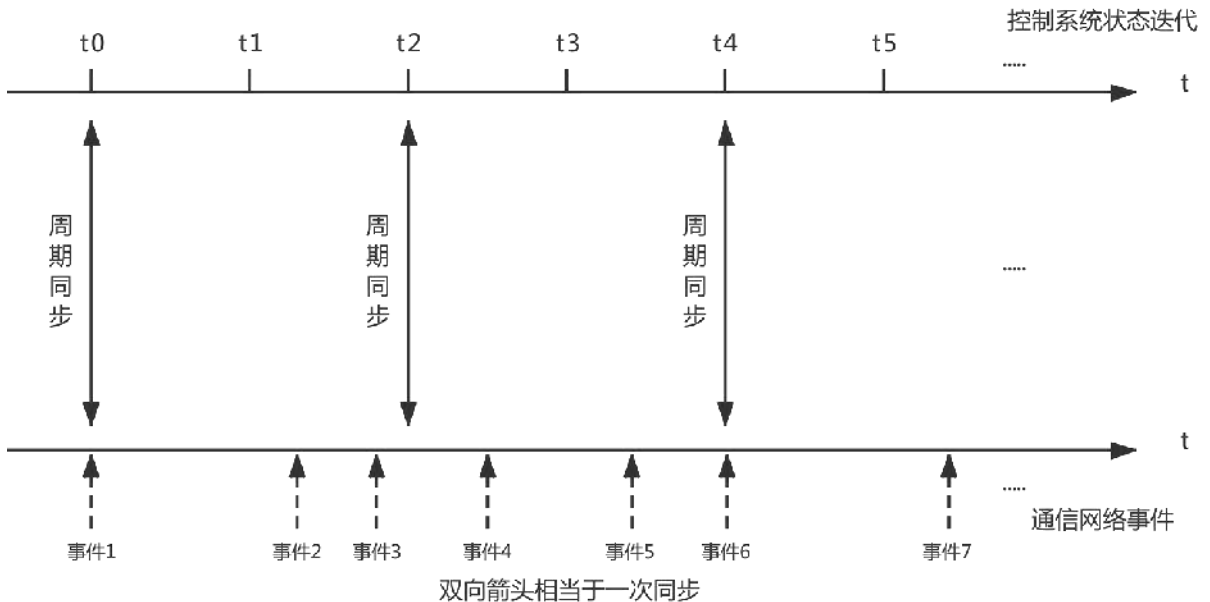


图3-3 时间步进同步运行机制示意图

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

(2) 主从式同步:为提高仿真在某些场景中的精确度,主从式同步改进了时间

步进同步方式，该同步方式下的周期事件和周期步长与时间步进式一致，然而在每个周期步长内各仿真器不再独立运行，而是让网络仿真器先进行一个周期的仿真，并记录这个周期内对控制系统仿真有影响的网络事件，当网络仿真器暂停在周期事件时间点后再推进控制系统仿真器仿真，并在推进仿真的过程中复现这些对控制系统仿真有影响的网络事件在控制系统的执行过程，之后控制系统仿真器暂停在周期事件时间点，两仿真器相互交换信息进行同步反应。该同步方式考虑了周期步长内异步网络事件对控制系统的影响，因而不会累积系统误差，然而该方式没有考虑到控制系统对异步网络事件的及时响应，控制系统仅能在周期事件时间点周期发送控制或传感信息，该方式运行机制如图 3-4 所示。

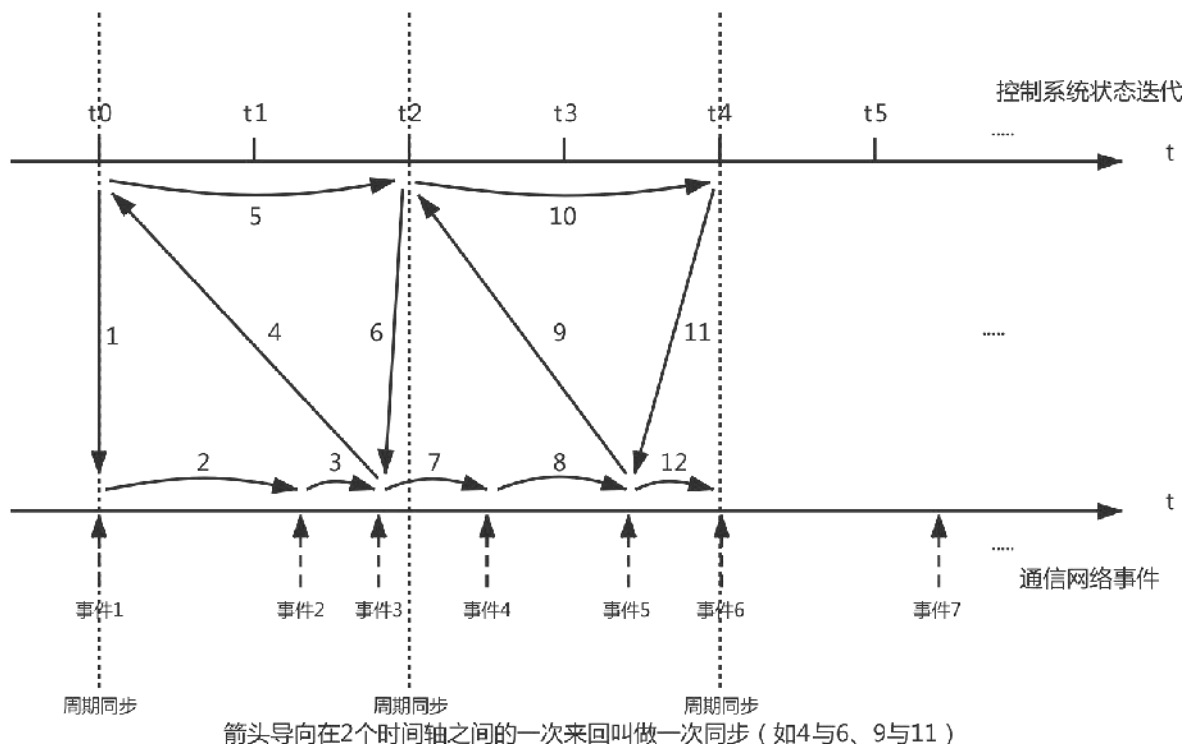


图3-4 主从式同步运行机制示意图

Figure 3-4. Schematic diagram of master-slave synchronous operation mechanism

(3) 全局事件驱动同步：为了让控制系统对非周期的异步网络事件进行及时响应，比如控制系统节点的数据包接收事件，在主从式同步的基础上改进提出了全局事件驱动同步方式。每当网络仿真器在周期步长内遇到异步网络事件，且该异步网络事件是对控制系统有影响的异步事件，则网络仿真器在该异步事件时间点暂停仿真，并推进控制系统仿真器的仿真进程到该异步事件时间点并暂停，之后控制系统仿真时会判断是否需要对该异步事件进行响应，并将响应结果作为交换信息的一部分进行同步反应。该方式既没有累积系统误差又能使控制系统及时响应某些非周期的异步网络事件，然而该方式在周期事件时间点的基础上增加了异

步事件时间点，仿真器间同步次数的增多导致整体协同仿真速度相当有限，该方式运行机制如图 3-5 所示。

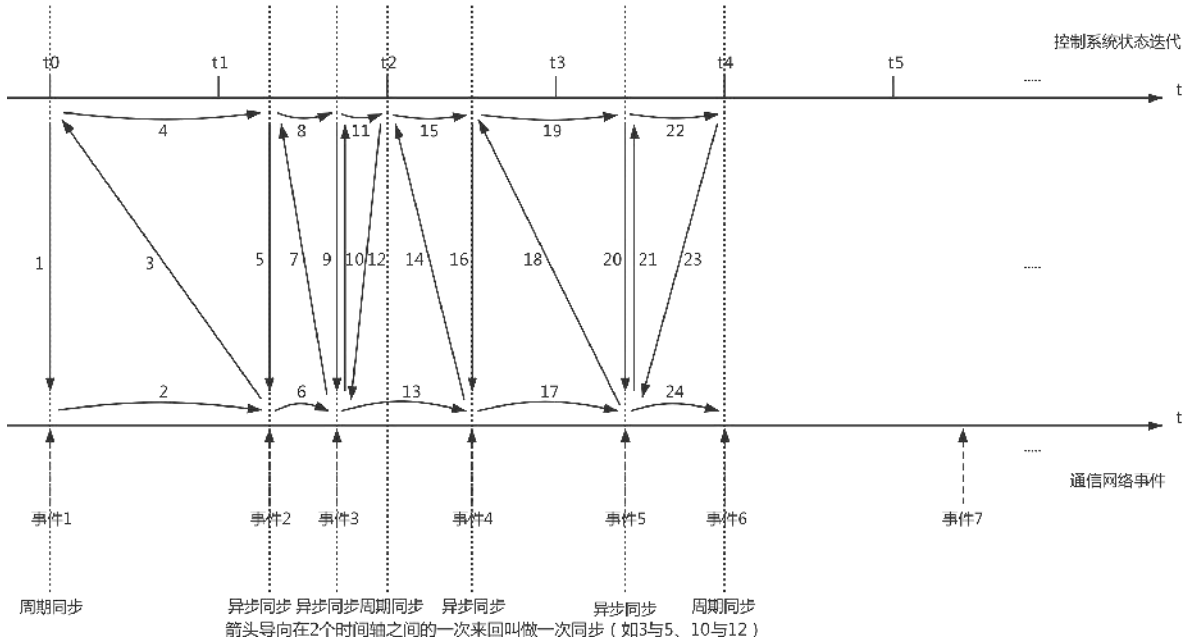


图3-5 全局事件驱动同步运行机制示意图

Figure 3-5. Schematic diagram of global event-driven synchronous operation mechanism

3.1.3 同步方案的选择

由于本仿真平台需要实现对网络化控制系统的仿真功能，而网络化控制系统是一种对数据包时延相当敏感的控制系統，如果采用时间步进同步会导致仿真中数据包事件的延误处理，相当于变相增加了数据包延迟，进而累积系统误差影响仿真的精度，因而本平台不使用第一种同步方式。

控制系统的节点按任务调用方案被划分为传感器节点、控制器节点和执行器节点，节点的驱动方式一般分为时间驱动和事件驱动 2 种，时间驱动方式为节点按一定周期采样并发送数据包，而事件驱动下数据的采样和发送都是由非周期事件触发的，这会影响系统中各节点间的同步方案与数据包传输延迟特性，详情如下表 3-1 所示。

在本仿真工具中将某一控制系统的传感器与执行器看作依赖被控对象的同一节点，而控制器看作是另一节点。关于各节点的驱动方式，默认传感器为时间驱动、执行器为事件驱动，控制器则对两种方式都予以支持，因此该平台能仿真上表中的 2 行与 3 行的情况。而两仿真器间的主从式同步和全局事件驱动同步分别能够实现对控制器时间驱动和控制器事件驱动两种驱动方式的仿真，因此在本仿真平台中采用了后两种同步方案并予以实施。

表 3-1 不同类型节点的驱动方式对系统同步和时延的影响

Table 3-1. Influence of different types of node driving methods on system synchronization and

delay				
传感器	控制器	执行器	时钟同步	整个控制回路时延特性
时间驱动	事件驱动	时间驱动	不需要	同步时间的倍数
时间驱动	事件驱动	事件驱动	不需要	变化
时间驱动	时间驱动	事件驱动	需要	变化
时间驱动	时间驱动	时间驱动	需要	同步时间的倍数

3.2 Linux 下的 Socket 编程

在该仿真平台中有多处涉及到套接字的使用，仿真软件间的同步反应通过 TCP 协议实现，而 NS3 中的节点间数据传输使用仿真的 linux TCP 与 UDP 协议栈实现，因此有必要简要介绍套接字编程。

套接字是使用标准系统文件描述符与其它程序通信的一种方式^[42,43]，在基于 C / S 的网络通信模型广泛使用，可用于本地网络或互联网上的计算机间的交互。

Linux 中 C / S 程序编写的基本步骤如下：

TCP 客户端：创建套接字→绑定套接字→与远程服务程序建立会话→读 / 写数据→终止连接。

TCP 服务器：创建套接字→绑定套接字→设置套接字为监听模式，进入被动接受连接请求状态→接受请求，建立连接→读 / 写数据→终止连接。

而 UDP 客户端的编写不需要建立会话，服务器的编写不需要设置套接字为监听模式。

3.3 同步信息交互模型

之前介绍的同步反应涉及到仿真过程中两仿真器间的数据交换，要实现该抽象过程就需要设计同步信息交互模型，交互模型采用 MATLAB 作为仿真的主导软件，利用套接字实现两仿真进程运行协同仿真时的进程间通信，因而分别在 MATLAB 与 NS3 中编写客户端和服务端程序^[44]。先在这两个程序中分别创建新的套接字，套接字都采用阻塞模式使发送数据后仿真能够阻塞，之后初始化 TCP 连接开始数据交换与同步。

仿真开始运行时，MATLAB 初始化并先产生一个通知结构体，该结构体包含每个节点被给予的指令、MATLAB 仿真结果等信息，MATLAB 客户端使用套接字将该结构体发送给 NS3，然后阻塞 MATLAB 仿真并等待从 NS3 发送的包含 NS3 仿真结果的通知。

之后轮到 NS3 执行仿真，其接收到通知结构体后开始在调度器中计划数据包发送事件。然后 NS3 运行所有预定的事件，直到仿真在下一个 NS3 同步事件时间点暂停，此时 NS3 会创建一个包含接收的数据包等信息的通知结构体，NS3 服务器使用套接字将该通知结构体发送给 MATLAB 客户端，之后 NS3 进入阻塞模式并等待下一个通知，而 MATLAB 在收到该通知后会进行相应处理，并再次运行一个仿真步长至与 NS3 同步事件时间点对应的 MATLAB 同步事件时间点，这样一轮时间同步就完成了。

这样的交互将会在仿真过程中不断发生，值得注意的是如果 MATLAB 仅仅运行客户端作为外部驱动，而不仿真自身的 Simulink 模型，那么 MATLAB 与 NS3 间的数据交换仅使用在开始仿真时建立的 TCP 连接，而如果需要仿真 Simulink 模型，即需跳出客户端程序运行 Simulink 模型，则采用客户端不断建立 TCP 连接又关闭连接的方式进行交互。此时 NS3 的服务器将作为迭代服务器存在，主套接字用于侦听客户端的连接请求，主套接字将不断创建关闭子套接字，子套接字用于接收新的连接请求并发送与接收通知结构体，如图 3-6 所示为同步信息交互模型。

3.4 本章小结

本章详细介绍了该网络化控制系统协同仿真平台时间同步方案的设计。首先，由于协同仿真系统对实时性和时序的严格要求，引入了时间同步的重要概念。其次，介绍了 NS3 与 MATLAB 仿真驱动方式的不同，MATLAB 为离散时间驱动而 NS3 为离散事件驱动。接着提出了三种不同的同步方案：时间步进、主从式和全局事件驱动同步，并依据仿真网络化控制系统的目标场景，采用了后两种同步方案。之后介绍了套接字编程与实现同步信息交换的具体模型。

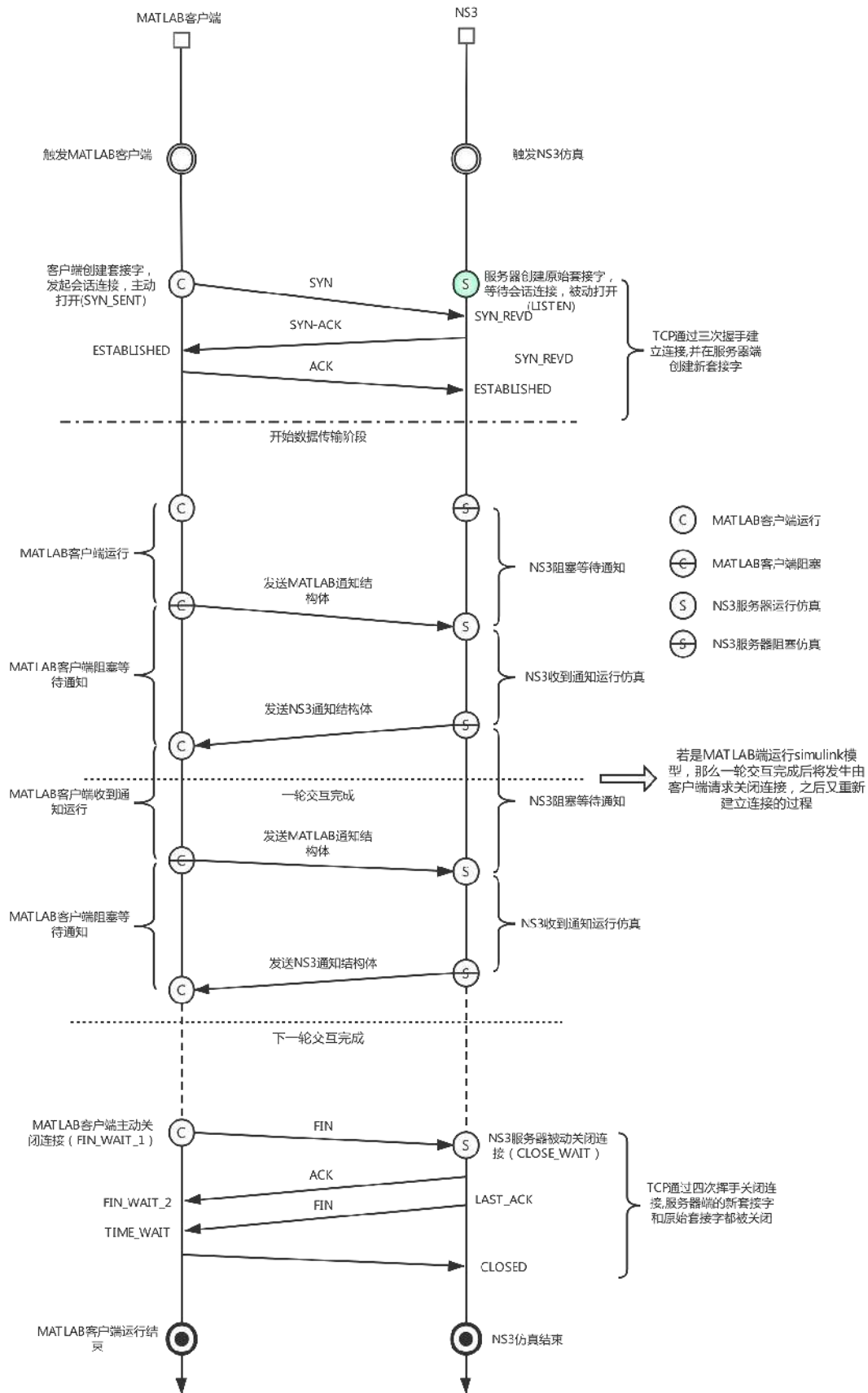


图3-6 同步信息交互模型运行机制示意图

Figure 3-6. Diagram of operation mechanism of synchronous information interaction model

第四章 网络化控制系统协同仿真平台底层模块设计

本章将根据第二章中介绍的仿真平台底层模块的具体功能展开设计。介绍了一些新的 NS3 模块的设计，这些类包括外部驱动仿真器类、配套的辅助类以及仿真脚本，同时开发了适用于协同仿真的 MATLAB 模块。本章将穿插介绍关于两种时间同步方式的具体设计思想与实现。

4.1 NS3 各模块程序设计

4.1.1 基本类的描述

下面描述的类在本平台中被考虑使用。

(1) Ptr 类-智能指针类

通常在 NS3 中使用智能指针，但有时也使用典型的 C++ 指针，知道如何在它们之间进行转换是很有用的，可借助 `GetPointer` 和 `PeekPointer` 方法进行转换。

(2) EventId 类

每个 `EventId` 标识一个独特的事件,这些事件通过 `Simulator::Schedule` 方法进行调度，`EventId` 在它预定了 `Simulator` 类的 `Cancel` 和 `Remove` 方法后可取消或删除该事件。

(3) Application 类

该类用作大多数 NS3 应用程序的基类。该类包括了常用的应用程序方法，如：`StartApplication`，`StopApplication` 等，应用程序与单个节点相关联，每个节点都包含一个指向其应用程序的智能指针列表。

(4) TypeId 类

`TypeId` 类是 NS3 中各个对象模型元信息的管理接口，在每个对象模型类中都有一个 `GetTypeId` 静态函数。该函数的作用就是为该类创建 `TypeId` 对象，相当于将对象模型基本信息注册到 NS3 中。

