**硕士学位论文**

(工程硕士)

证券委托交易系统之消息中间件设计与实现

**THE DESIGN AND IMPLEMENTATION OF MESSAGE ORIENTED MIDDLEWARE FOR SECURITIES TRADING COMMISSION SYSTEM**

**王云龙**

**哈尔滨工业大学**

**2015 年 6 月**

国内图书分类号：TP311 学校代码：10213

国际图书分类号：621.3 密级：公开

**工程硕士学位论文**

证券委托交易系统之消息中间件设计与实现

|  |  |  |
| --- | --- | --- |
| 硕士研究 Th | ： | 王云龙 |
| 导 师 | ： | 刘远超 副教授 |
| 副 导 师 | ： | 石伟新 高级工程师 |
| 申 请 学 位 | ： | 工程硕士 |
| 学 科 | ： | 软件工程 |
| 所 在 单 位 | ： | 软件学院 |
| 答 辩 日 期 | ： | 2015 年 6 月 |
| 授予学位单位 | ： | 哈尔滨工业大学 |

Classified Index：TP311 U.D.C.: 621.3

Dissertation for the Master’s Degree in Engineering

**THE DESIGN AND IMPLEMENTATION OF MESSAGE ORIENTED MIDDLEWARE FOR SECURITIES TRADING COMMISSION SYSTEM**

|  |  |
| --- | --- |
| **Candidate：** | Wang Yunlong |
| **Supervisor：** | Associate Prof. Liu Yuanchao |
| **Associate Supervisor:** | Senior Engineer Shi Weixin |
| **Academic Degree Applied for：** | Master of Engineering |
| **Speciality：** | Software Engineering |
| **Affiliation：** | School of Software |
| **Date of Defence：** | June, 2015 |
| **Degree-Conferring-Institution：** | Harbin Institute of Technology |

摘 要

最近几年来，随着个人投资者直接参加证券业务的积极性提高，原有的证券行业交易系统已经表现出处理请求能力不足。开放式分布式的证券委托交易系统成为可以根本解决这一矛盾关键技术。但是如何有效的传递各个组件之间的消息成为关键问题。

本论文是在证券委托业务背景下，在分布式系统的环境下，解决如何稳定和高效的在客户端程序与交易柜台之间传递委托请求。解决的理论基础即是消息中间件。即提供节点之间可靠的消息传递并且保证有且只有一次，对传输的消息进行压缩从而提高性能，对组件间传输的消息进行加密并且组件之间建立连接时进行身份认证等手段提高消息中间件安全性。

论文分析对比了现有比较成熟的商用消息中间件产品，研究其通信模型和工作模型，分析其提供服务的亮点和不足之处，并结合分析证券交易业务数据交换通信协议的消息格式。提出适用于证券业务背景下的消息中间件产品需求。通过分析证券委托交易系统的部署结构，定义消息中间件说起的作用。然后进行总体设计与详细设计。在实现时采用C++与LUA结合方式编程，尽可能的平衡业务逻辑实现的便捷性和消息处理的高效性与稳定性。

本课题实现的消息中间件分为三个模块：业务数据展示模块、数据传输模块、柜台状态监测模块。业务数据展示模块提供控制客户端展示菜单项、页面样式和用户交互模式。数据传输模块保证客户端程序和交易柜台之间数据安全可靠稳定的交换，同时采用日志文件记录和中间件状态监测保证消息中间件的稳定性和问题可追溯性。柜台状态监测模块及时反馈消息中间件所处于的工作状态。

最后分别对各个模块设计相应的测试用例，并进行测试，最终测试结果表明消息中间件满足已定义的需求。并通过专门的测试工具证明在稳定性和性能上，达到上线运行的要求。可以作为产品投入到证券委托交易的业务中支持生产。

关键词：分布式；消息中间件；证券委托交易系统；多线程模式；混合编程

Abstract

In recent years, with the increase of individual investors to participate in securities business, the original Securities Trading Commission System has demonstrated the ability to deal with the problem of insufficient. Open distributed system of Securities Trading Commission System has become the key technology to solve this problem. But how to effectively transfer the messages between the various components becomes a key issue.

The thesis is to solve the problem of how to deal with the stability and efficiency in the distributed system under the background of the securities business. The theoretical basis is the Message Oriented Middleware. To ensure that between nodes provides reliable message transfer and has one and only once again, on the transmission of messages compression to improve performance, encryption and between the components to establish a connection for authentication so as to improve the safety of message transmission between components.