4.1.2 外部驱动仿真器类（EDS 类）程序设计

NS3 的默认仿真器没有提供协同仿真功能，这就需要设计一个新的仿真器类，使其适用于协同仿真。

通过继承 `DefaultSimulatorImpl` 类创建新的仿真器类 `ExternallyDrivenSim`，之后将会在新类中增添和修改相应的辅助函数，他们将会有助于新仿真器类的实现。

`ExternallyDrivenSim` 类用来在仿真脚本中创建外部驱动仿真器对象，其具有的

数据成员与成员函数如 UML 类图 4-1 所示，这里介绍该类的成员函数：



图4-1 ExternallyDrivenSim类UML类图

Figure 4-1. ExternallyDrivenSim class UML class diagram

(1) ExternallyDrivenSim 类构造函数

该函数将会创建一个用于仿真的外部驱动仿真器对象，在构造函数内新建了一个基于 TCP 协议的服务器。

(2) Listen 函数

该函数定义服务器逻辑为一个迭代服务器，迭代服务器依次处理客户端的轮询连接，接收并解析 MATLAB 发送的通知结构体中的数据，并使用 RunScheduleTransmitTo 函数计划节点数据包发送事件。

(3) GetNotices 函数

该函数收集服务器的运行结果，以此生成即将发送给 MATLAB 的数据包通知

结构体。

(4) Run 函数

该函数调用 ProcessOneEvent 函数运行所有被计划的事件，并通过使用仿真时间控制模块推进仿真，该模块允许仿真在每次给出的时间上限暂停。通过将现在的仿真事件时间和在类中定义的私有属性 m_TimeLimit 比较，只要该仿真事件时间小于等于 m_TimeLimit，那么将会调用私有函数 ProcessOneEvent 推进该仿真事件，如果该仿真事件的时间大于 m_TimeLimit，会把 m_TimeLimit 增加周期步长的时间来更新 m_TimeLimit 的时间上限，之后与 MATLAB 客户端进行同步交互，调用 TransmitNotices 函数发送给 MATLAB 该段仿真的结果，并调用 Listen 函数从 MATLAB 接收下一段仿真的节点数据包发送计划，之后再运行新计划的仿真事件，通过不断重复该过程，使得仿真时间以及仿真事件受到一定控制的往前推进。如图 4-2 所示为时间控制模块的运行流程图。

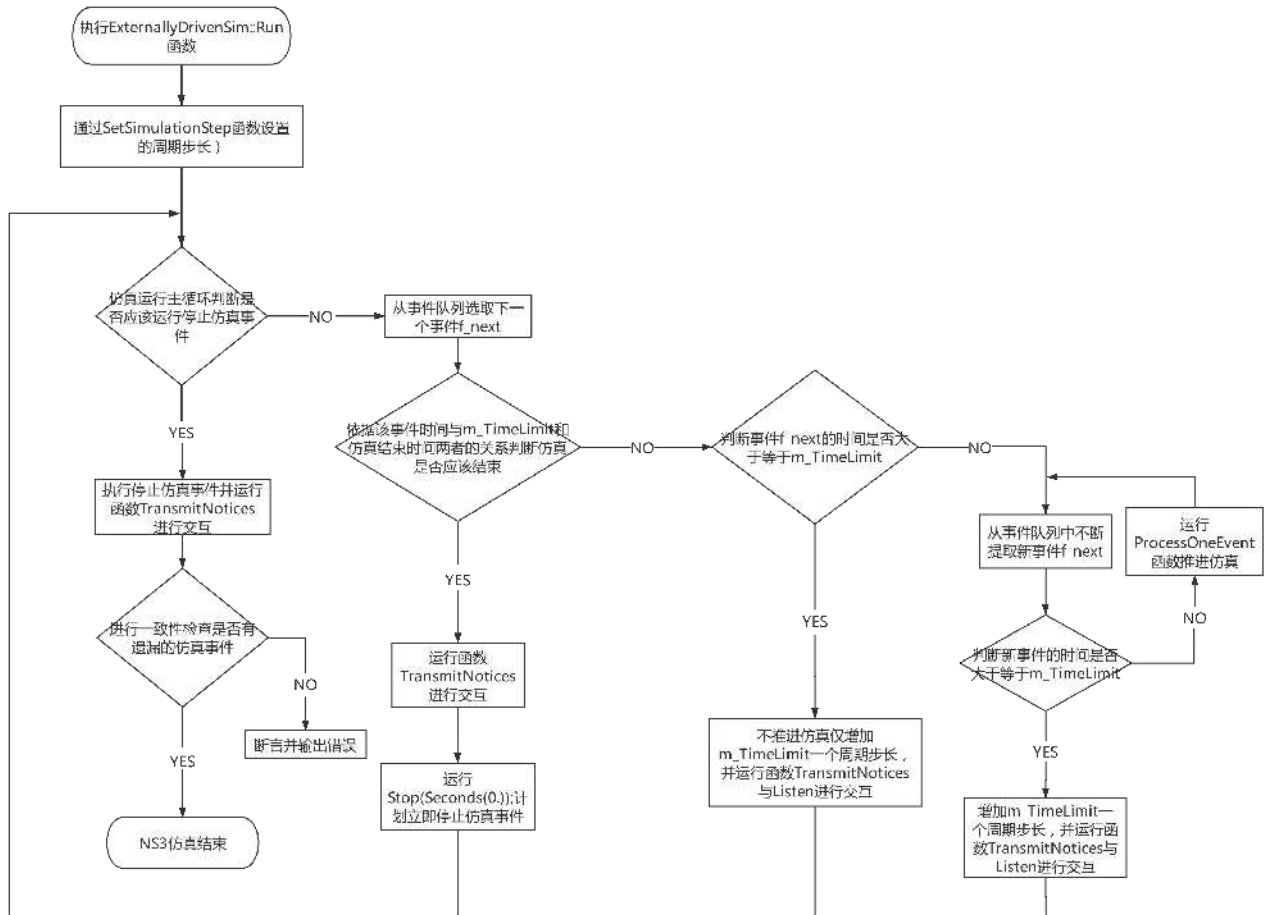


图4-2 时间控制模块运行流程图

Figure 4-2. Time control module operation flow chart

(5) trans_noti_insam 函数

该函数在在事件驱动模式下被调用，在一个仿真周期步长内部，当服务器接

收到数据包时设置异步仿真事件标志，而异步仿真事件标志用于触发同步反应。

4.1.3 辅助类的设计

设计的辅助类主要包括以下几个：

(1) UDP 客户端应用类：该类是适配外部驱动仿真器的 UDP 客户端，它将发送从外部驱动仿真器获得的数据内容，将其封装为一个数据包类并发送给指定的目标节点服务器。同时当用户希望指定发送数据包的延迟和丢包率时，该类的对象将跳过网络的底层实现，直接按给定延迟与丢包率将数据包发送给同一层的服务器，该类的 UML 类图如 4-3 所示。

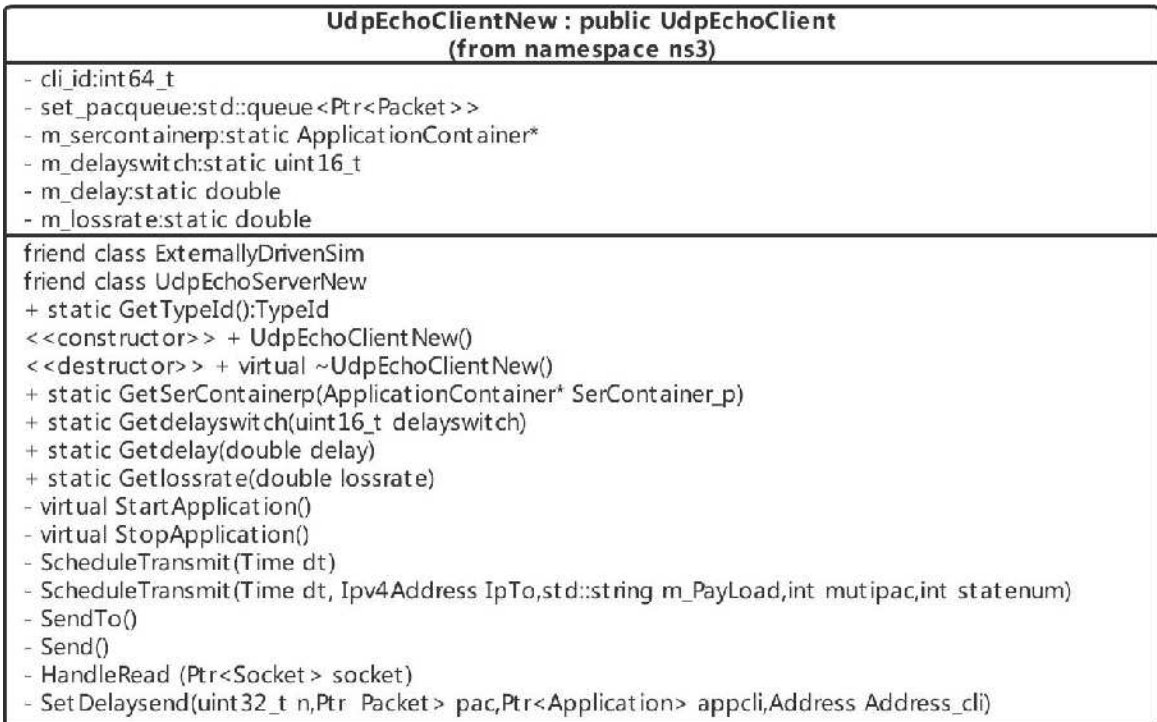


图4-3 UDP客户端应用类UML类图

Figure 4-3. UDP client application class UML class diagram

(2) UDP 服务器应用类：该类是适配外部驱动仿真器的 UDP 服务器，它将接收客户端发送的数据包，同时调用 GetNotices 函数生成需要发送给 MATLAB 的通知。在事件驱动模式下，该类通过成员函数 Schedule_Asyn_event 计划一个异步事件以实现与 MATLAB 的交互，同时该模式下计算的数据包延迟分为两类：真实延迟和“推算”延迟，真实延迟即是数据包收发时间之差，而“推算”延迟则将数据包的接收时间减去该数据包发送事件前的周期发送事件时间点作为该数据包的延迟。这是为了在 MATLAB 中能更简单地推算出该数据包的接收时间，进而推进 MATLAB 的仿真，该类的 UML 类图如 4-4 所示。

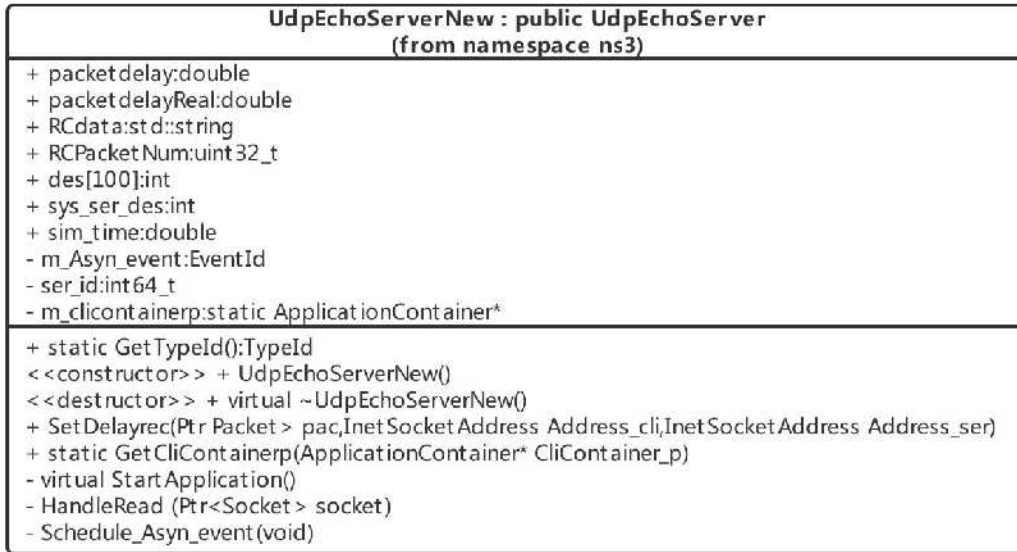


图4-4 UDP服务器应用类UML类图

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

(3) TCP 客户端应用类：该类是适配外部驱动仿真器的 TCP 客户端，其功能类似于 UDP 客户端应用类，不同的是其将 TCP 套接字的段大小属性设置为即将要发送的数据包的大小，该类的 UML 类图如 4-5 所示。

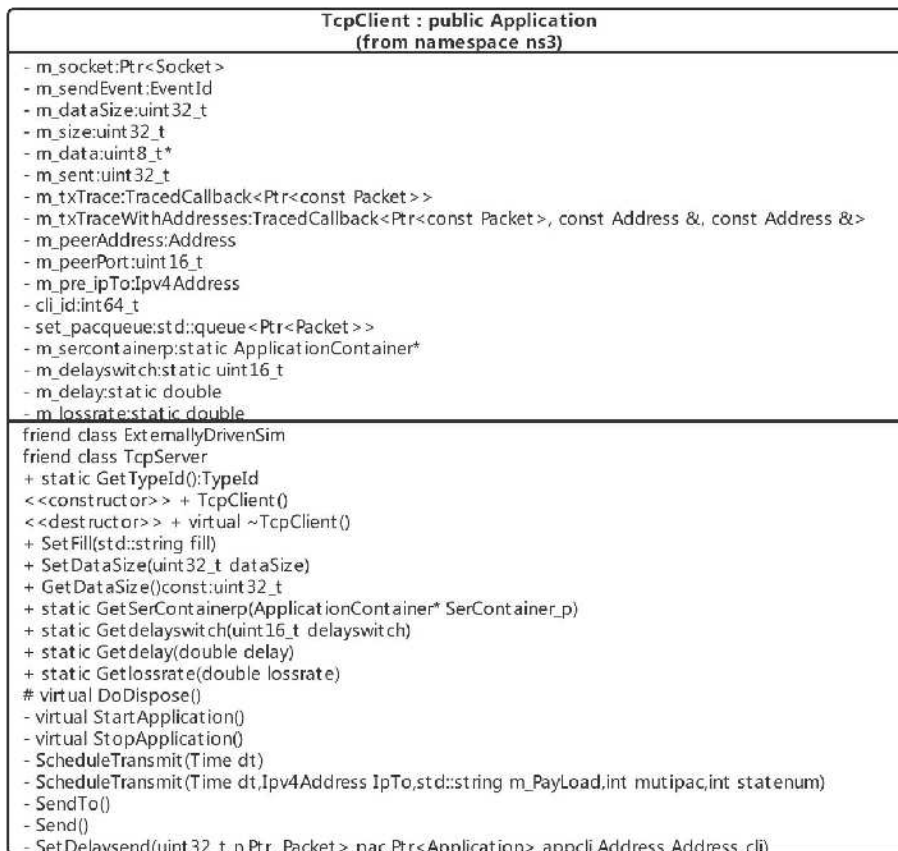


图4-5 TCP客户端应用类UML类图

Figure 4-5. TCP client application class UML class diagram

(4) TCP 服务器应用类：该类是适配外部驱动仿真器的 TCP 服务器，其功能类似于 UDP 服务器应用类，只是多了 TCP 特有的侦听、接收与断开会话连接的功能，该类的 UML 类图如 4-6 所示。



图4-6 TCP服务器应用类UML类图

Figure 4-6. TCP server application class UML class diagram

(5) 应用程序安装助手类：这些类是以上各个应用程序在仿真脚本中的安装助手类，它们具有的数据成员与成员函数如 UML 类图 4-7 所示。

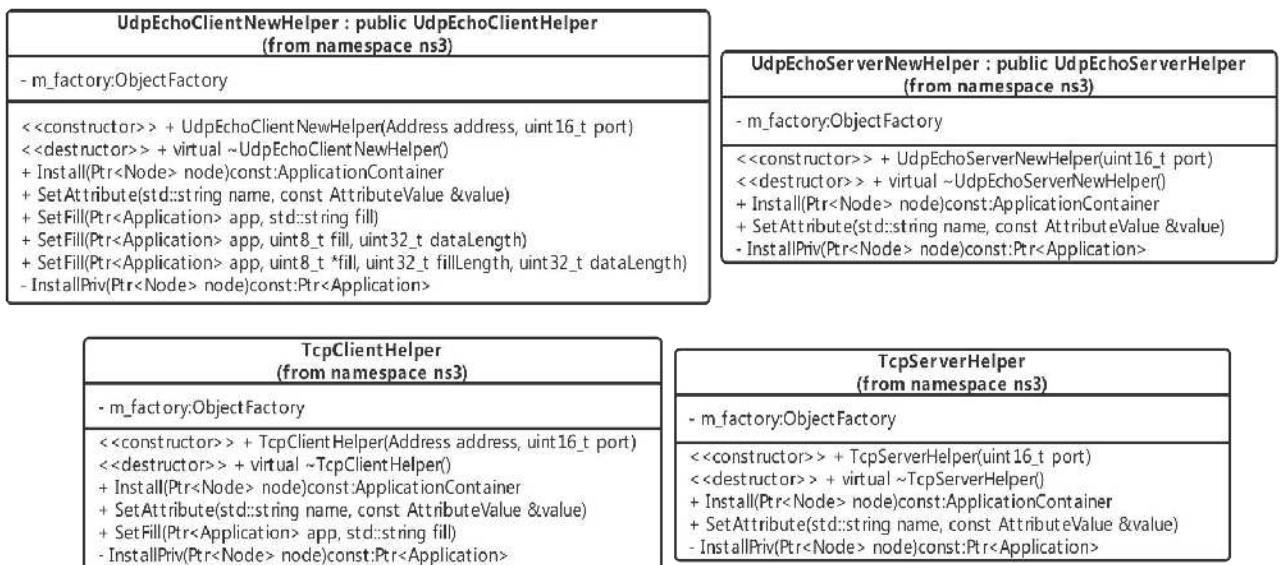


图4-7 应用程序安装助手类UML类图

Figure 4-7. Application installation helper class UML class diagram

(6) 标签类：客户端节点发送的数据包上都会打上相应的标签对象，并在服务器接收到的数据包中解析出这些标签，打上的标签不属于数据包的内容，因而不会影响数据包的大小和节点间数据传输的质量。为记录下客户端发送的数据包编号、多包传输下的状态编号、多包传输下状态数及数据包的发送时间，分别设计了四种标签类，它们具有的数据成员与成员函数如 UML 类图 4-8 所示。

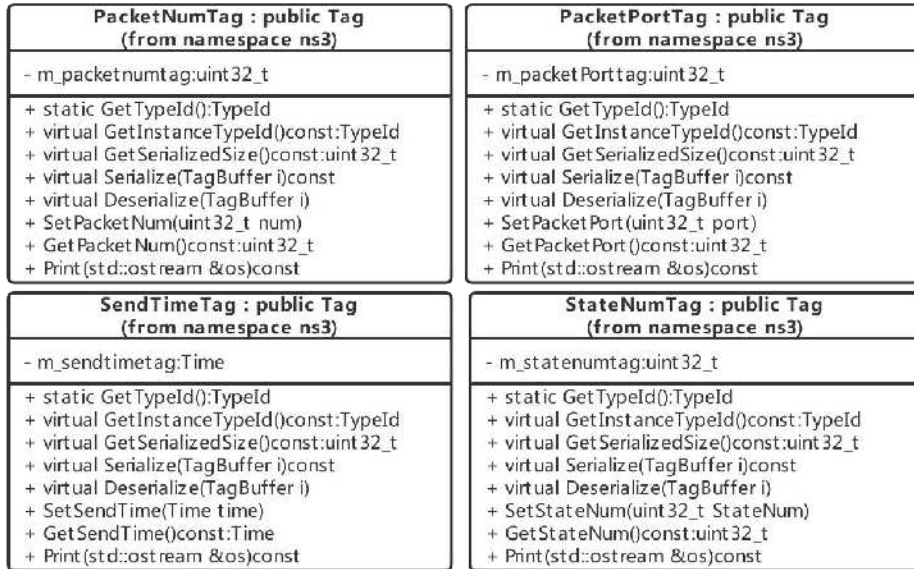


图4-8 标签类UML类图

Figure 4-8. Label class UML class diagram

(7) 结构体 IpIdcontent：如下表 4-1 所示，该结构体用来保存发送事件的目标节点 IP、事件 ID 和 packetcontent 等信息，同时该结构体的缓存数组能容纳 50 个待发送事件的信息。

表 4-1 IpIdcontent 结构体定义

Table 4-1. IpIdcontent structure definition

```

struct IpIdcontent
{
    Ipv4Address m_ipTo;// 目标节点 ip
    EventId m_eid;// 事件 id
    std::string m_packetcontent;// 数据包内容
    int m_mutipac;// 多包传输数据包个数
    int m_statenum;// 多包传输下包的状态编号
};
    
```

(8) 结构体 Data_MATLAB 与 Data_NS3：如下表 4-2 所示为两个结构体的数据成员。Data_MATLAB 结构体保存着 MATLAB 客户端发送的某一节点在 MATLAB 中仿真所得的信息，同步时会生成结构体数组发送给 NS3,而 Data_NS3 结构体则保存 NS3 中每个节点的仿真结果，它将通过服务器发送给 MATLAB 进行解析。

表 4-2 Data_MATLAB 与 Data_NS3 结构体定义
Table 4-2. Data_MATLAB and Data_NS3 structure definition

```

struct Data_MATLAB
{
    int sig;// 该节点在网络仿真中发送数据包的触发信号
    int des;// 该节点发送数据包的目标节点索引
    char packetcontent[900]);// 数据包的内容
    bool sys_send_node;// 该节点是否是控制系统中的节点
    int pos_vel_send_sig;// 是否需要从 MATLAB 中获得位置与速度信息的标志
    double pos;// 该节点获得的位置信息
    double vel;// 该节点获得的速度信息
};

```

```

struct Data_NS3
{
    int sig;// 该节点在两次同步之间是否接收到过数据包
    char packetcontent[900]);// 该节点在两次同步之间接收到的数据包的内容
    char Timedelay[500]);// 该节点在两次同步之间接收到的数据包的“推算”时延
    char PacketNum[200]);// 该节点在两次同步之间接收到的数据包编号
    char Sim_time[500]);// 该节点在两次同步之间接收到的数据包的接收时间
    char TimedelayReal[500]);// 该节点在两次同步之间接收到的数据包的真实时延
};

```

4.1.4 NS3 仿真脚本的设计

用户在使用 NS3 时需要编写仿真脚本以搭建网络仿真场景，本仿真脚本使用外部驱动仿真器以实现协同仿真。

仿真脚本的执行流程如下：执行主函数→初始化测试脚本类→利用命令行参数设置仿真参数→运行 Run()函数搭建仿真场景→运行 SeedManager()函数设置仿真随机种子→运行 Configure_phyMode_Defaults()函数配置物理层传输模式与全局参数→运行 CreateNodesphymacmobility()函数，其功能包括：1.从 NS3 节点拓扑模块中提取节点个数与节点位置 2.为物理层和 mac 层的发送和丢包跟踪配置回调 3.为节点配置移动与位置回调 4.配置物理层和 mac 层并安装通信设备在节点上，开启网络跟踪文件生成→配置外部驱动仿真器参数→运行 InstallInternetStack()函数安装 internet 协议栈，其功能包括：1.安装 IP 层路由协议，开启路由表跟踪文件生成 2.安装 internet 协议栈并为所有节点分配 IP 地址→运行 InstallApplication()函数，其功能包括：1.选择安装 TCP 或 UDP 的应用程序 2.如果用户选择 LTE 网络，激活边链路承载 3.如果期望协同仿真使用设定的延迟与丢包率运行，运行 Set_delayloss()函数进行设置→运行 SetupWaveMessages()函数，如果用户的物理层模式选为 WAVE 协议，设置并发送 bsm（基本安全消息）数据包→运行 ConfigureLogTracing 函数生成额外的跟踪文件，并判断是否开启 log 系统中必要的日志组件→配置网络节点动画与数据流量文件生成→仿真场景搭建完毕，启动并结束仿真。通过命令行改变仿真脚本的参数可以自由组合出多样的仿真场景。

4.2 MATLAB/Simulink 各模块程序设计

4.2.1 MATLAB 仿真驱动模块

MATLAB 仿真驱动模块由一组紧密关联的 MATLAB 脚本子模块组成，包括仿真初始化模块、仿真主循环模块、单步仿真驱动模块、通知解析与单步规划模块、Simulink 仿真启停控制模块，根据控制器驱动方式的不同（同步方式的不同），这些模块各自都分别按照时间驱动（采用主从式同步）与事件驱动（采用全局事件驱动同步）两种方式运行，此外，还包括了 Simulink 暂停回调模块、控制器响应模块（仅用于事件驱动）与仿真模型关闭模块，接下来将逐一介绍它们的功能。另外，脚本中使用 eval 函数以实现多组控制系统模型在同一仿真场景下的仿真。

(1) 仿真初始化模块：该模块用于初始化仿真运行的参数，并打开 Simulink 模型文件。值得注意的是，如果在交互界面中选择的路由协议编号为 1 (olsr 协议)，由于 olsr 协议的路由拓扑发现需要一段时间，在这段时间内节点应用层不能够正确接收数据包，过早的发送数据包会导致 NS3 仿真中 olsr 协议的路由拓扑发现失败，所以对于应用层我们需要预留一段时间不发送数据包，使得 olsr 协议在没有流量负载的情况下收敛（路由发现成功），因此当使用该协议时，变量 olsr_trigger_time 被初始化为 10，即 Simulink 中的模型在开始仿真后前 10 秒钟参考输入为空（0）。该模块时间驱动与事件驱动的差异在于初始化参数略有不同，同时在事件驱动下该模块会保存节点的发送状态，用于客户端设置通知结构体中数据包发送动作的触发信号，如图 4-9 为该模块的运行流程，虚框表示事件驱动下才运行的流程。

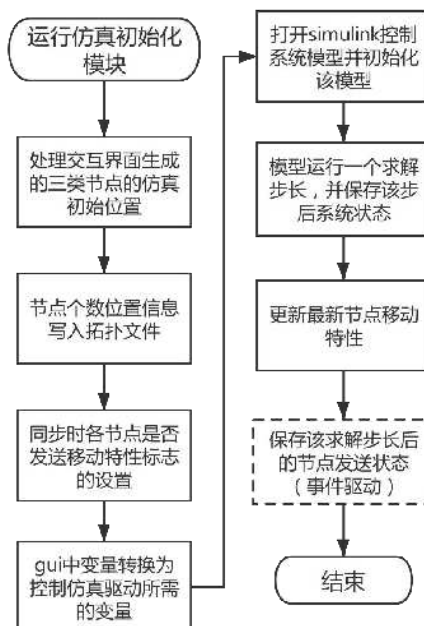


图4-9 仿真初始化模块运行流程图

Figure 4-9. Simulation initialization module running flowchart

(2) 仿真主循环模块：它将不断循环调用单步仿真驱动模块，以推进 Simulink 中控制系统的仿真过程。该模块时间驱动与事件驱动的差异仅仅在于：模块开始运行时，采用时间驱动的模式将会调用 mex 编译器编译时间驱动客户端的源程序生成客户端组件，在接下来的整个仿真过程中将一直使用该组件完成与 NS3 的交互，而采用事件驱动的该模块不会执行该步骤。如图 4-10 为该模块的运行流程。

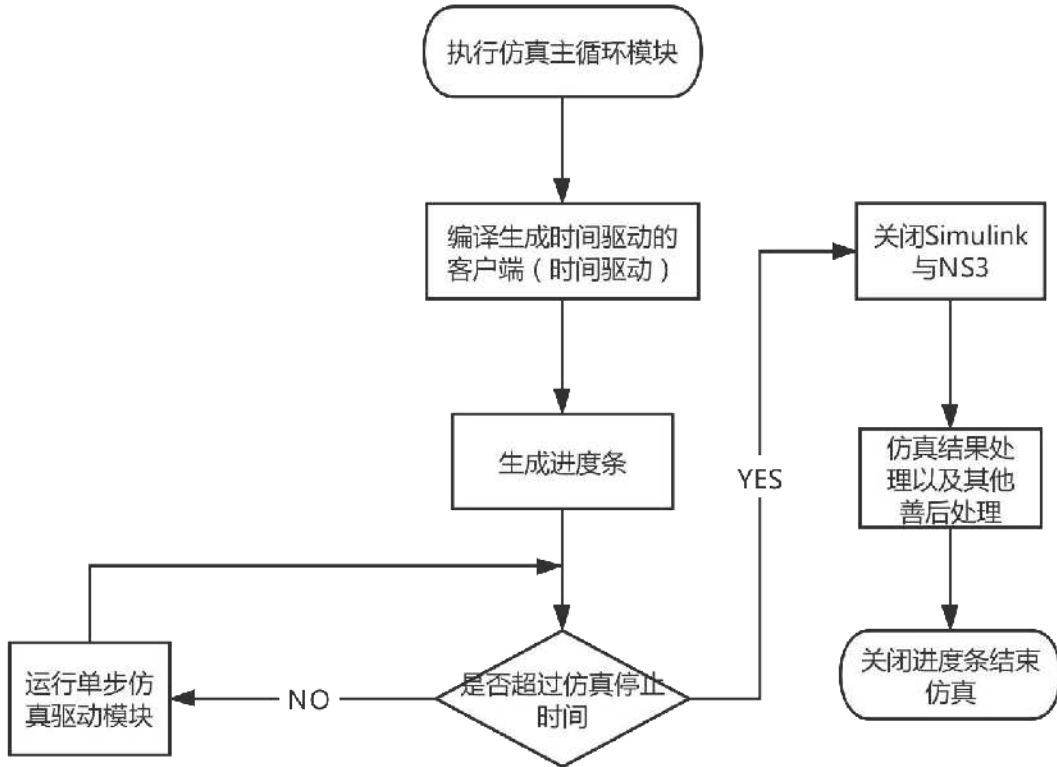


图4-10 仿真主循环模块运行流程图

Figure 4-10. Simulation main loop module operation flow char

(3) 单步仿真驱动模块：该模块运行 MATLAB 时间轴中仿真同步事件之间的单步，该模块在两种驱动方式的流程将不大相同，采用时间驱动时将会直接运行仿真主循环模块中编译生成的时间驱动客户端组件，而在事件驱动下每次与 NS3 进行同步时，由于客户端通知结构体中数据包发送动作的触发信号被修改，为确保修改值被正确发送，每次运行该模块前都将重新编译一次事件驱动客户端源文件。之后，该模块将分别运行各自的通知解析与单步规划模块及 Simulink 仿真启停控制模块，事件驱动下还需要运行控制器响应模块以完成可能的控制器响应。如图 4-11 为该模块的运行流程，虚框表示事件驱动下才运行的流程。

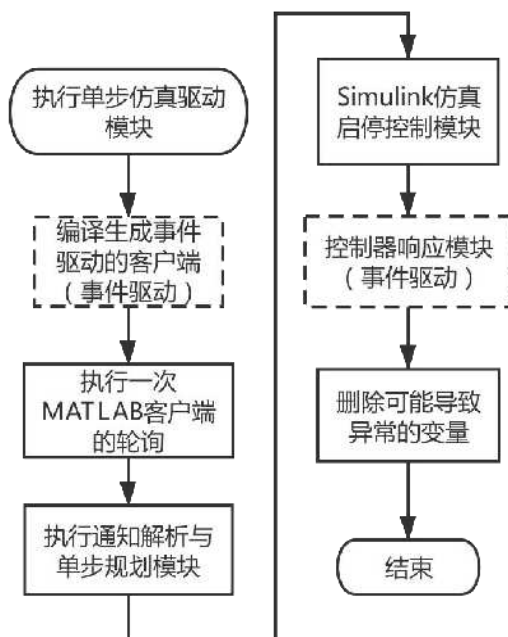


图4-11 单步仿真驱动模块运行流程图

Figure 4-11. Single-step simulation driver module operation flow chart

(4) 通知解析与单步规划模块：当客户端解析收到的通知结构体后，该模块根据不同的方式规划接下来一个单步中的仿真过程。该模块实现的核心思想在于：控制系统仿真复现了同样单步内网络仿真中控制系统的运行情况。另外该模块定义了两种应用层数据包处理模式，一种方式是设置接收数据包延迟的阈值，防止接收延迟过大的数据包，另一种则是将当前接收事件的数据包编号与之前接收事件后仿真使用的数据包编号进行比较，如果当前接收到的数据包编号大于之前使用过的数据包编号，则采用该数据包内数据，否则丢弃该数据包。该模块在两种驱动方式下的设计思想与运行流程也不大相同，在时间驱动下，单步为一个周期步长，如果周期步长内没有接收到数据包，该单步内将该数据包延迟近似为一个周期步长，通过计算每个数据包接收事件的时延跨过的周期步长的个数，以及在当前步长内的时间长度，将该周期步长内求解步长的索引取值范围分割成一系列上下限的组合，这些组合将会用于确定接下来分段执行该周期步长内仿真的求解步长的索引，同时收集每一次数据包接收事件的相关信息并保存到相应结构体，并根据接收事件的先后次序将结构体进行冒泡排序，排序后的结构体将用于依次复现数据包接收事件，而在事件驱动下，单步为相邻两次同步之间的仿真时间，因此只需通过该仿真时间对应的求解步长索引的上下限即可执行该单步。另外不同于时间驱动使用真实数据包延迟确定某段仿真的求解步长索引的上下限，事件驱动采用“推算”延迟确定该单步的求解步长索引的上下限。如图 4-12 为该模块的运行流程。

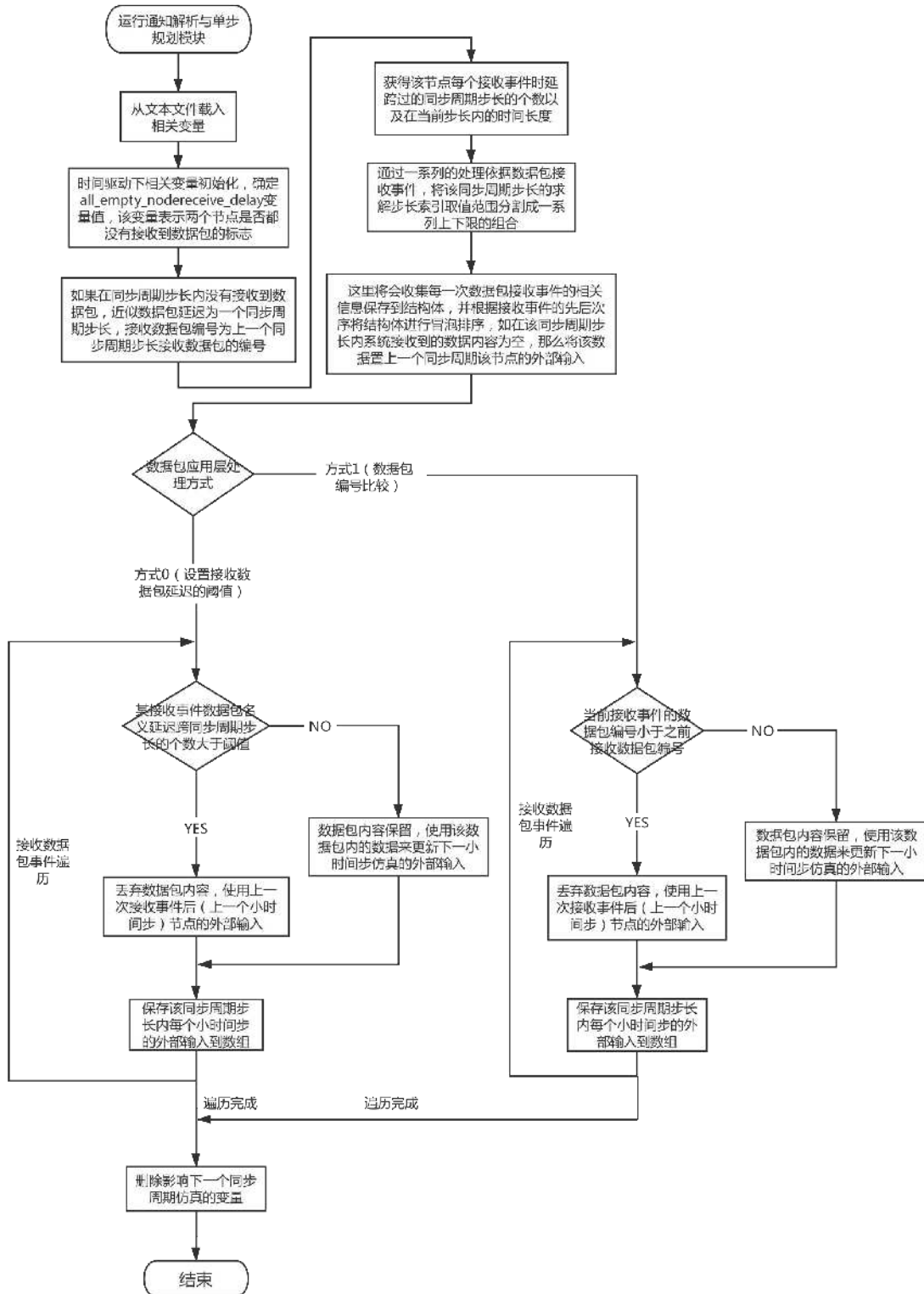


图4-12 (a) 通知解析与单步规划模块运行流程图 (时间驱动)

Figure 4-12. (a) Notification parsing and single step planning module flow chart (time driven)

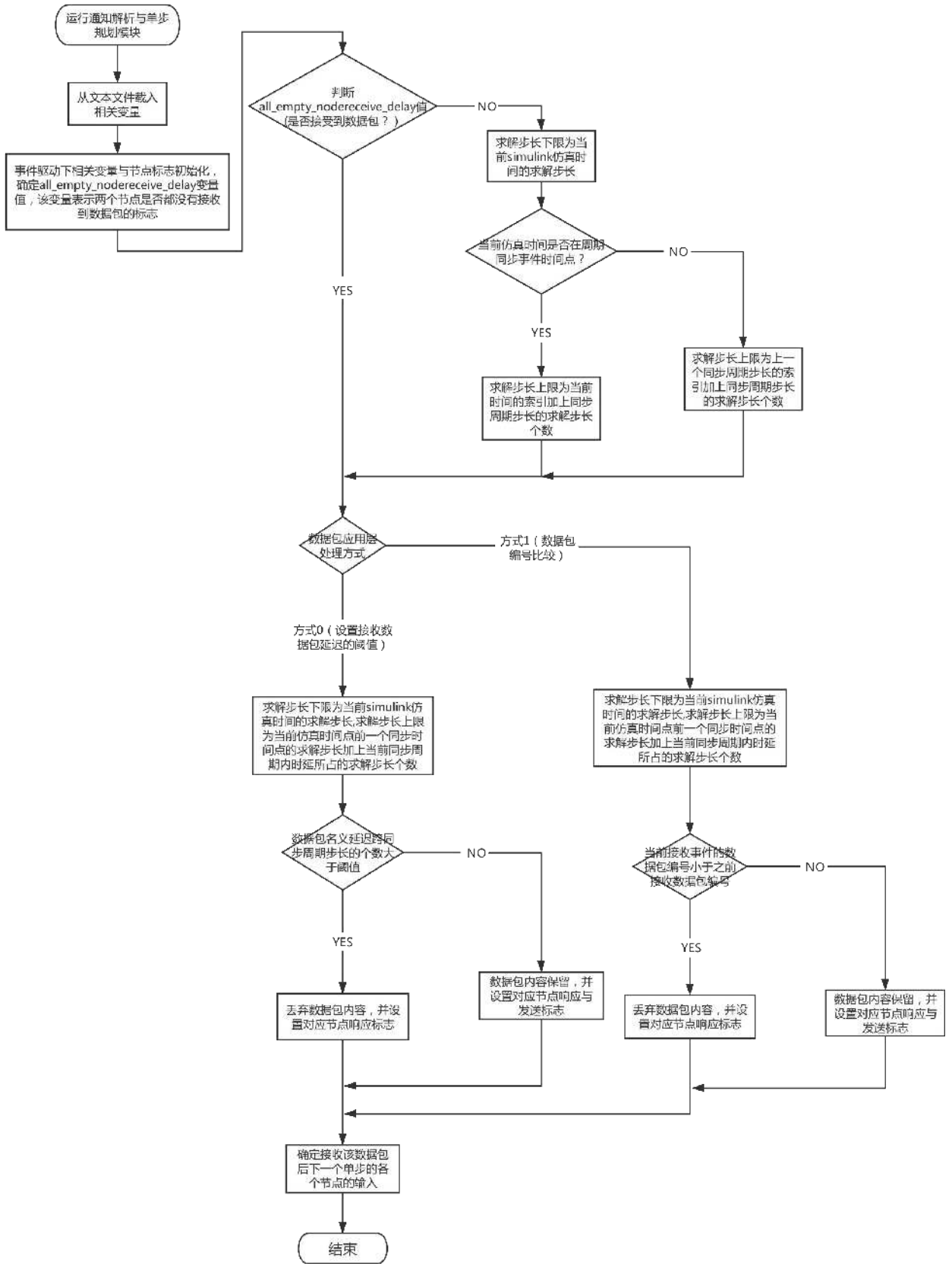


图4-12 (b) 通知解析与单步规划模块运行流程图 (事件驱动)