The paper analyze and compare the existing mature commercial Message Oriented Middleware products, researches on the communication mo del and operation model, analyse service highlights and shortcomings, and combining with the analysis of securities trading data exchange communication protocol message format analysis. The paper proposes the requirements of Message Oriented Middleware products in the context of securities business. Through the analysis of the structure of the Securities Trading Commission System, the role of Message Oriented Middleware is defined. Then defined the overall design and detailed design. In the implementation of C++ and LUA in the way of programming, as far as possible the balance of business logic to achieve the convenience and efficiency of message processing and stability.

In this paper, the Message Oriented Middleware is divided into three modules: business data display module, data transmission module, and counter state monitoring module. Business data display module control to client of displaying menu item, page style and user interaction model. Data transmission module to ensure client program and traded over the counter between data security and stable and reliable exchange. The log file records and middleware state monitoring ensure the Message Oriented Middleware stability and issues can be traced. The working status of the counter state monitoring module in time feedback the Message Oriented Middleware.

Finally, the test cases are designed and tested, and the results show that the Message Oriented Middleware meets the requirements of the definition. And through special testing tools to prove the stability and performance, to meet the requirements of on-line operation. Can be used as a product to support the production of securities entrusted business.

**Keywords:** Distributed; Message Oriented Middleware; Securities trading comission system; Multi thread mode; Mixed; Programming

目 录

[摘 要](#_Toc686432067) 3

[Abstract](#_Toc686432068) 4

[第1章 绪论](#_Toc686432069) 5

[1.1 课题来源及研究意义](#_Toc686432070) 5

[1.2 与课题相关的国内外研究综述](#_Toc686432071) 5

[1.2.1 消息中间件的定义及工作模式](#_Toc686432072) 5

[1.2.2 消息中间件优势](#_Toc686432073) 5

[1.2.3 国内外研究现状](#_Toc686432074) 6

[1.3 本论文的主要工作内容](#_Toc686432075) 6

[第2章 消息中间件需求分析与总体设计](#_Toc686432076) 7

[2.1 消息中间件的需求分析](#_Toc686432077) 7

[2.1.1 系统角色](#_Toc686432078) 7

[2.1.2 消息中间件功能性需求](#_Toc686432079) 7

[2.1.3 消息中间件非功能需求](#_Toc686432080) 8

[2.2 消息中间件的总体设计](#_Toc686432081) 8

[2.2.1 证券委托交易系统拓扑结构](#_Toc686432082) 8

[2.2.2 消息中间件总体架构](#_Toc686432083) 8

[2.2.3 消息结构定义](#_Toc686432084) 8

[2.2.4 功能模块划分](#_Toc686432085) 9

[2.3 本章小结](#_Toc686432086) 10

[第3章 消息中间件的详细设计与实现](#_Toc686432087) 10

[3.1 业务数据展示模块设计与实现](#_Toc686432088) 10

[3.1.1 页面与菜单项载入模块设计与实现](#_Toc686432089) 10

[3.1.2 交互模式模块设计](#_Toc686432090) 11

[3.2 数据传输模块设计](#_Toc686432091) 13

[3.2.1 数据传输模块设计](#_Toc686432092) 13

[3.2.2 柜台连接建立模块实现](#_Toc686432093) 15

[3.2.3 请求消息预处理模块实现](#_Toc686432094) 15

[3.2.4 请求消息处理模块实现](#_Toc686432095) 15

[3.2.5 应答消息返回模块实现](#_Toc686432096) 15

[3.2.6 日志文件记录模块实现](#_Toc686432097) 16

[3.2.7 中间件状态监测模块实现](#_Toc686432098) 16

[3.3 柜台监测模块设计](#_Toc686432099) 16

[3.3.1 柜台状态监测模块设计](#_Toc686432100) 16

[3.3.2 柜台异常处理模块设计](#_Toc686432101) 16

[3.4 本章小结](#_Toc686432102) 17

[第4章 消息中间件的测试](#_Toc686432103) 17

[4.1 功能测试](#_Toc686432104) 17

[4.1.1 测试环境](#_Toc686432105) 17

[4.1.2 测试用例](#_Toc686432106) 18

[4.2 非功能测试](#_Toc686432107) 21

[4.2.1 稳定性测试](#_Toc686432108) 21

[4.2.2 性能测试](#_Toc686432109) 21

[4.3 本章小结](#_Toc686432110) 22

[结论](#_Toc686432111) 22

[参考文献](#_Toc686432112) 22

[哈尔滨工业大学学位论文原创性声明和使用权限](#_Toc686432113) 23

[个人简历](#_Toc686432114) 24

# 第1章 绪论

## 1.1 课题来源及研究意义

随着国内经济的高速发展，证券交易逐渐活跃，网上证券委托交易也慢慢被个人投资者所热衷。个人投资者对直接开展金融交易活动的积极参与，整个证券行业比任何时候都需要分布式架构系统来应对海量的交易请求和数据处 理。多层次立体式的安全体系建设，不做区分的内外网Web服务、依赖各个相关系统的认证机制和对重要请求的负载均衡来应对数据处理的缓慢与请求的实时性要求的矛盾，这些重要的组件之间的传输要保证：可靠性，发出消息一定