Figure 4-12. (b) Notification parsing and single step planning module flow chart (event driven)

(5) Simulink 仿真启停控制模块: 该模块直接控制 Simulink 中控制系统仿真模型完成一个求解步长的仿真, 它通过在每个求解步长使用 `set_param` 函数调用 Simulink 仿真命令 `start`、`pause`、`step` 与 `stop` 实现在每个求解步长启停 Simulink 仿真, 以达到在某一精确时间点更新系统外部输入的目的。在时间驱动下依次执行通知解析与单步规划模块中得到的单步中各数据包接收事件序列, 在数据包接收事件发生时采用新的求解步长索引的上下限, 并更新模型的外部输入, 循环启停执行每个求解步长直到下一个事件点, 当事件序列没有剩余事件时则该单步执行完成。而在事件驱动下仅通过某单步的求解步长索引的上下限运行相邻两次同步之间的仿真, 之后更新外部输入变量为下一次同步中收到的该模型的输入。如图 4-13 所示为该模块的运行流程。

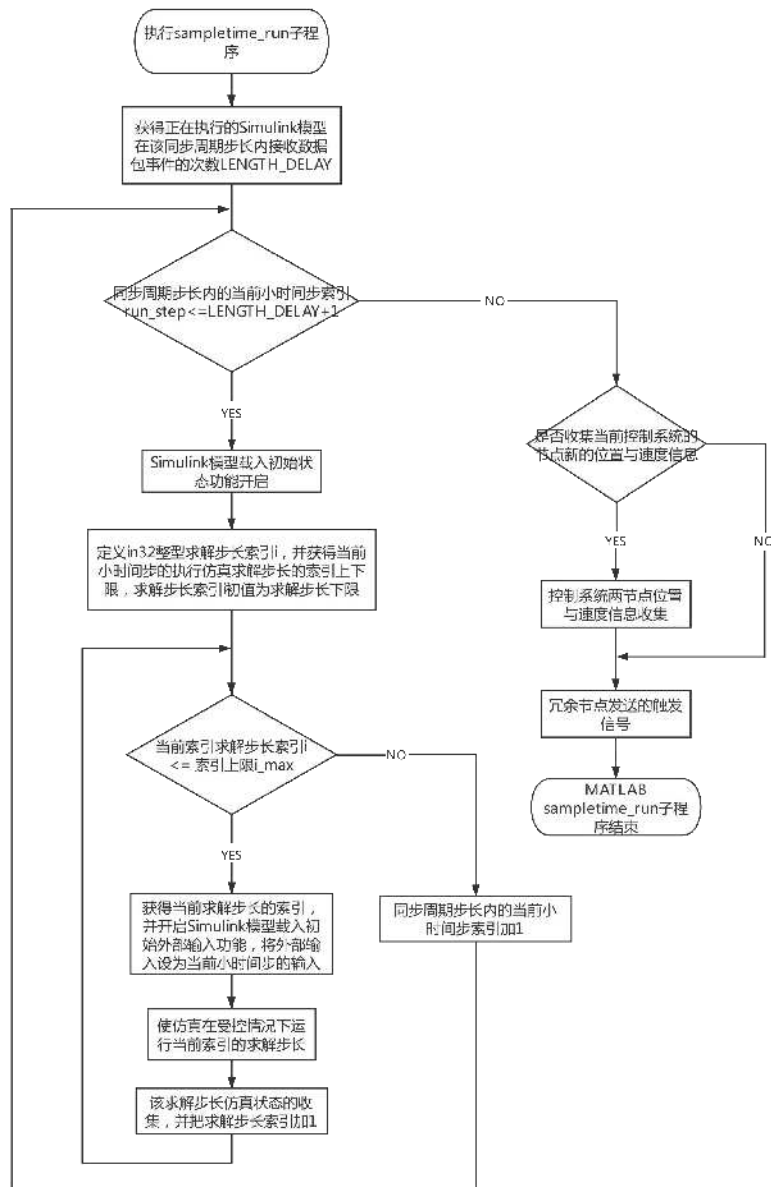


图4-13 Simulink仿真启停控制模块运行流程图

Figure 4-13. Simulink simulation start-stop control module operation flow chart

(6) Simulink 暂停回调模块：在 Simulink 仿真模型中设置模型回调函数 Stopfcn，Stopfcn 函数将在每个求解步长暂停时回调运行 Simulink 暂停回调模块。该模块根据两种仿真的驱动方式选择两种回调子程序，将仿真中每个求解步长中控制系统的内部状态与输出状态回调给工作空间相应变量中，并写入到文本文件中，该模块简略代码如下表 4-3 所示。

表 4-3 Simulink 暂停回调模块代码

Table 4-3. Simulink pauses callback module code

```

% 判定当前运行模型的模型 ID
    if coupleid == det_coupleid
% 获得当前求解步长的索引保存到变量 k
    k_coupleid = i_coupleid;
% 收集该模型的节点 1 的输出状态保存到 Node1out_step 作为单求解步长输出状态
    Node1out_step_coupleid(1,1:n_node1_var(coupleid)) = ...
    yout_coupleid(1,1:n_node1_var(coupleid));
% 收集该模型的节点 2 的输出状态保存到 Node2out_step 作为单求解步长输出状态
    Node2out_step_coupleid(1,1:n_node2_var(coupleid)) = ...
    yout_coupleid(1,n_node1_var(coupleid)+1:n_node_var(coupleid));
% 如果为时间驱动方式或者事件驱动方式且该时刻未发生异步事件则执行该子程序
    if ((qudong_mode == 1)|| (qudong_mode == 2 && yibushijian == false))
% 变量 Node1out 为节点 1 的输出状态序列,序列的索引 k 分量保存该次求解步长节点 1 输出状态
    Node1out_coupleid(k_coupleid,1:n_node1_var(coupleid)) = ...
    Node1out_step_coupleid(1,1:n_node1_var(coupleid));
% 变量 Node2out 为节点 2 的输出状态序列,序列的索引 k 分量保存该次求解步长节点 2 输出状态
    Node2out_coupleid(k_coupleid,1:n_node2_var(coupleid)) = ...
    Node2out_step_coupleid(1,1:n_node2_var(coupleid));
    end
% 将每个节点的单步输出状态保存到 node_coupleid.txt 文件中去
    save('node_coupleid.txt','tout_coupleid','Node1out_step_coupleid','Node2out_step_coupleid','-ASCII');
end

```

(7) 控制器响应模块：在事件驱动模式下，该模块执行每次控制器接收到数据包后计算控制量的任务，并设置该事件下的控制系统各节点数据包发送标志。该模块判定控制器收到数据包且控制器没有丢弃该数据包，则运行一个求解步长以获得控制器的控制量，执行完该求解步长后仿真将回退该求解步长，以防协同仿真时间出现不一致。

4.2.2 控制系统 Simulink 模型

如前所述，本仿真工具将某一控制系统中的传感器和执行器看做为依赖被控对象的同一节点，而控制器看做是另一节点。因此在 Simulink 控制系统文件中创建两个子系统模块以代表两个节点，如图 4-14 所示。

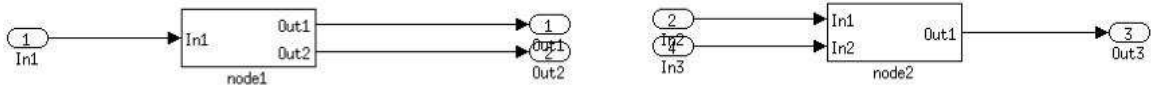


图4-14 可嵌入的控制系统Simulink模型

Figure 4-14. Embedable control system Simulink model

node1 代表被控对象节点，其内部封装有被控对象模型，通过输入端口在每个求解步长导入最新的控制量，输出端口输出被控对象的输出到相应变量，类似的 node2 代表控制器节点，内部封装有控制器模型，输入端口在每个求解步长导入最新的被控对象输出，输出端口则输出控制器的控制量，当然也可以在两节点子系统模块中各封装有一个完整的控制系统，而不仅仅是把一个控制系统按功能拆开封装至对应子系统模块。因此该模型兼容多类控制系统的嵌入。

4.2.3 MATLAB 客户端组件

MATLAB 客户端组件也根据驱动方式的不同分为两种，客户端从 MATLAB 工作空间及文本文件中获得 Simulink 仿真一个单步的结果，并将其自定义封装后发送给 NS3 迭代服务器，之后进入阻塞模式直到收到 NS3 发送的包含新一轮仿真结果的数据包，数据包解析完成后客户端关闭。客户端发送与接收数据包代码如下表 4-4 所示。

表 4-4 MATLAB 客户端组件部分代码

Table 4-4. MATLAB client component partial code

```

memset(snd_buf,0,sizeof(snd_buf));
memcpy(snd_buf,&trans,sizeof(trans));
memcpy((long*)((long)snd_buf + sizeof(trans)),&QUIT,sizeof(int));
send(sock, &snd_buf, sizeof(snd_buf), 0);
for(int buf_next_ass_loop = 0;buf_next_ass_loop<SYSNUM;buf_next_ass_loop++)
{
    Timebuf[buf_next_ass_loop] = nextTime[buf_next_ass_loop];
}
send_pac = true;
...
if(send_pac == true)
{
    int size_rbuf;
    size_rbuf = sizeof(recv);
    char recv_buf[size_rbuf];
    memset(recv_buf,0,sizeof(recv_buf));
    recv(sock, &recv_buf, sizeof(recv_buf), 0);
    memset(&rec,0,sizeof(rec));
    memcpy(&rec,recv_buf,sizeof(recv_buf));
}

```

4.2.4 仿真平台交互界面

根据用户不同仿真场景的需求，把用户可设置的仿真参数分为全局参数、控制系统参数、节点网卡参数、NS3 参数及冗余节点发送参数，此外用户可以通过按钮使用载入仿真模型、控制协同仿真过程、绘制仿真结果、演示仿真场景动画及重置与退出界面 5 个功能，用户交互界面如图 4-15 所示。

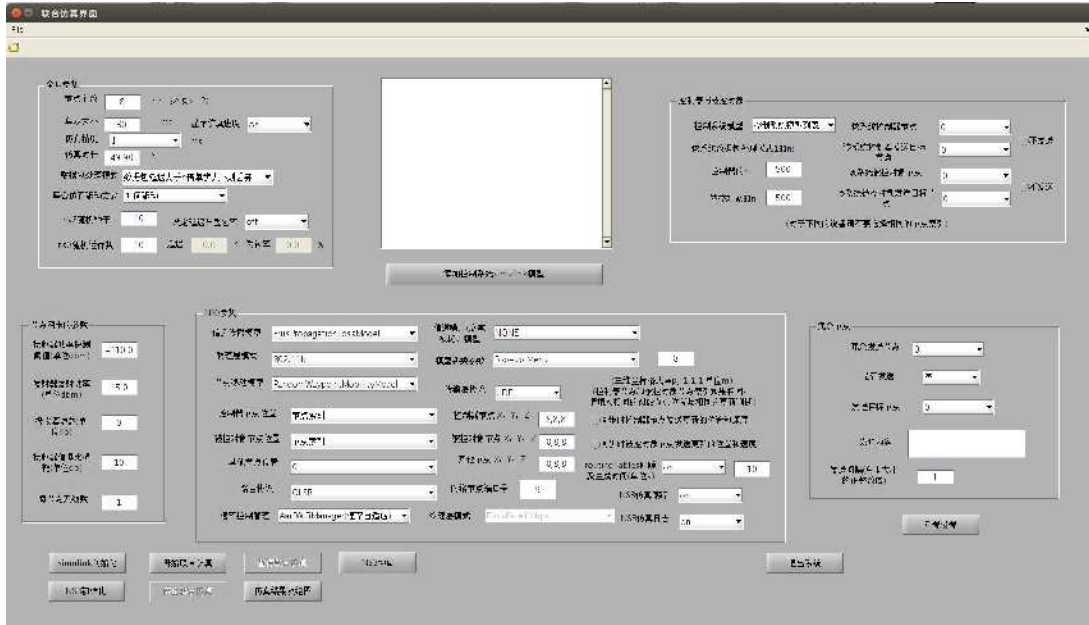


图4-15 仿真平台交互界面

Figure 4-15. Simulation platform interactive interface

界面的布局使用 MATLAB 内置的 MATLAB GUI 实现，每个交互控件对象分配回调函数增加监听事件，从而实现当某控件事件发生时，应用会做出响应并运行预定的功能。接下来依次介绍各类参数的设置与控件功能，用户可依据介绍使用该平台。

(1) 全局参数用于设置从整体上影响仿真的参数：包括节点个数、单步大小（该单步指两种驱动方式下都存在的周期步长）、仿真精度（指 Simulink 的求解步长）、仿真时长、应用层数据包处理模式、NS3 的随机种子以及人为设置仿真中数据包的延迟与丢包率。

(2) 控制系统参数用于设置影响控制系统仿真的参数，通过控制系统模型控件可切换选中的控制系统模型，选中某一控制系统模型后用户可以设置该模型的控制器和被控对象节点的索引，以及发送目标节点的索引，同时用户也可以设置不发送该节点的数据。此外用户可以设置数据包处理模式 1 下按延迟大小处理的阈值。

(3) 节点网卡参数可以设置网卡的各项参数包括接收器功率探测阈值、发射器发射功率、接收器增益、接收器信噪比以及节点天线个数。

(4) NS3 参数用于设置网络仿真的场景，包括信道传播模型、信道损耗模型、模型的参数、物理层采用的通信协议、节点移动模型、传输层通信协议、节点位置、是否同步设备节点的速度和位置、路由协议、节点端口号、wifi 速率管理方式、常速率模式下的速率模式以及仿真跟踪与日志是否开启。

(5) 冗余节点参数将设置除控制系统节点外其他节点的数据发送情况，用户可以切换冗余发送节点控件设置每个剩余节点，包括剩余节点发送数据包的触发信号，发送数据包的内容以及发送数据包的周期。

(6) 仿真模型的载入功能通过界面中央的列表框与添加控制系统模型按钮两控件实现，用户按下该按钮将跳出添加 Simulink 模型的对话框，添加的模型文件的名称和路径将会显示在列表框中，如果重复添加同一个文件将跳出错误提示。

(7) 协同仿真过程的控制通过 Simulink 初始化、NS3 初始化、开始联合仿真、停止联合仿真与暂停联合仿真 5 个按钮实现。按下 Simulink 初始化按钮将触发仿真初始化模块运行，而按下 NS3 初始化将调用 system 命令启动 NS3 仿真进程，如果没有添加控制系统模型将跳出错误提示，在初始化前需要根据仿真需求选择合适参数，如果初始化后修改了参数设置则必须重新初始化仿真。按下开始联合仿真按钮将触发仿真主循环模块运行。按下停止联合仿真按钮则设置仿真停止标志 quit，使协同仿真将会在完成一个单步后停止。按下暂停联合仿真按钮仿真暂停并进入调试模式，如果希望继续进行仿真，可以通过按“F5”或在命令行输入 return 继续仿真。

(8) 仿真结果的绘制通过仿真结果及绘图按钮实现，当仿真处于暂停或停止时，按下该按钮将跳出如图 4-16(a)所示的绘图器，用户可以选中左边列表框中任意两个变量并选择 5 种不同的绘图命令，同时可切换两变量自变量与因变量的关系，之后会跳出如图 4-16(b)所示对话框，用户可选择某变量的几个分量绘制与对比仿真结果。



图4-16 (a) 简单绘图器

Figure 4-16. (a) Simple plotter

(b) 选择变量状态

(b) Select variable states

(9) 仿真场景动画演示功能通过 NS3 动画按钮实现。NS3 自带有两种网络动画演示软件，分别是 PyViz 和 NetAnim，由于该平台修改了 NS3 默认的仿真器内核，因此作为仿真即时演示的 PyViz 将不能够使用，所以选择离线回放演示软件 NetAnim 演示仿真动画，NetAnim 使用在仿真期间收集的 XML 跟踪文件来演示动画、路由表与流量监控，如图 4-17 所示。

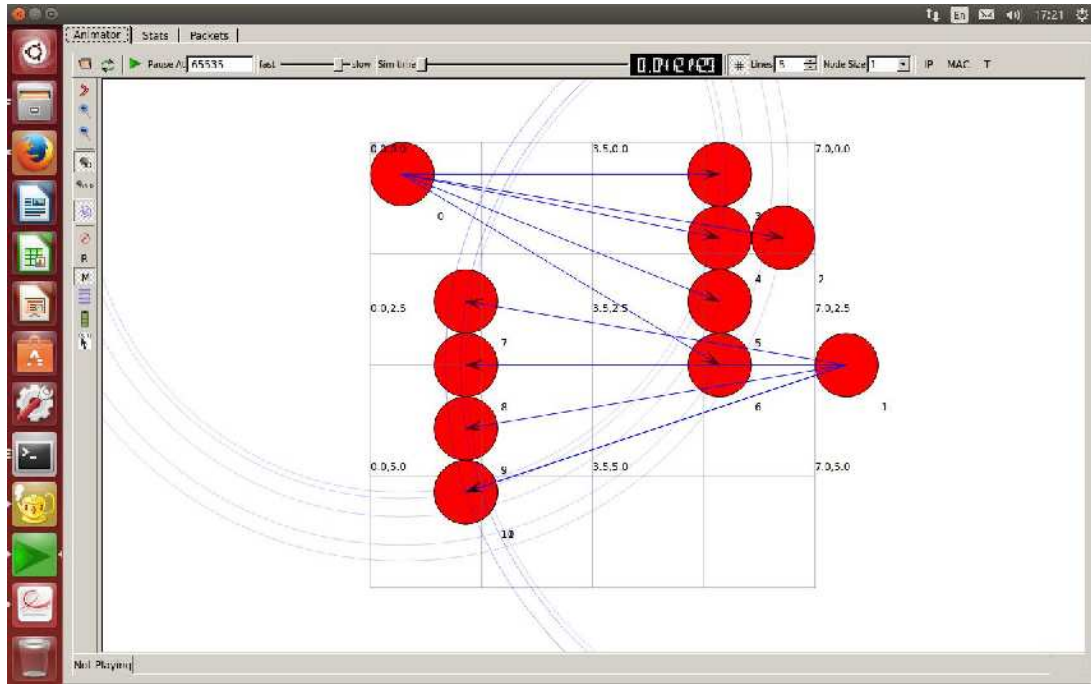


图4-17 NetAnim动画回放网络仿真

Figure 4-17. NetAnim animation playback network simulation

4.3 本章小结

本章详细介绍了协同仿真平台底层模块的设计与实现。首先，简要介绍了该平台中 NS3 有必要使用的基本类。其次，介绍了 NS3 几种新类的设计，特别是改造默认的仿真器类获得的外部驱动仿真器类，改进后的仿真器能获得外部应用事件并推进仿真。接着介绍了 MATLAB/Simulink 的各模块，包括用于控制仿真过程的 MATLAB 仿真驱动模块、具有较强兼容性的 Simulink 控制系统模型、MATLAB 客户端组件及灵活全面的仿真平台交互界面。

第五章 仿真平台测试与结果分析

本章将在搭建的仿真平台上针对多个网络化控制系统进行测试以验证平台灵活与可靠性。包括桥式吊车远程控制系统与通信受限网络化控制系统的仿真，仿真结果将展现通过网络传输数据对控制系统造成的影响，另外针对这两种网络化控制系统设计了相应的补偿方案以提升系统性能。

5.1 桥式吊车远程控制系统仿真测试

5.1.1 系统描述与控制方案

桥式吊车是现实中在工业环境中应用最为广泛的控制系统，如图 5-1 所示，点 A 表示运行在桥架上的吊车，其中 s_A 是小车在 s 轴上的坐标 ($s_A \neq 0, z_A = 0$)， m_A 是小车质量， F_A 是作用在小车上由马达产生的水平驱动力， p 是吊钩加上负载施加在小车上的绳索拉力。点 B 表示吊钩， s_B, z_B 分别是吊钩在 s 与 z 轴上的坐标， m_B 是吊钩质量。 l, θ 分别是绳索长度、绳索同垂直方向之间的夹角。

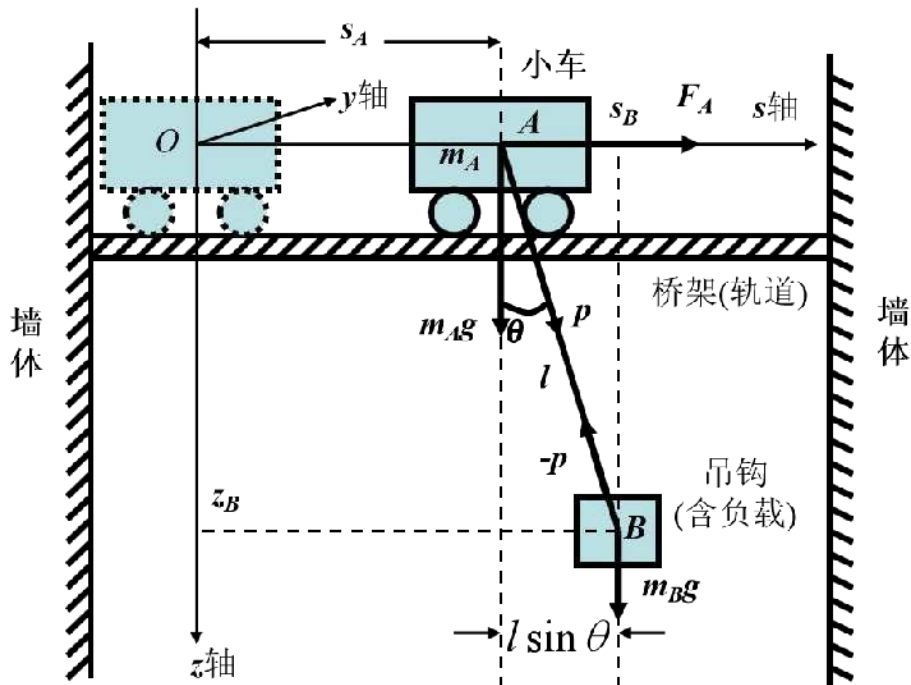


图5-1 桥式吊车控制系统示意图

Figure 5-1. Schematic diagram of the Bridge crane control system

吊车的工作任务在于：首先将负载从地面上吊至一个预定的位置(改变 zB)，然后再送至某个对象的上方(改变 sA)，最后将负载在一个确定的位置卸下(再次改变 zB)。

由于行车系统(开环)本身是不稳定的，从调节控制技术角度讲，可采用全状态负反馈进行调节控制。由于小车的位置 sA 是一个能观测的被控对象状态变量，所以可以利用小车位置 sA 的测量值设置全状态观测器重构系统状态，并配合全状态反馈输出反馈量使控制系统闭环稳定，全状态反馈器的反馈参数根据极点配置情况通过 Ackerman 公式确定，如图 5-2 所示是加入全状态反馈器以及全状态观测器后的控制系统框图。

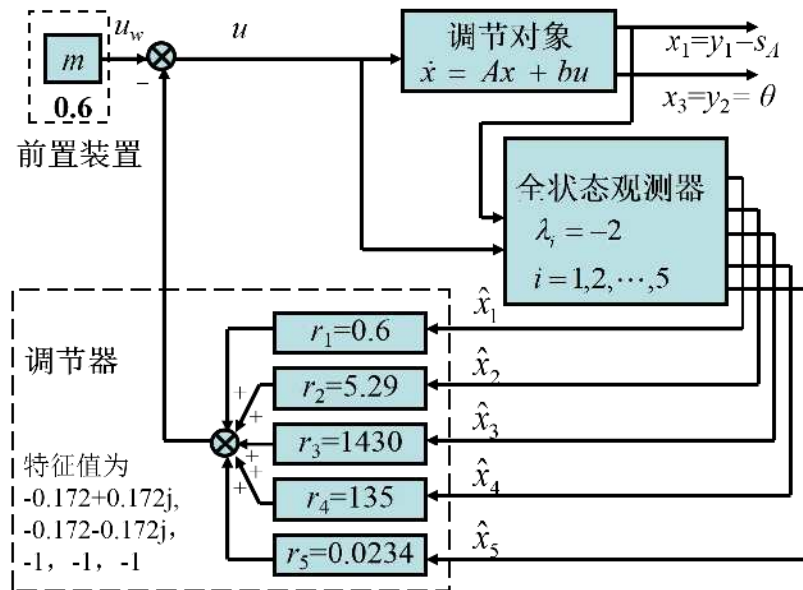


图5-2 加入观测器、调节器与前置装置的闭环系统方框图

Figure 5-2. Closed-loop system block diagram of the observer, regulator and pre-device

5.1.2 吊车远程控制系统仿真

一般来说，可以把全状态反馈器以及全状态观测器作为一个整体看作控制器，通常这类工业现场的控制器是通过嵌入式单片机或 PLC 进行现场控制，现场的布线一般通过有线网络，网络的运行对控制精度没有很大的影响，但是如果现场的控制器的控制器受到了损毁，无法在现场计算控制量，为保持控制系统的继续运转，可考虑通过无线网络让远程控制器介入控制，这时通信网络的特性将对控制系统产生较大的影响。

这里可以使用前面介绍的协同仿真平台来仿真该远程控制场景，仿真场景的参数设置^[45]如下表 5-1 所示。

表 5-1 吊车远程控制系统仿真参数设置

Table 5-1. Crane remote control system simulation parameter setting

参数列表	所选参数
节点个数	2 个
单步大小	30ms
仿真精度	1ms
仿真时长	40 秒
仿真驱动方式	时间驱动
信道路径传播模型	对数距离衰减模型
信道噪声模型	NormalRandom (慢衰落模型)
发射器发射功率	15dBm
接收器功率探测阈值	-110dBm
物理层协议	802.11b
节点移动模型	常位置模型
传输层协议	UDP
路由协议	aodv
物理层传输速率模式	常速率模式 DsssRate11Mbps

控制器被依次放置在离被控对象距离 30m, 35m, 40m 的位置, 并依次进行仿真。本章提出如下假设: 1.完整的被控对象传感和控制信息分别由单个数据包承载, 大小分别为 33 字节和 17 字节。数据包大小较小保证了短的包传输时间并产生低网络流量。2.采样、驱动和控制量计算的任务时间与网络延迟相比可以忽略不计。如图 5-3 所示, 为受控制器距离影响的小车位置 sA 跟踪性能的仿真结果。

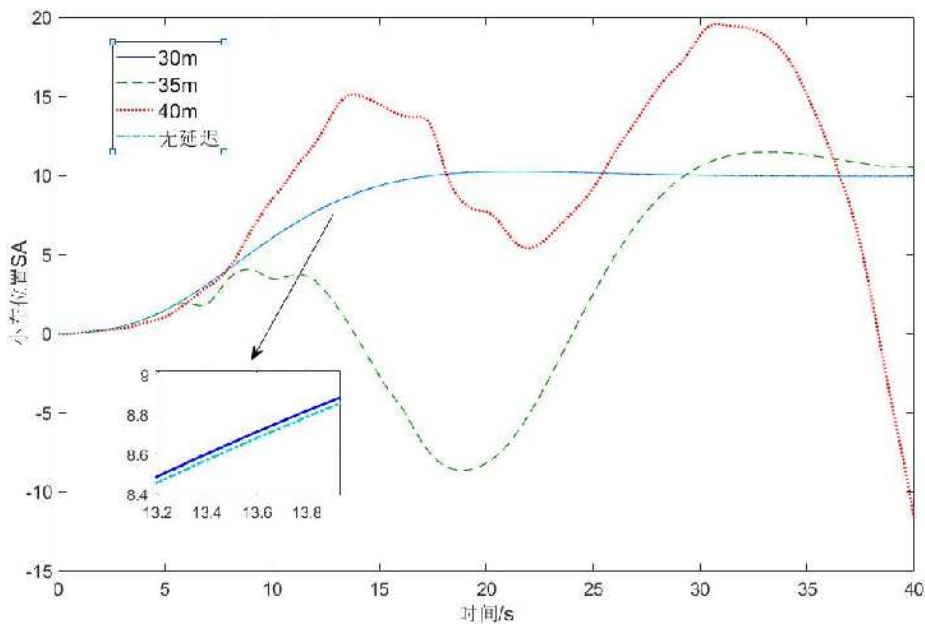


图5-3 小车位置跟踪效果图

Figure 5-3. Car position tracking effect chart

由仿真结果可看出，在控制器距离为 30m 时，节点间数据连接十分稳定，延迟很小且几乎无丢包，小车位置跟踪曲线超调量为 2.8%，相比无延迟下的超调量 2.6% 仅仅高了 2%，因此通信网络对控制性能的影响微乎其微。而当控制器距离超过 30m 后，因信道功率衰减上升使接收器无法探测到功率较小的数据包，导致丢包率上升、时延变大与建立连接时间延长，跟踪性能迅速降低，在控制器距离为 35m 时，跟踪曲线出现抖动，小车在 12s 至 19s 出现逆行后才得到有效控制，当控制器距离为 40m 时跟踪曲线剧烈震荡后直接发散，小车已无法得到控制，因此控制器有效的安置距离为 30 米以内，表 5-2 展示了控制器在不同距离下的系统性能。

表 5-2 控制器距离对系统性能的影响

Table 5-2. Influence of controller distance on system performance

控制器距离	系统性能			
	超调量	丢包率	整回路平均时延	建立连接时长
30m	2.8%	0%	0.0049s	0.0528s
35m	15%	0.18%	0.0914s	0.0514s
40m	X	5.07%	0.616s	3.0584s

为一定程度上解决由被控对象与控制器间距离增大导致的动态性能与稳定性恶化的问题，最直接的方法是提高节点网卡的发射器发射功率或降低接收器功率探测阈值以减少丢包与时延，如图 5-4 所示为将发射器发射功率提升至 25dBm 的结果，图 5-5 为接收器功率探测阈值降低至-120dBm 的结果，表 5-3 为这两种方法下系统性能表现，可以看出这两种方法都提高了系统的动态性能与稳定性。

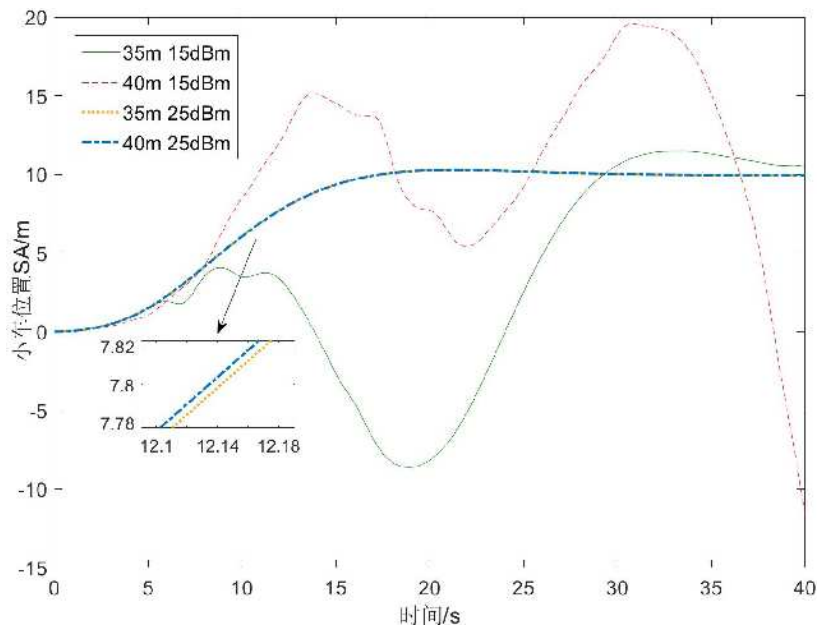


图5-4 不同发射器发射功率下的小车位置跟踪比较图

Figure 5-4. Comparison of car position tracking under different transmitter transmit power

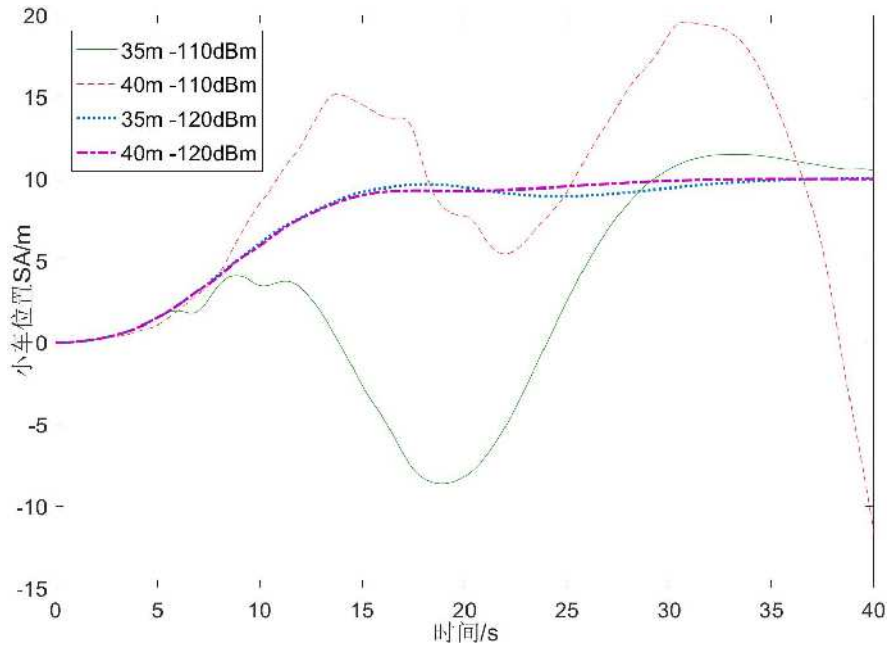


图5-5 不同接收器功率探测阈值下的小车位置跟踪比较图

Figure 5-5. Comparison of car position tracking under different receiver power detection thresholds

表 5-3 不同发射器发射功率与接收器功率探测阈值下的系统性能

Table 5-3. System performance under different transmitter transmit power and receiver power detection thresholds

控制器距离	发射器发射功率	接收器功率探测阈值	系统性能			
			超调量	丢包率	整回路平均时延	建立连接时长
35m	25dBm	-110dBm	2.7%	0%	0.0039s	0.0445s
40m	25dBm	-110dBm	2.7%	0%	0.0043s	0.0487s
35m	15dBm	-120dBm	0.02%	0.21%	0.037s	0.0514s
40m	15dBm	-120dBm	X	0.21%	0.052s	0.0514s

另外可以在控制器中采用主动补偿方案，该方案采用控制器接收的相邻数据包的发送间隔与数据包时延估计被控对象的状态，以补偿因这两种因素引起的状态观测器观测被控对象状态不准的情况，具体补偿方法如下，控制器在 k 时刻从接收到数据包，其包含了被控对象 $k - \tau_{sc,k}$ 发送时刻（ $\tau_{sc,k}$ 为该传感器到控制器的数据包的时延）的小车位置状态输出 $y(k - \tau_{sc,k})$ 、被控对象的外部输入 $u(k - \tau_{sc,k})$ ，同时根据打上的数据包标签获得该数据包的时延 $\tau_{sc,k}$ 以及该数据包的发送时间 $k - \tau_{sc,k}$ ，此时控制器内保存有前一次数据包的发送时间 $k - \tau_{sc,k} - \tau_i$

(τ_i 为接收的相邻数据包的发送间隔)、之前估计出的该时刻的状态 $\hat{x}(k - \tau_{sc,k} - \tau_i)$ 与之前的外部输入 $u(k - \tau_{sc,k} - \tau_i)$ ，之后依据被控对象的状态空间模型、 τ_i 、 $\hat{x}(k - \tau_{sc,k} - \tau_i)$ 、 $u(k - \tau_{sc,k} - \tau_i)$ 与 $u(k - \tau_{sc,k})$ 估计出被控对象 $k - \tau_{sc,k}$ 时刻的系统状态 $\hat{x}(k - \tau_{sc,k})$ ，基于模型估计的外部输入根据经验设为：

$$\hat{u} = 0.3u(k - \tau_{sc,k} - \tau_i) + 0.7u(k - \tau_{sc,k}) \quad (5-1)$$

估计过程分为基于模型的时间更新与测量更新，量测更新使用前一次估计的误差 e_i 进行补偿，该误差定义为：

$$e_i = y(k - \tau_{sc,k} - \tau_i) - C\hat{x}(k - \tau_{sc,k} - \tau_i) \quad (5-2)$$

补偿的增益 K 使该估计误差收敛，该过程定义如下：

$$\hat{x}(k - \tau_{sc,k} - \tau_i + j + 1) = A\hat{x}(k - \tau_{sc,k} - \tau_i + j) + B\hat{u} + Ke_i \quad \text{for}(j = 0 \cdots (\tau_i - 1)) \quad (5-3)$$

之后更新 e_i 为新的估计误差：

$$e_i = y(k - \tau_{sc,k}) - C\hat{x}(k - \tau_{sc,k}) \quad (5-4)$$

通过以上过程就估计获得了被控对象发送数据包时刻 $k - \tau_{sc,k}$ 较为精确的状态 $\hat{x}(k - \tau_{sc,k})$ ，之后将根据该次发送的数据包的时延进行补偿，以估计出 k 时刻的被控对象状态 $\hat{x}(k)$ ，并基于该状态对 k 时刻之后未接收到新的数据包的时刻 $k + \tau_f$ 进行状态预测 $\bar{x}(k + \tau_f)$ 。估计与预测过程分为基于模型的时间更新与测量更新，量测更新使用发送数据包时刻 $k - \tau_{sc,k}$ 的误差 e_d 进行补偿，该误差定义为：

$$e_d = y(k - \tau_{sc,k}) - C\bar{x}(k - \tau_{sc,k}) \quad (5-5)$$

补偿的增益 K 使该估计误差收敛，该过程定义如下：

$$\hat{x}(k - \tau_{sc,k} + j + 1) = A\hat{x}(k - \tau_{sc,k} + j) + Bu(k - \tau_{sc,k}) + Ke_d \quad \text{for}(j = 0 \cdots (\tau_{sc,k} - 1)) \quad (5-6)$$

之后对未来时刻的状态进行预测，该过程定义如下：

$$\begin{cases} \bar{x}(k + 1) = A\hat{x}(k) + Bu(k - \tau_{sc,k}) + Ke_d & \tau_f = 1 \\ \bar{x}(k + \tau_f) = A\bar{x}(k + \tau_f - 1) + Bu(k - \tau_{sc,k}) + Ke_d & \tau_f \neq 1 \end{cases} \quad (5-7)$$

之后控制器将在每个采样周期时间点发送依据状态预测值与状态反馈参数计算得到的反馈量，并对反馈量进行限幅处理在 -20V 至 30V 的范围内，使用该方案是为了补偿状态观测器直接估计得到的滞后状态，防止发送滞后的反馈量。如图 5-6 所示，为采用该补偿方案与未采用时的小车动态性能对比图。从图中的曲线可以看出，在相同网络条件下，控制器距离为 35m 时该补偿方案使小车跟踪曲线消除了抖动与反向移动，动态性能良好，控制器距离为 40m 时该补偿方案使小车跟

踪曲线消除了剧烈震荡，虽然跟踪曲线有较为明显的超调，但消除了不稳定的发散，能够有效跟踪系统的参考输入。图 5-7 与表 5-4 为有与无数据包发送间隔补偿下的小车位置跟踪效果与系统性能对比，可看出控制器采用状态观测器与基于数据包时延的状态估计将引起系统超调与震荡，而在相同网络条件下，将状态观测器替换为基于数据包发送间隔的补偿能够大大提升系统的动态性能与稳定性。

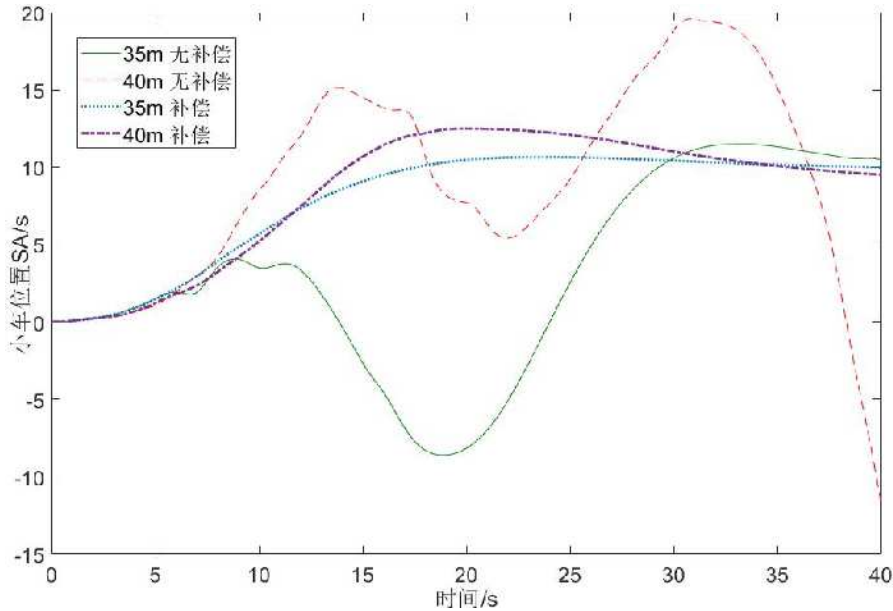


图5-6 无补偿与有补偿下的小车位置跟踪效果对比图

Figure 5-6. Comparison of the position tracking effect of the car without compensation and compensation