要到达目的节点并且保证只有一次；安全性，消息不能在传输过程中被篡改[1]。在异介质的网络协议和操作系统之间，消息中间件就作为一个消息传输通道，可以保证可靠性和安全性。从而消息中间件可以有效的保证在分布式是系统中组件间的通信，保证分布式系统有效的对外提供服务。

本课题来源于浙江核新同花顺网络信息股份有限公司开发的证券委托交易系统中的重要组件——消息中间件。证券委托交易系统是覆盖主要证券业务的交易委托，信息获取发布平台。业务以股票为主，包括基金、期货、外汇；服务以委托交易为主，包括资讯获取、行情获取、个性化服务。证券委托交易系统的消息中间件是依据公司多年来对证券行业的业务发展的分析和预测，对证券行业相关系统多年来运营经验的总结和创新，在原有的中间件理论模型的基础上，提出的针对证券行业的消息中间件。根据消息中间件作为数据传输通道，可以有效屏蔽前端客户端的种类和后端各种交易柜台的接口不同，有利于企业二次开发，扩展以后证券业出现的新业务新规则支持，同时有利于对原有业务和规则由于证监会修改的支持。由于对于上述的修改只是涉及脚本层，可以共用已经实现的复杂的网络访问，所以可以有效缩短应用开发周期，减少程序设计与实现的复杂性，从而节约应用的开发成本。对于公司架构设计而言，可以简化现有系统之间进行集成的复杂性。尤其由于历史原因，各种技术使用，导致基于各种接口标准和协议开发而成。对于使用公司而言可以降低开发应用程序的失败可能性，从而保护已有的投资。

## 1.2 与课题相关的国内外研究综述

随着中国经济不断向好发展，改革力度不断的深入化细致化。这些都使得证券交易前所未有的活跃。今年4月20日下午14时30分，上海证券交易所股

票交易破万亿[1]。至收盘，深圳证券交易所成交6274.96亿。全天成交超过1.65

万亿。

同时也收到一人一户的政策放开影响，证券开户人数急剧增加。中登公司统计显示：“4月新增投资者497.53万，新增股票账户同比大增3727.17%，从 3

月份的486.89万户直接跳升1294.73万户。其中投资者新开A股账户数614430户，同时开通上海证券交易所和深圳深圳证券交易所的账户。其中，上海证券交易所环比上升230910户，每日新开A股账户数平均为342791户，涨幅为

206.39%；深圳深圳证券交易所环比上升162389户，A股账户数每日新开平均为271638户，涨幅为148.64%。上海证券交易所和深圳深圳证券交易所合计环比上升了1.7.13%，日均基金开户数为249116户“[2]。

移动互联网的快速发展并逐渐占据主导地位，越来越多的投资者需要随时随地的关注股票行情和相关的资讯信息，同时也有相当一部分投资者需要进行委托交易[3]。以上这些都会导致大量的请求数据需要交易委托系统承担并处理，这样开放式系统技术的应用就成为必然。

规模庞大的开放式分布结构在企业中很明确的表现出。企业中的计算机系统的数量和规模日益增长，系统间开放性的要求逐渐增加，组件之间联接互通已经成为基础设施[4]。在开放式分布式结构中，存在着各种类型的服务，资源和结点，而结点在地理上的分布就需要可互联互通互操作，这样实现实现资源和服务的共享[5]。因此各种系统之间互联互通，共享数据的需求日益增加。以此为背景下消息中间件技术与理论应运而生上并快速发展。

中间件的学术上的定义为：“中间件是一种软件产品，也是一种系统间节点组织通讯方式，在不同的系统结构、操作系统、通讯协议和数据库管理系统下，有效的使各个系统节点之间进行有效的通讯，从而实现相互的协同操作。”[6]

### 1.2.1 消息中间件的定义及工作模式

在异质的应用系统之间、网络通讯协议和操作系统消息中间件

（Message-Oriented Middleware, MOM）提供可恢复和可靠的的消息传送。这类中间件对消息的实时性传送，只需要保证消息的可靠到达目的地且只有一次即可。消息中间件可以在异步和同步方式两种方式实现。但相对于异步方式，同