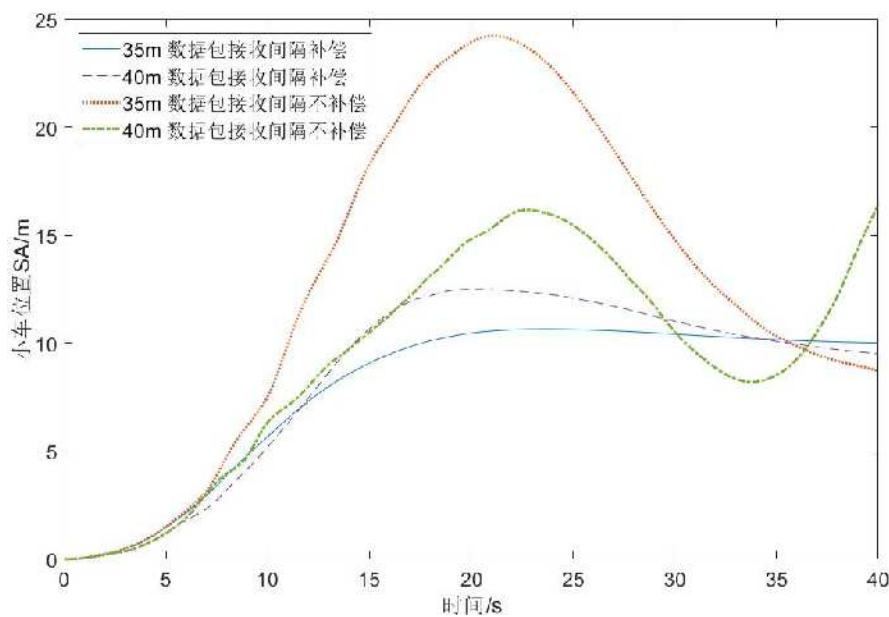


图5-7 有与无数据包发送间隔补偿下的小车位置跟踪效果对比图

Figure 5-7. Comparison of car position tracking effect with and without packet transmission interval compensation

表 5-4 基于数据包发送间隔与时延的补偿方案下的系统性能

Table 5-4. System performance based on compensation scheme for packet transmission interval and delay

控制器距离	是否补偿接收的相邻数据包的发送间隔	系统性能			
		超调量	丢包率	整回路平均时延	建立连接时长
35m	是	6.49%	0.18%	0.0914s	0.0514s
40m	是	24.89%	5.07%	0.616s	3.0584s
35m	否	142.3%	0.18%	0.0914s	0.0514s
40m	否	X	5.07%	0.616s	3.0584s

5.2 通信资源受限网络控制系统主动补偿方法的仿真测试

5.2.1 系统描述与控制方案

对存在通信资源受限且具有多个传感器的网络化控制系统而言，因其在结构组成上与一般网络化控制系统有差别，每个传感器发送给控制器的信息可能需要经由不同的数据包进行传输^[46]，术语“多包传输”用于描述这类特定的网络化控制系统。如图 5-8 所示为多包传输网络化控制系统的框图。

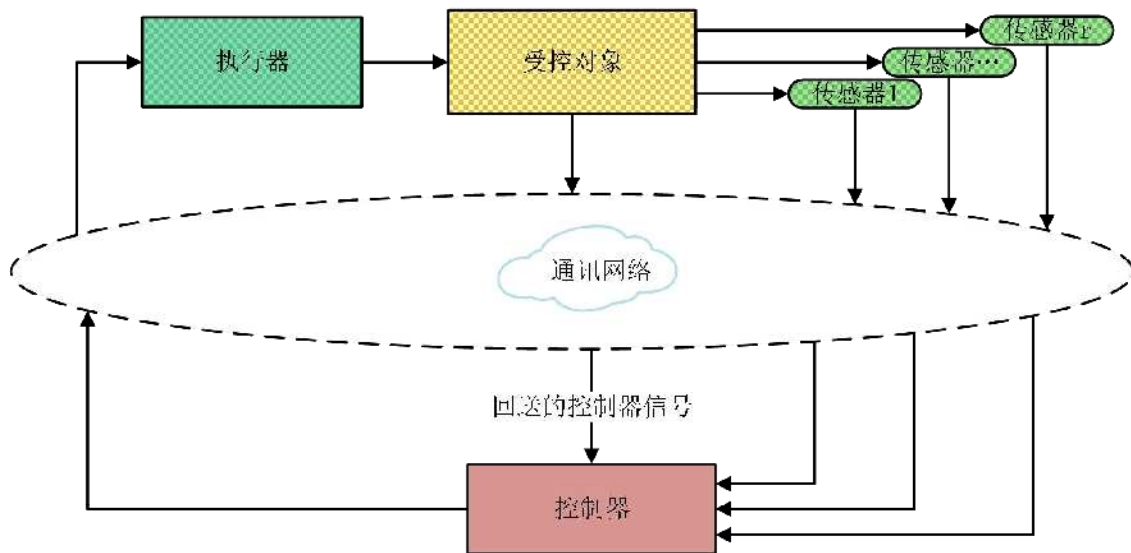


图5-8 由地理隔离造成的多包传输框图

Figure 5-8. Multi-packet transmission block diagram caused by geographical isolation

被控对象为一个离散线性系统：

$$x(k+1) = Ax(k) + Bu(k) \tag{5-8}$$

其中, $x \in R^n, u \in R^r, A \in R^{n \times n}, B \in R^{n \times m}$ 。假设控制器可以直接获得或通过状态估计获得被控对象状态, 状态量都是经由不同的传感器单独进行传输, 因而控制量是在 k 时刻基于传感器信息计算获得的, 该过程可表示为:

$$x_{\tau_{sc,k}}(k) = [y_1^T(k - \tau_{sc,k}^1) \cdots y_r^T(k - \tau_{sc,k}^r)]^T \quad (5-9)$$

其中, $y_i(k - \tau_{sc,k}^i)$, $i=1,2,3 \cdots, r$ 为 k 时刻控制器从第 i 个传感器接收的状态分量, $\tau_{sc,k}^i$ 是状态分量 $y_i(k - \tau_{sc,k}^i)$ 的时延。

如果 k 时刻控制器从传感器收到包含所有独立状态的数据包, 如果因数据包时延大小不同而采用统一的最大时延 $\tau_{sc,k}^{\max} = \max\{\tau_{sc,k}^i : i=1, \dots, r\}$, 这意味着控制系统可用的状态量时延是小于计算所用的实际时延, 或者由于包含某一状态的数据包在该时刻没有收到, 那么该时刻的状态量将无法使用。为解决这两种问题, 因此首先需要在控制器根据不同时延的传感器数据对被控对象状态进行重建。

重建状态的方案为: 重建最新接收的状态分量的发送时刻的被控对象的状态 $\bar{x}(k - \tau_{sc,k}^{\text{last}})$, 这里 $\bar{x}(k - \tau_{sc,k}^{\text{last}})$ 表示的状态量是包含了预测值与测量值的矩阵, 即:

$$\tau_{sc,k}^{\text{last}} := \text{last}\{\tau_{sc,k}^i : i=1, \dots, r\} \quad (5-10)$$

为简单起见, 这里定义 $\tau_{sc,k}^* = \tau_{sc,k}^{\text{last}}$, 同样也定义 $\bar{x}(k - \tau_{sc,k}^*) = \bar{x}(k - \tau_{sc,k}^{\text{last}})$ 。

同时把执行器实际使用的控制量反馈给控制器, 如图 5-8 所示, 同时假定反馈控制量的时延比传感器的时延要小, 即:

$$\tau_{sc,k}^u \leq \tau_{sc,k}^*, \forall k \quad (5-11)$$

设从状态 $\bar{x}(k - \tau_{st})$ 开始重构 $\bar{x}(k - \tau_{sc,k}^*)$, 且 $\tau_{st} > \tau_{sc,k}^*$, 当 $1 \leq l \leq \tau_{st} - \tau_{sc,k}^*$ 时, 状态分量的预测值定义如下:

$$\hat{y}_i(k - \tau_{st} + l) = \sum_{j=1}^r A^{ij} \bar{y}_j(k - \tau_{st} + l - 1) + B^i u(k - \tau_{st} + l - 1) \quad (5-12)$$

这里 $\bar{y}_j(k - \tau_{st} + l)$ 的定义有两种不同的形式, 当传感器发送过来的被控对象状态分量测量值可用时, 使用该测量值来定义 $\bar{y}_j(k - \tau_{st} + l)$, 如果被控对象的状态分量测量值不可用, 则将预测值作为被控对象的状态分量, 为了表述不产生歧义这里用下面的式子来表示 $\bar{y}_j(k - \tau_{st} + l)$:

$$\begin{cases} y_j(k - \tau_{st} + l), & \text{如果 } y_j(k - \tau_{st} + l) \text{ 可用} \\ \hat{y}_j(k - \tau_{st} + l), & \text{其他情况} \end{cases} \quad (5-13)$$

并在接收到可用的被控对象状态分量测量值时, 使用测量值和预测值共同组成的混合状态量作为 $\bar{x}(k - \tau_{sc,k}^*)$, 并以此状态量来计算控制序列。

且根据 (5-11) 可知, 当 $1 \leq l \leq \tau_{st} - \tau_{sc,k}^*$ 时, 系统的控制量 $u(k - \tau_{st} + l - 1)$ 总是

可用且是被控对象的最新输入。

最后进行基于数据包的 MPC 控制器设计，这种基于主动补偿的方案中，目标函数的定义如下：

$$\min_{U(k|k-\tau_{sc,k}^*)} J_{k,\tau_{sc,k}^*} = \|\bar{X}(k|k-\tau_{sc,k}^*)\|_Q^2 + \|U(k|k-\tau_{sc,k}^*)\|_R^2 \quad (5-14)$$

这里 $J_{k,\tau_{sc,k}^*}$ 是系统在时刻 k 的目标函数， Q 、 R 分别为系统的状态和控制加权

矩阵，系统的控制序列 $U(k|k-\tau_{sc,k}^*)$ 定义如下：

$$U(k|k-\tau_{sc,k}^*) = [u(k-\tau_{sc,k}^* | k-\tau_{sc,k}^*) \cdots u(k-\tau_{sc,k}^* + N_u - 1 | k-\tau_{sc,k}^*)]^T \quad (5-15)$$

系统状态轨迹的预测量用 $\bar{X}(k|k-\tau_{sc,k}^*)$ 表示：

$$\bar{X}(k|k-\tau_{sc,k}^*) = [\bar{x}(k-\tau_{sc,k}^* + 1 | k-\tau_{sc,k}^*) \cdots \bar{x}(k-\tau_{sc,k}^* + N_p | k-\tau_{sc,k}^*)]^T \quad (5-16)$$

其中 N_p 是预测时域，是控制时 N_u 域，通常 $N_p \geq N_u$ ，这里假设

$$N_u > \max_{k \geq 1} \{\tau_{sc,k}^* + \tau_{ca,k}^*\}。$$

求解目标函数得预测的最优控制序列为：

$$U(k|k-\tau_{sc,k}^*) = K_{\tau_{sc,k}^*}^* \bar{x}(k-\tau_{sc,k}^*) \quad (5-17)$$

其中 $K_{\tau_{sc,k}^*}^*$ 为最优反馈控制律，当执行器收到 $U(k|k-\tau_{sc,k}^*)$ 之后就会依据当前控制器到执行器的时延从控制序列中挑选一个控制量作用于执行器。执行器挑选出控制量的过程定义如下：

$$u(k + \tau_{ca,k}^*) = U_{\tau_k^*}^*(k|k-\tau_{sc,k}^*) \quad (5-18)$$

这里 $\tau_k^* := \tau_{sc,k}^* + \tau_{ca,k}^*$ 。

该方案的控制结构如图 5-9 所示。

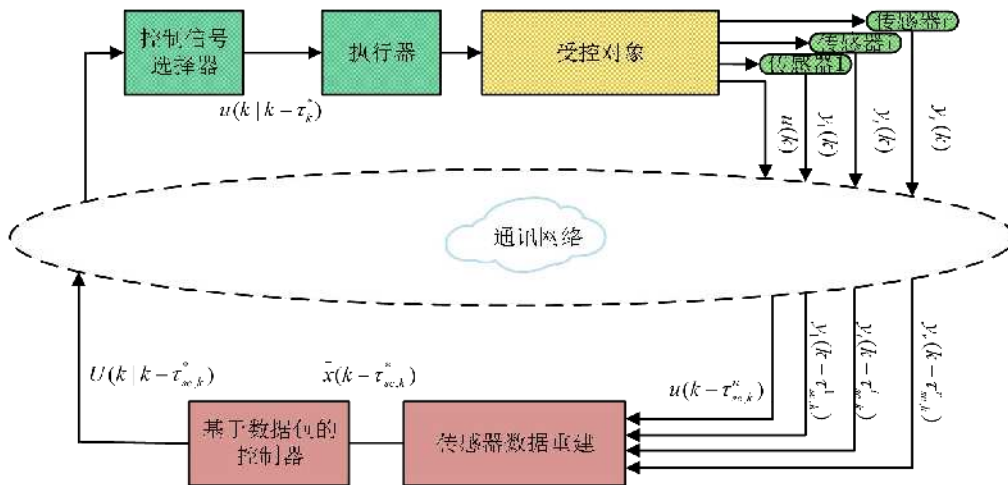


图5-9 基于模型的多包传输网络化控制系统主动补偿方案示意图

Figure 5-9. Schematic diagram of active compensation scheme for multi-packet transmission networked control system based on model

5.2.2 基于 TrueTime 的仿真

首先在 MATLAB 上的 TrueTime 工具箱搭建仿真网络进行仿真，本次实验挑选的对比方法为 LQR 方法，在实验参数的处理上，为控制单一变量，除了控制方法外的所有条件都相同。

本次仿真中考虑了如下系统^[47]：

$$\dot{X} = \begin{pmatrix} -3 & -7 & -5 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} X + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} u \quad (5-19)$$

$$Y = (0 \ 1 \ 2)X$$

将其以 5ms 的采样周期离散化后可得如下的离散状态空间方程：

$$x(k+1) = \begin{pmatrix} 0.9850 & -0.0348 & -0.0248 \\ 0.0050 & 0.9999 & -0.0001 \\ 0 & 0.0050 & 1 \end{pmatrix} x(k) + \begin{pmatrix} 0.0050 \\ 0 \\ 0 \end{pmatrix} u(k) \quad (5-20)$$

被控对象的三个状态分别通过不同传感器以周期 5ms 采集，并经过各自独立信道发送至控制器，并给数据包打上时间戳以获得数据包的时延，由于系统是离散，分别将前向通道与反馈通道时延除以采样周期 T 获得对应时延的离散步长。

在仿真的参数设置上，LQR 方法的 Q 为单位矩阵， R 为 0.1，系统的三个初始状态分量都设为 -1，传感器工作模式设为周期型采样，控制器与执行器都设为事件触发方式，信道路径衰减模型为对数距离衰减模型。通信网络的类型设为 802.11b 网络，考虑到实际网络运行时的情况，这里也考虑了丢包率并设其设为 5%，数据的传输速率为 800000bits/s。图 5-10 为使用了两种方法的系统状态量随时间变化的对比图，图 5-11 为两种方法在当前网络情况下的系统传感器到控制器的时延 τ_{sc} ，图 5-12 为两种方法在当前网络情况下的系统控制器到执行器的时延 τ_{ca} 。从图中曲线可看出，该补偿方法能在保证系统稳定性的前提下使系统达到更好的控制性能。

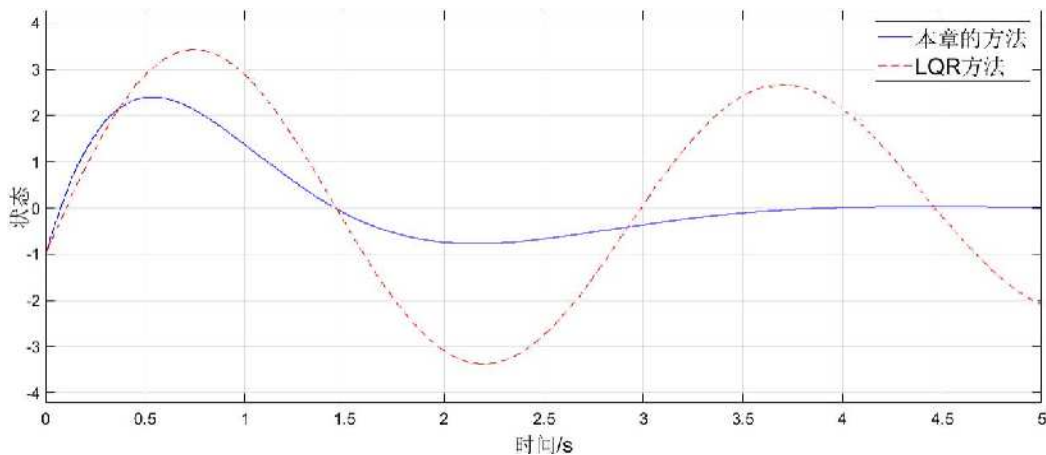


图5-10 系统状态图

Figure 5-10. System state diagram

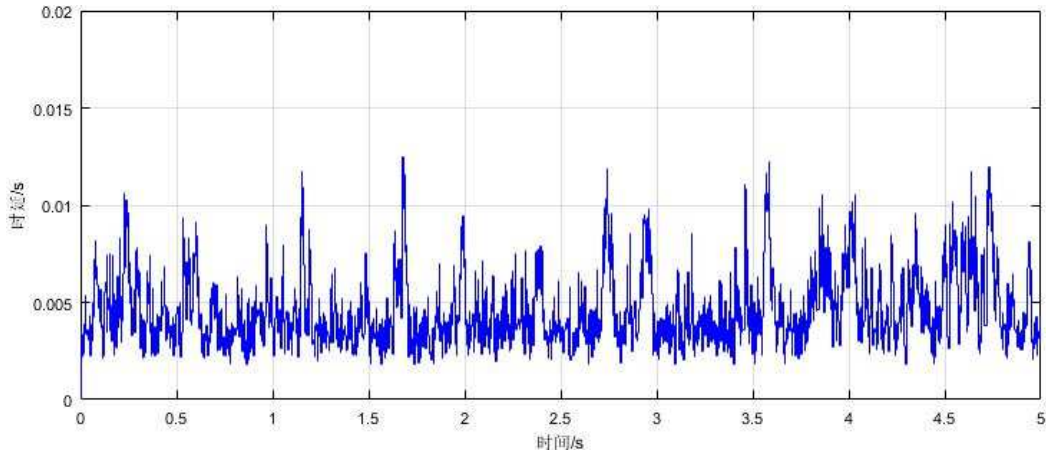


图5-11 传感器到控制器的时延分布图
Figure 5-11. Sensor to controller delay profile

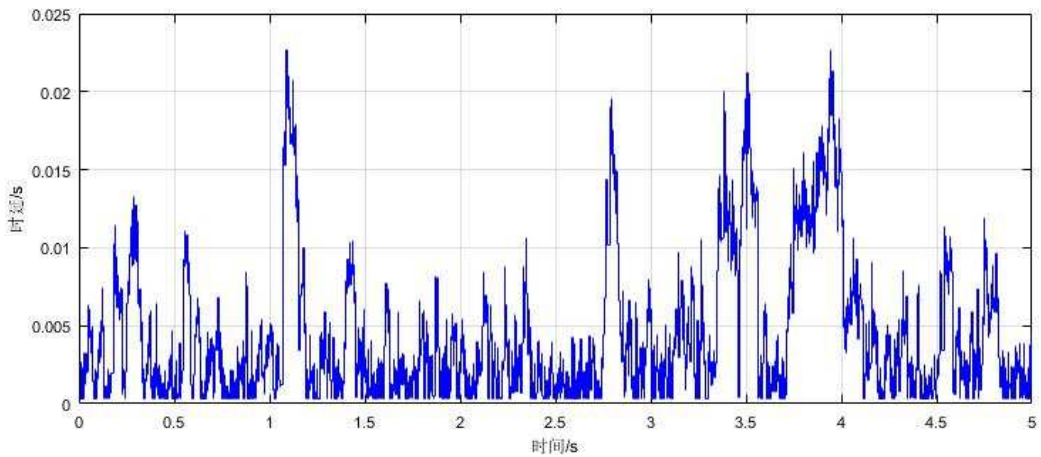


图5-12 控制器到执行器的时延分布图
Figure 5-12. Controller to actuator delay profile

5.2.3 基于 STM32 的测试

此外在两块 STM32 开发板间建立 WIFI UDP 连接以对该方案进行测试，测试的具体步骤如下：

(1) 选择两块 STM32 开发板并都接入 ESP8266WIFI 模块，这里选用 STM32F407 作为被控对象，STM32F429 作为控制器。

(2) 编写在两块开发板间建立 WIFI UDP 连接的驱动程序，注意将 WIFI 模块设为透传模式以屏蔽底层协议消息。另外在被控对象开发板驱动程序中设置 3 个定时器，定时器 TIM3 定时周期为 5ms，每 5ms 通过定时中断仿真一次被控对象的系统状态更新；定时器 TIM7 定时周期为 1ms 以反复查询并读取数据接收区的数据，通过判断接收连续 2 个字符之间的时间差不大于 1ms 来决定是不是一次连续的数据包的数据；LQR 算法下定时器 TIM5 定时周期为 30ms，每 30ms 通过定时中断将被控对象的系统状态等变量通过 WIFI 发送给控制器，本章补偿方案定时器

TIM5 定时周期也为 30ms，但每次中断内间隔 30ms 分 3 次发送数据包，则代表同一状态的数据包发送周期实际上为 90ms。控制器开发板驱动程序中仅设置定时器 TIM7，其作用与被控对象的定时器 TIM7 作用相同，当控制器开发板接收到数据后将触发控制量求解事件并立即反馈给被控对象开发板。

(3) 在 LQR 算法下不需要两块 STM32 间时钟同步，而本章设计的补偿方案为了通过时间戳获得开发板间数据包传输时延需要建立两块 STM32 间时钟同步。方案为在 PC 机系统时钟的秒级更新事件发生时将系统时间通过串口发送给 STM32，STM32 接收并更新 RTC 时钟以建立 PC 与 STM32 的时钟同步，这样两块 STM32 的 RTC 时钟都同步为 PC 的系统时钟，也就建立两块 STM32 之间的时钟同步。

(4) 本章补偿方案需要获得微秒级时延，因此将 RTC 时钟的异步分频系数设为 0X00，同步分频系数设为 0X7FFF，这样亚秒时间精度就为 1/32768 秒，能轻松获得微妙级时间；本章设计的补偿方案的 C 程序通过 Simulink 代码生成的方式获得；另外为确保 STM32 接收数据的完整性，加入数据校验对接收到的数据进行过滤。

(5) 最后将两个 STM32 放置在相距 10m 位置进行测试，如图 5-13 所示。测试结果如图 5-14 所示，测试结果与使用 TrueTime 工具箱的仿真结果基本一致。

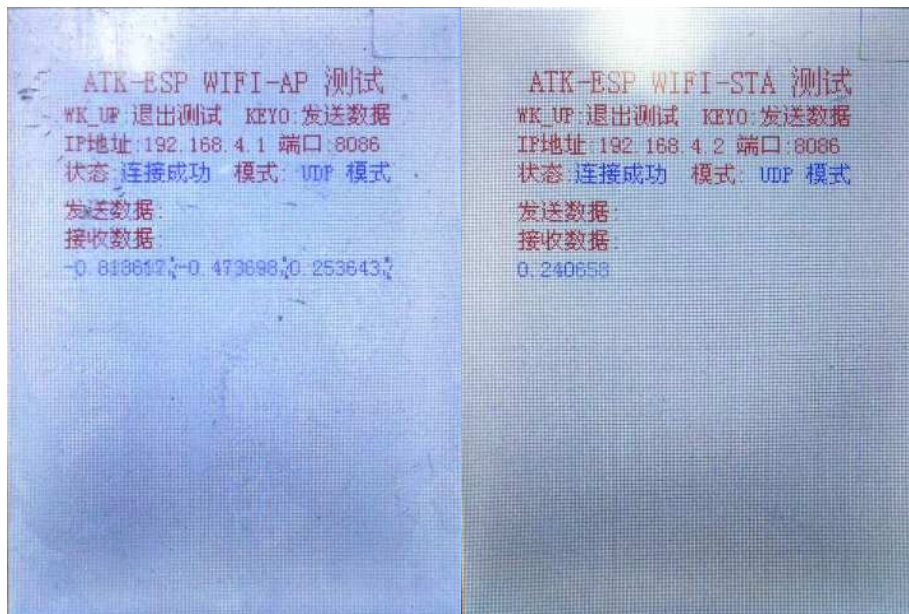


图5-13 STM32使用WIFI UDP下控制系统测试

Figure 5-13. STM32 control system test using WIFI UDP

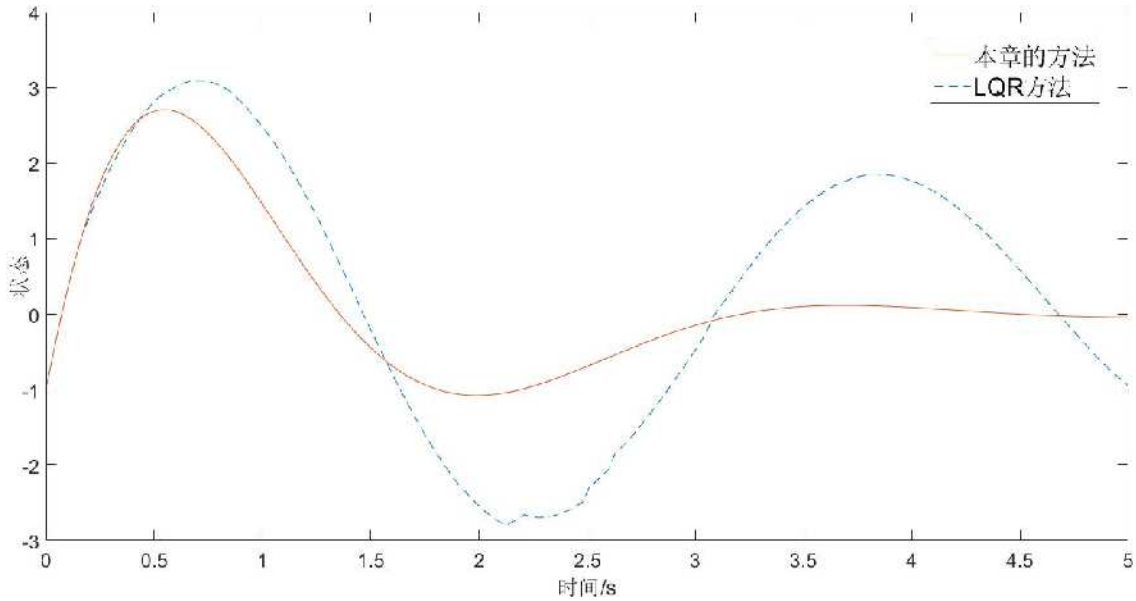


图5-14 STM32使用WIFI UDP下测试的系统状态图

Figure 5-14. System state diagram for STM32 test using WIFI UDP

5.2.4 基于协同仿真平台的仿真

类似于使用 truetype 工具箱进行仿真，使用本文中的平台进行仿真测试，同样的，使用相同实验参数下的 LQR 方法作为对比，仿真场景的参数设定如下表 5-5 所示。

表 5-5 多包传输网络控制系统仿真参数设置

Table 5-5. Multi-packet transmission network control system simulation parameter setting

参数列表	所选参数
节点个数	2 个
单步大小	10ms
仿真精度	1ms
仿真时长	5 秒
仿真驱动方式	事件驱动
信道路径传播模型	对数距离衰减模型
信道噪声模型	NormalRandom plus Nakagami (慢衰落叠加快衰落模型)
发射器发射功率	15dBm
接收器功率探测阈值	-110dBm
物理层协议	802.11b
节点移动模型	常位置模型
传输层协议	UDP 与 TCP
路由协议	aodv
物理层传输速率模式	常速率模式 DsssRate11Mbps

控制器被放置在距被控对象 15m 的位置，控制器与被控对象自身的状态更新周期为 1ms 并进行对应的离散化处理。被控对象的三个状态分别通过不同传感器以周期 10ms 采集，三个初始状态分量都设为-1，控制器的预测时域和控制时域都为 $5 \times (\tau_{step,sc,k} + 15)$ ms，控制器与执行器也通过打包的时间戳处理获得时延的离散步长。接下来的图 5-15 到图 5-24 展示了两种协议下使用如上两种方法得到的系统状态和前向通道与反馈通道的时延。

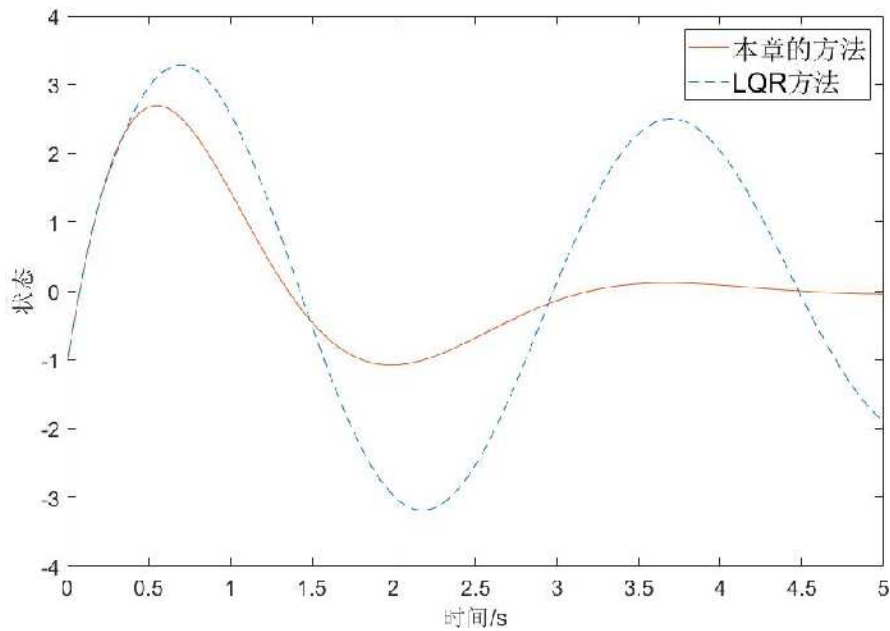


图5-15 UDP协议下系统状态图

Figure 5-15. System state diagram under UDP protocol

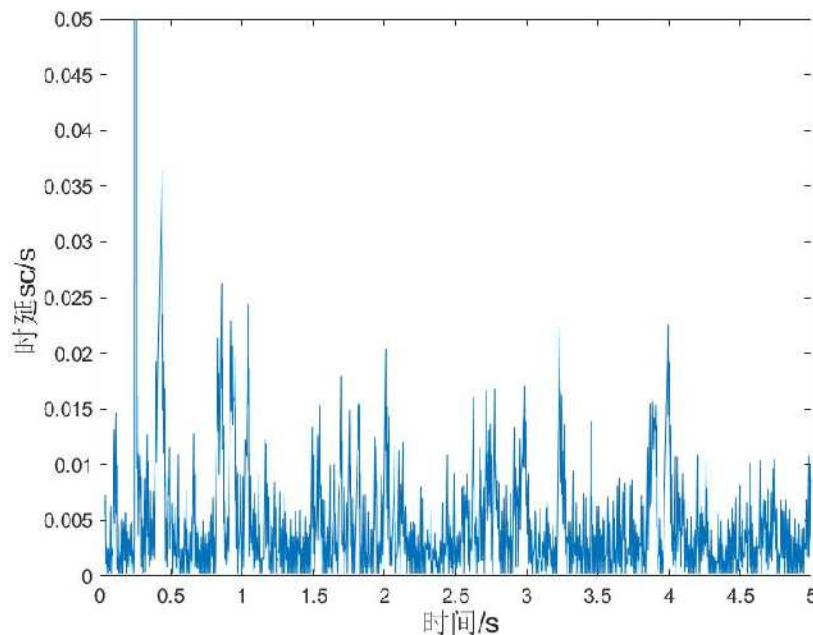


图5-16 UDP协议下使用补偿方案的传感器到控制器的时延分布图

Figure 5-16. Sensor-to-controller delay profile using compensation scheme under UDP protocol

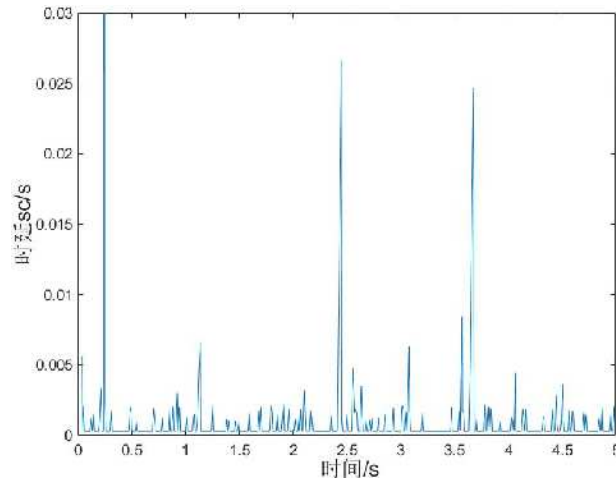


图5-17 UDP协议下使用LQR方案的传感器到控制器的时延分布图

Figure 5-17. Time delay profile of sensor-to-controller using LQR scheme under UDP protocol

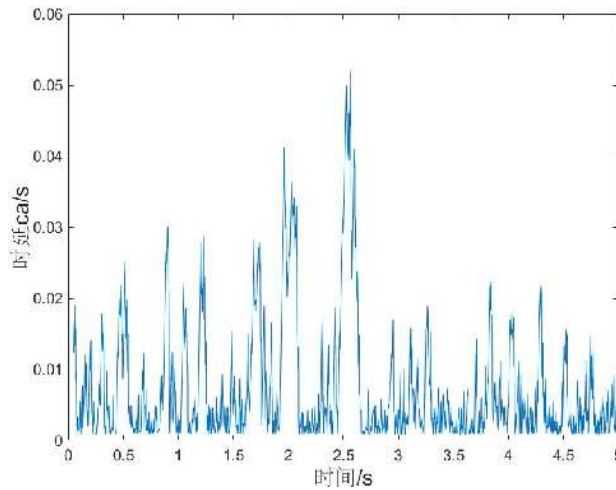


图5-18 UDP协议下使用补偿方案的控制器到执行器的时延分布图

Figure 5-18. Controller-to-executor delay profile using compensation scheme under UDP

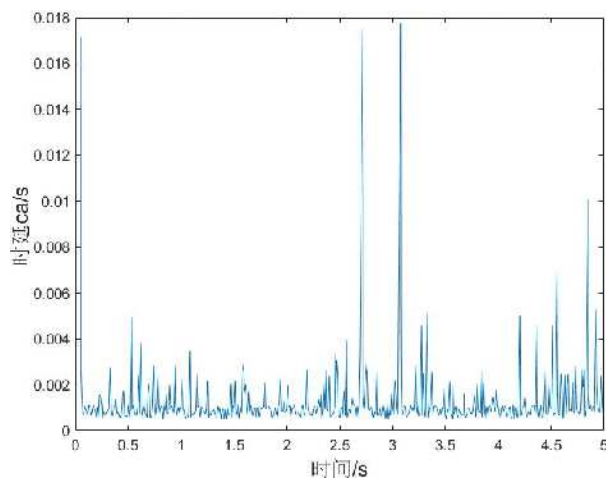


图5-19 UDP协议下使用LQR方案的控制器到执行器的时延分布图

Figure 5-19. Delay profile of controller to actuator using LQR scheme under UDP protocol

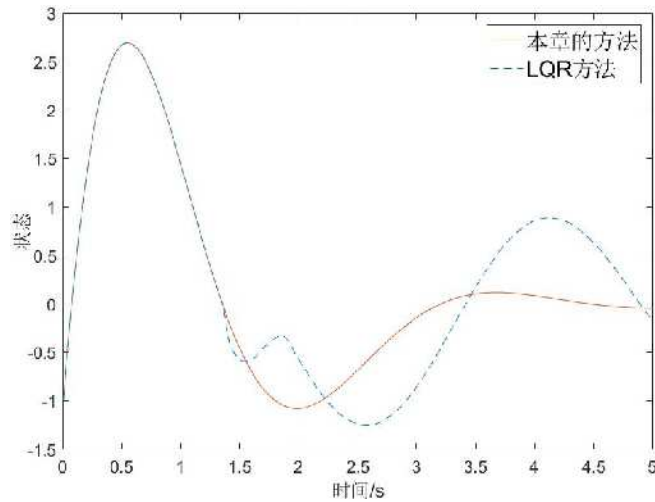


图5-20 TCP协议下系统状态图

Figure 5-20. System state diagram under TCP protocol

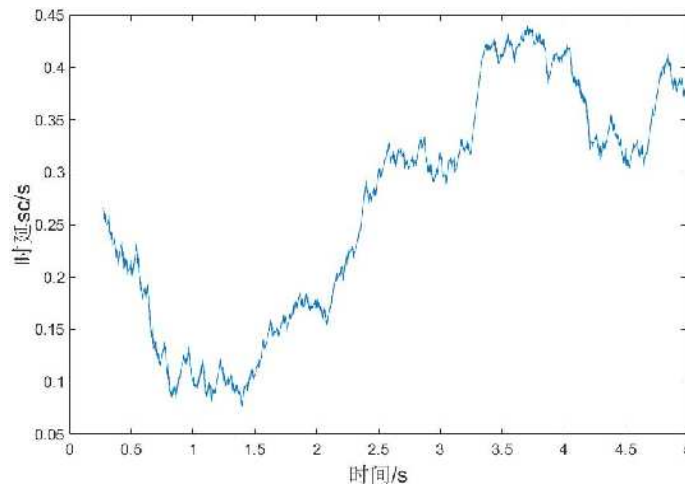


图5-21 TCP协议下使用补偿方案的传感器到控制器的时延分布图

Figure 5-21. Time delay profile of sensor-to-controller using compensation scheme under TCP

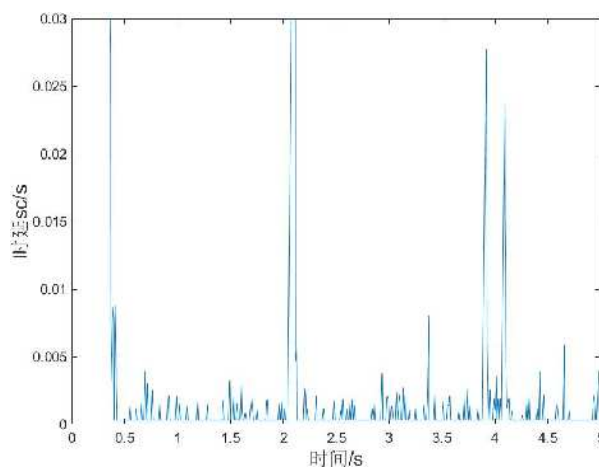


图5-22 TCP协议下使用LQR方案的传感器到控制器的时延分布图

Figure 5-22. Delay profile of sensor-to-controller using LQR scheme under TCP protocol

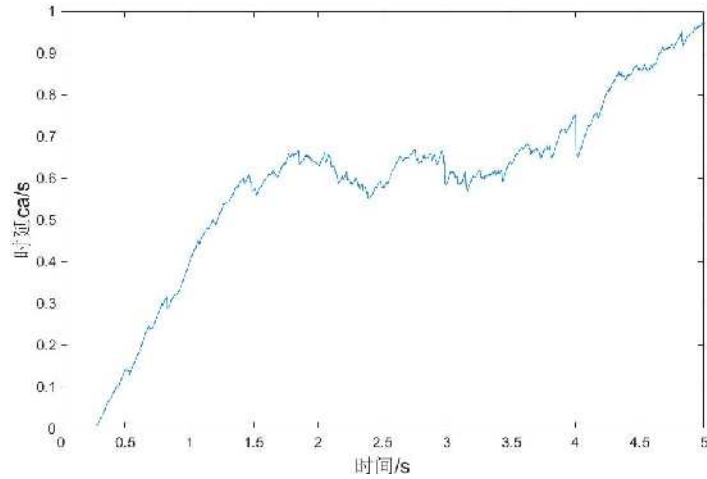


图5-23 TCP协议下使用补偿方案的控制器到执行器的时延分布图

Figure 5-23. Controller to actuator delay profile using compensation scheme under TCP

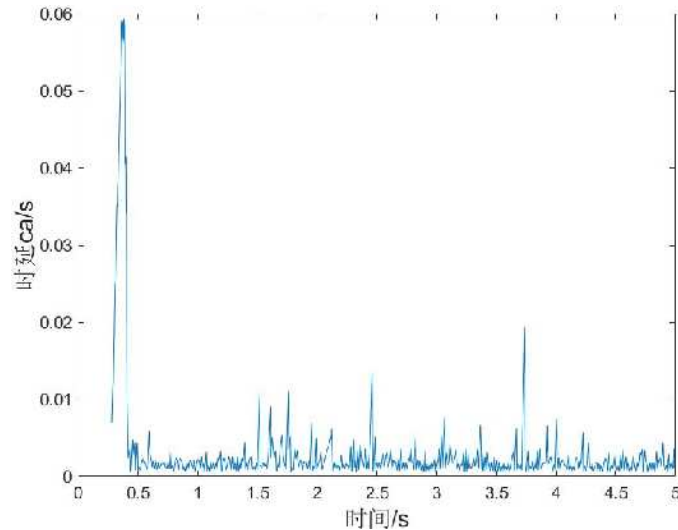


图5-24 TCP协议下使用LQR方案的控制器到执行器的时延分布图

Figure 5-24. Delay profile of controller to actuator using LQR scheme under TCP protocol

由这一系列图可看出，传感器多包传输会引起明显的信道资源受限，使数据包传输时延增大，而该补偿方法可以有效的提升该情况下系统的控制性能，在不同网络传输层协议与传输时延下系统的控制效果都能得到有效提升，验证了该补偿方案的通用性。同时从两种协议的系统状态图和传输时延图可看出，因被控对象采样频率较快，导致数据包以较快频率发送的情况下，由于 TCP 协议存在流量控制和拥塞控制机制限制网络信道中数据包的数量，同时还存在数据包超时与丢包重传机制，以确保每个数据包能够被有效接收且保证数据包的收发顺序一致，这两种机制导致信道网络负载增大进而会导致网络信道拥塞以及建立连接时间变长，相比与采用 UDP 协议时时延较大、传输速度较慢，控制实时性不如 UDP 协议良好，因而网络化控制系统的传输层协议建议使用 UDP 协议。

从以上两个仿真实例可看出，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 TrueTime 工具箱仿真和利用 STM32 测试的结果基本一致，同时相比于使用 TrueTime 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，相比于使用 TrueTime 工具箱用户在高层网络协议和节点移动模型等方面具有更多选择权限，并能够对网络化控制系统的性能予以针对性的分析与改进。

5.3 本章小结

本章将在搭建的仿真平台上分别仿真了桥式吊车远程控制系统与通信资源受限控制系统，并测试平台的相关的功能。桥式吊车远程控制系统的仿真体现了控制器与被控对象之间距离拉大降低了小车对参考输入的跟踪性能，为提升跟踪性能提出了基于传感器到控制器传输时延的状态预测补偿方案，通信资源受限控制系统展现了由于传感器多包传输而造成的信道资源受限的情况，并提出了多时延主动补偿方案以提升系统的控制性能，并将该系统在该仿真平台的仿真结果与使用 STM32 实验和 TrueTime 仿真的结果进行比较。针对以上两个实例的仿真验证了仿真平台的灵活与可靠性。

第六章 结论与展望

6.1 结 论

本文在分析网络仿真器技术和网络化控制系统仿真平台研究现状的基础上，将具有可扩展、易使用的 NS3 引入到网络化控制系统仿真中，构建了一种基于 NS3 和 MATLAB 的网络化控制系统协同仿真平台，为进一步的研究奠定了基础，对推进我国网络化控制技术有着积极的意义。本文主要完成的工作如下：

(1) 调研了网络仿真器技术和网络化控制系统仿真平台研究现状，并针对多仿真器协同仿真平台的特点与需求，提出了一种基于 NS3 和 MATLAB 的网络化控制系统协同仿真平台的设计方案。

(2) 设计了网络化控制系统协同仿真平台时间同步方案，由于协同仿真系统对实时性和时序的严格要求，以及两种仿真器的驱动方式存在差异，引入了时间同步的重要概念并提出了 3 种同步方案，并选择其中两种方案予以实现。其次，详细介绍了利用 TCP 套接字通信实现的同步信息交互模型。

(3) 在仿真平台软件系统设计上，首先在 NS3 中改造默认的仿真器类以实现外部驱动仿真器类，同时设计了应用类、助手类和相关结构体与仿真场景的脚本。接着开发了 MATLAB/Simulink 各模块，包括 MATLAB 仿真驱动模块、Simulink 控制系统模型、MATLAB 客户端组件以及利用 MATLAB GUI 开发的仿真平台交互界面。

(4) 在 NS3 和 MATLAB 的网络化控制系统协同仿真平台上分别搭建了桥式吊车远程控制系统与通信资源受限控制系统仿真模型并进行功能测试。在此基础上，验证了网络环境对控制系统的影响，并针对两种控制系统分别提出了对应的补偿方案以提升系统的控制性能。仿真结果表明，本仿真平台具有与 TrueTime 工具箱类似的功能，且具有较好的可靠性与稳定性，满足仿真平台设计的基本要求。

6.2 展 望

本文所设计的仿真平台还存在如下几点不足和改进之处：

(1) 现有平台的功能已基本满足仿真的需求，但是仍然存在许多优化的空间，比如可以使系统底层代码具有更强的健壮性，能够嵌入各类控制系统模型和设置

各类网络场景而不需要修改底层代码，同时系统内部各模块需要进一步解耦以提升系统的可扩展性，以及提升对多节点、多回路的复杂网络化控制系统的仿真支持。

(2) 车联网^[48]作为现有网络化控制系统应用研究的热点，可以搭建相应的控制系统模型与网络场景，以研究各类车联网应用例如：自动驾驶、车辆排的运行机制，进行相应的控制算法的设计与验证。

(3) 协同仿真平台的搭建不仅仅局限于本文所实现的方案，可使用新的适用协同仿真的标准与框架搭建新的平台或移植现有平台，包括符合 HLA 标准^[49]的 `prti`、`MAK rti` 等框架，以及在环仿真工具开发中广泛采用的接口标准 `FMI`^[50]。

致 谢

两年半前，我带着对知识的渴望来到浙江工业大学开启了我的硕士生涯，在这里生活、学习与成长，光阴似箭、岁月如梭，两年半的硕士生涯也即将结束。借助完成硕士论文之际，我要由衷的感谢在我攻读硕士期间给我学习和生活中提供过帮助的老师、同学以及家人朋友们。

首先要感谢我的导师赵云波教授，他始终从学生的角度出发，努力激发每个学生的潜能，引导学生充分发挥自己的主观能动性，提升他们在学习、科研和生活中独当一面的能力。同时他对教学和科学研究的严谨细致的精神也让我受益匪浅。同时我也要感谢各位课程的任课老师，是他们丰富了我的知识储备，拓展了我的视野面。有了这些老师的引导，我才能顺利的进入研究生的科研状态。

其次，感谢我的师兄韩康、何江涛、袁征、黄涛、姚俊毅与师姐李天舒，同学蒋传鹏、潘晓康、许德衡、李灏、林建武与学弟苏艺帆等在我遇到问题时不厌其烦的跟我讨论解决方案，由衷的感谢他们给我提供的帮助，也要感谢我的室友高佳斌、陈相旭、张硕，他们为我提供了一个舒适愉悦的生活环境，使我能够更好的投入到学习和科研当中去。

最后，我要感谢我的父母和家人一直以来给予我物质上和精神上的无私奉献和爱，有了他们的支持，我才能一路走到今天。

谨以此文献给所有关心和帮助过我的人，祝愿他们心想事成，万事如意！

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] Nilsson J , Bernhardsson B , Wittenmark B. Stochastic analysis and control of real-time systems with random time delays [J]. Automatica, 1998,34(1); 57-64.
- [3] Cetinkaya A , Ishii H , Hayakawa T. Networked Control under Random and Malicious Packet Losses [J]. IEEE Transactions on Automatic Control, 2016,PP(99); 2434-2449.
- [4] Wang Y L , Shi P , Lim C C , et al. Event-Triggered Fault Detection Filter Design for a Continuous-Time Networked Control System [J]. IEEE Transactions on Cybernetics, 2016,46(12); 3414–3426.
- [5] 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.
- [6] 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.
- [7] 王宝仁. 网络化运动控制系统多轴协同关键技术研究[D]. 山东大学, 2008.
- [8] 姬帅. 网络化运动控制系统的关键技术研究[D]. 山东大学, 2014.
- [9] 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, 2017,20(4):2774-2779.
- [10] 张云. NS3 网络模拟器路由协议集成设计与实现[D]. 电子科技大学, 2018.
- [11] 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.
- [12] Walsh G C , Ye H , Bushnell L G. Stability analysis of networked control systems [J]. IEEE Transactions on Control Systems Technology, 2002,10(3); 438-446.
- [13] Liu G P , Mu J X , Rees D , et al. Design and stability analysis of networked control systems with random communication time delay using the modified MPC [J]. International Journal of Control, 2006,79(4); 288-297.
- [14] Zhao Y B , Liu G P , Rees D. Integrated predictive control and scheduling co-design for networked control systems [J]. IET Control Theory and Applications, 2008,2(1); 7-15.
- [15] Zhao Y B , Liu G P , Rees D. Design of a Packet-Based Control Framework for Networked Control Systems [J]. IEEE Transactions on Control Systems Technology, 2009,17(4); 859-865.
- [16] Zhao Y B , Liu G P , Rees D. Packet-Based Deadband Control for Internet-Based Networked Control Systems [J]. IEEE Transactions on Control Systems Technology, 2010,18(5); 1057-1067.
- [17] Walsh G C , Beldiman O , Bushnell L. Asymptotic behavior of networked control systems [C]. // Proceedings of the 1999 IEEE International Conference on Control Applications. IEEE, 1999. 1448-1453.
- [18] Branicky M S , Phillips S M , Zhang W. Scheduling and Feedback Co-Design for Networked Control Systems (I) [C]. // Proceedings of the IEEE Conference on Decision and Control. IEEE, 2003. 1211-1217.
- [19] Dolk V S , Ploeg J , Heemels W P M H. Event-Triggered Control for String-Stable Vehicle Platooning [J]. IEEE Transactions on Intelligent Transportation Systems, 2017,PP(99); 1-15.

- [20] 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, 2003,23(3); 16-30.
- [21] Simon,G. Prowler:Probabilistic wireless network simulator[OL]. [2009-02-13]. <http://www.isis.vanderbilt.edu/Projects/nest/prowler/>.
- [22] Andreu, D. Implementation and performance evaluation of iee 802.15.4 protocol[D]. KTH, School of Electrical Engineering (EES), Automatic Control.2011.
- [23] Baldwin P, Kohli S, Lee E A, et al. Modeling of sensor nets in Ptolemy II [C]. // Proceedings of the Information Processing in Sensor Networks,2004. Third International Symposium on. IEEE, 2004. 359-368.
- [24] 周力, 吴在军, 孙军,等. 融合时间同步策略的主从式信息物理系统协同仿真平台实现[J]. 电力系统自动化, 2017,41(10):9-15.
- [25] Cn Z N E. Co-simulation Tools for Networked Control Systems [C]. // Proceedings of the 11th International Workshop on Hybrid Systems. Computation & Control. Springer-Verlag, 2008. 16-29.
- [26] Tong X. The Co-simulation Extending for Wide-area Communication Networks in Power System [C]. // Proceedings of the 2010 Asia-Pacific Power & Energy Engineering Conference. IEEE, 2010. 1-4.
- [27] Park S I, Savvides A, Srivastava M B. SensorSim: a simulation framework for sensor networks [C]. // Proceedings of the 3rd ACM international workshop on Modeling. analysis and simulation of wireless and mobile systems. ACM, 2000. 104-111.
- [28] Aminian B, Araujo J, Johansson M, et al. GISOO: A virtual testbed for wireless cyber-physical systems [C]. // Proceedings of the IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society. IEEE, 2013. 5588-5593.
- [29] Lin H, Veda S S, Shukla S S, et al. GECCO: Global Event-Driven Co-Simulation Framework for Interconnected Power System and Communication Network [J]. IEEE Transactions on Smart Grid, 2012,3(3); 1444-1456.
- [30] Matteo.Morelli. A System-Level Framework for the Design of Complex Cyber-Physical Systems from Synchronous-Reactive Models[D]. Scuola Superiore Sant'Anna.2015.
- [31] Cremona F, Morelli M, Natale M D. TRES: A Modular Representation of Schedulers, Tasks, and Messages to Control Simulations in Simulink [C]. // Proceedings of the 30th Annual ACM Symposium on Applied Computing. ACM, 2015. 1940-1947.
- [32] Heimlich O, Sailer R, Budzisz L. NMLab: A Co-simulation Framework for Matlab and NS-2 [C]. // Proceedings of the 2010 Second International Conference on Advances in System Simulation. SIMUL, 2010. 152-157.
- [33] Kohtamaki T, Pohjola M, Brand J, et al. PiccSIM Toolchain - design, simulation and automatic implementation of wireless networked control systems [C]. // Proceedings of the 2009 IEEE International Conference on Networking, Sensing and Control. IEEE, 2009. 49-54.
- [34] 李伟林, 张晓斌, 董延军. 电力系统综合仿真方法研究(一):VPNET(英文)[J]. 中国电机工程学报, 2012,32(13):95-102.
- [35] Kudelski M, Gambardella L M, Di Caro G A. RoboNetSim: An integrated framework for multi-robot and network simulation [J]. Robotics and Autonomous Systems, 2013,61(5); 483-496.
- [36] 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.
- [37] 陈寅, 宋杨, 费敏锐. 基于 Simulink 和 OPNET 的交互式联合仿真研究[J]. 系统仿真技术, 2011,7(3):242-247.
- [38] 李伟林, 张晓斌, 董延军. 电力系统综合仿真方法研究(二):MPNET 及实时 HIL 平台(英文)[J]. 中国电机工程学报, 2012,32(16):100-106.
- [39] Hopkinson K , Wang X , Giovanini R , et al. EPOCHS: A Platform for Agent-Based Electric Power and Communication Simulation Built From Commercial Off-the-Shelf Components [J]. IEEE Transactions on Power Systems, 2006,21(2); 548-558.
- [40] Al-Hammouri, Ahmad T. A comprehensive co-simulation platform for cyber-physical systems [J]. Computer Communications, 2012,36(1); 8-19.
- [41] 董振海. MATLAB 编译程序和外部接口[M]. 国防工业出版社, 2010.
- [42] 曹玲芝, 王新金, 魏尚北,等. 基于 DDE 和 Socket 技术的网络控制系统仿真平台[J]. 通信技术, 2009,42(8):30-33.
- [43] 闵秋应, 李津发. 无线通信技术在 LMSS 系统中的应用[J]. 自动化技术与应用, 2004,(3):46-51.
- [44] Pan Z , Xu Q , Chen C , et al. NS3-MATLAB co-simulator for cyber-physical systems in smart grid [C]. // Proceedings of the 35th Chinese Control Conference. IEEE, 2016. 9831-9836.
- [45] Bakhtin A , Volkov A , Muratchaev S. Development of MANET network model for space environment in NS3 [C]. // Proceedings of the 2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). IEEE, 2017. 111-114.
- [46] Sahebsara M. Optimal filtering in multiple channel networked control systems with multiple packet dropout [C]. // Proceedings of the 47th IEEE Conference on Decision and Control. IEEE, 2008. 3366-3371.
- [47] Chen C T. Linear system theory and design[M]. Holt: Rinehart and Winston, 1984.
- [48] Benin J , Nowatkowski M , Owen H. Vehicular Network simulation propagation loss model parameter standardization in ns-3 and beyond [C]. // 2012 Proceedings of IEEE Southeastcon. IEEE, 2012. 1-5.
- [49] 柴旭东, 李伯虎,等. 高层体系结构 HLA/RTI 及其实现综述[J]. 系统仿真学报, 1999,11(2):92-96.
- [50] Savicks V , Butler M , Colley J. Co-simulating Event-B and continuous models via FMI [C]. // Proceedings of the 2014 Summer Simulation Multiconference. ACM, 2014. Article No. 37.
- [51] 朱其新, 刘红俐, 胡寿松. 确定性网络控制系统的多步预测控制器设计[J]. 兵工学报, 2009,30(8):1124-1128.
- [52] 刘磊明, 童朝南, 武延坤. 一种带有动态输出反馈控制器的网络控制系统的 Markov 跳变模型[J]. 自动化学报, 2009,35(5):627-631.
- [53] 王天宝, 吴成东, 王玉龙. 基于多包传输的网络控制系统控制器设计[J]. 东北大学学报(自然科学版), 2013,34(2):157-161.
- [54] 朱其新. 网络控制系统的建模、分析与控制[D]. 南京航空航天大学, 2003.
- [55] 查利娟. 基于事件触发机制网络控制系统的若干问题研究[D]. 东华大学, 2017.
- [56] Kachan D. Integration of NS-3 with MATLAB/Simulink[D]. Luleå Tekniska Universitet.2010.

- [57] Choudhury A , Maszczyk T , Math C B , et al. An Integrated Simulation Environment for Testing V2X Protocols and Applications [J]. Procedia Computer Science, 2016,80; 2042-2052.
- [58] Zimmermann J , Stattelmann S , Viehl A , et al. Model-driven virtual prototyping for real-time simulation of distributed embedded systems [C]. // Proceedings of the 7th IEEE International Symposium on Industrial Embedded Systems. IEEE, 2012. 201-210.
- [59] 黄海. 网络控制系统实验平台建设及算法设计[D]. 浙江大学, 2007.
- [60] 夏元清, 李慧芳, 张金会. 控制与计算理论的交互: 云控制[J]. 指挥与控制学报, 2017,3(2):99-118.

作者简介

1 作者简历

1995年6月出生，重庆长寿。

2017年8月—2020年1月，浙江工业大学信息工程学院控制工程专业学习，获得工程硕士学位。

2 参与的科研项目及获奖情况

[1] 国家自然科学基金面上项目：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）

3 发明专利

[1] 赵云波，顾慧卿，苏艺帆，韩康.一种基于 NS3 和 MATLAB 的网络化控制系统协同仿真平台。（除导师外第一作者，201910596703.8，受理）

[2] 赵云波，顾慧卿，苏艺帆，韩康.基于 TCP 协议的 NS3 与 MATLAB 集成的联合仿真接口方法。（除导师外第一作者，201811396214.X，公开）

学位论文数据集

密 级*	中图分类号*	UDC*	论文资助
公开	TP391.9	681.5	国家自然科学基金
学位授予单位名称*	学位授予单位代码*	学位类型*	学位级别*
浙江工业大学	10337	工程硕士	全日制专业型硕士
论文题名*	基于 NS3 和 MATLAB 的网络化控制系统协同仿真平台设计		
关键词*	协同仿真平台, NS3 与 MATLAB, 网络化控制系统, 时间同步, 主动补偿		论文语种*
并列题名	Design of Cooperative Simulation Platform for Networked Control System Based on NS3 and MATLAB		中文
作者姓名*	顾慧卿	学 号*	2111703369
培养单位名称*	培养单位代码*	培养单位地址*	邮政编码*
浙江工业大学信息 学院	10337	杭州市西湖区留和 路 288 号	310023
学科专业*	研究方向*	学 制*	学位授予年*
控制工程	网络化控制	2.5 年	2020
论文提交日期*	2020 年 01 月		
导师姓名*	赵云波	职 称*	教授
评阅人	答辩委员会主席*	答辩委员会成员	
盲评	石崇源	赵云波, 宣琦, 周晓, 陈晋音	
电子版论文提交格式: 文本 (<input checked="" type="checkbox"/>) 图像 (<input type="checkbox"/>) 视频 (<input type="checkbox"/>) 音频 (<input type="checkbox"/>) 多媒体 (<input type="checkbox"/>) 其他 (<input type="checkbox"/>)			
电子版论文出版 (发布) 者	电子版论文出版 (发布) 地	版权声明	
论文总页数*	70		
注: 共 33 项, 其中带*为必填数据, 为 25 项。			