步方式在强容错性相对较弱，所以异步方式可以保证正常传输数据，尤其在系统故障时。发布/订阅模式、消息队列模式和点对点模式是消息中间件有三种基本的工作方式[7]。

“由一个节点出发，将消息发送给一个或多个节点”是点对点模式工作方式。消息中间件在点对点模式下，其特点主要表现为：节点之间是面向连接的，在应用程序之间，首先需要建立专门的直接的通信服务；节点之间是位置不透明，通信双方在进行通信之前，需要提前知道对方的标识。这里所说的标识可能是通过系统名字服务——分布式开放性系统自身所提供的确定其位置的服务或通信对方的名称[8]。

发布/订阅模式，是多对多的，即多个节点可以订阅消息，同时多个节点可以发布消息。发布消息的节点仅仅要求以主题方式发送出去简单的消息，接下来所有订阅该主题的订阅消息节点都会接收到由消息代理来负责发送而来的消息。发布/订阅模式的特点：节点之间不需要建立通讯通道，通过使用消息代理，发送与接收双方进行相互之间的消息交换，而发送方与接收方关系的确定由接收方订阅了发送方某个主题而确立的。发布/订阅模式能够提供较好的容错性，因为消息代理可以实现信息的动态路由功能[9]；对应用程序完全透明，因为不需要与目的节点直接建立连接，由消息代理负责消息的传送，节点只需要简单将消息发送给消息代理；可以简化了应用系统的配置，可以实现多维空间上松耦合，发送节点不需要知道接收节点的网络地址和具体节点数量。

“发送方仅仅要做的是将消息发送到消息队列中，接收方通过访问消息队列的方式接收消息”是消息队列模式的工作方式。而消息队列模式的特点：应用系统之间不再需要直接建立专用通道，通过队列进行彼此间通信[10]。好处是应用系统不再需要保持完全同步，在通信过程中，不需要关心与子通信的对方是否存在或是否准备好处理这个消息，应用系统只需要将消息放在消息队列中

[7]；发送方与接收方无需直接建立通讯链路，通过消息队列即可交换消息；采用

异步方式，同时提供对信息的缓冲支持，具有安全可靠的优点，特别适用于不需要直接连接的应用系统[11]。

### 1.2.2 消息中间件优势

##### （1）有效降低开发系统时间

以屏蔽操作系统、网络访问和数据库访问的复杂性为目的，中间件可以起到良好的效果。从而程序开发人员只需要考虑在一个简单的开发环境中进行系

统的构建，有效的减少在系统设计的复杂性，从而大大减少技术上的复杂程度，程序开发人员只需要集中注意力在与系统面对场景相关的系统业务逻辑上[12]。

##### （2）降低系统研发成本

虽然额外研发中间件产品的成本较高，但是中间件作为一种可以广泛复用的软件产品。如果应用系统基于中间件平台开发的，那么从长远发展和总体目标实施考虑，中间件产品的研发是值得的[13]。

##### （3）减少研发应用系统失败次数

中间件可以做到屏蔽应用系统在底层面临的复杂运行环境，通过中间件的隔离作用，程序开发人员只需要面对一个相对简单的开发环境。由于问题简答化，设计与实现难度则有效的降低，所以研发应用系统的失败次数相对的会有所减少[14]。

##### （4）对已有的成果进行保护

中间件作为对底层复杂环境的屏蔽，可以对不同历史时期产生的不同通讯协议、操作系统和数据库管理系统进行复杂性隔离。这样企业在集成不同历史时期产生的应用系统时，难度大幅度降低，同时整体协调工作大幅度提高。公司可以不断的对历史应用系统进行集成，而充分利用过去产生的成果[15]。

##### （5）在应用系统集成时降低复杂性

随着企业的历史越悠久，规模越庞大，历史性的应用系统集成是不可逃避的问题。但是由于历史技术与理论应用与不同系统中，特别是基于不同接口标准和通讯协议后，使得集成各个应用系统变得几乎成为不可能完成的任务。而通过中间件就可以有效的降低系统集成的复杂性[16]。

##### （6）降低对历史系统维护费用

屏蔽不同通讯协议、操作系统与数据库管理系统，简化开发环境复杂性，降低不同系统集成难度。中间件带来的不仅仅是开发时间的缩减，同时也减少了系统维护、运营的工作量，从而减少了对系统整体的维护费用[17]。

### 1.2.3 国内外研究现状

中间件已经成为和数据库和操作系统并列的重要软件系统，现在已经存在了各种消息中间件产品，为企业提供了良好的数据集成服务。例如国外产品IBM的MqSerice和Microsoft的MSMQ，国内的产品如东方通科技的Tong LINK/Q、

FioranoMQ等，开源产品有OpenJMS、JBoss Messaging、ActiveMQ等。当然商业化的软件相对来说比开源软件会更加稳定和高效，可以获得更加专业的支持服务[18]。

#### 1.2.3.1 国内研究现状

金蝶iMTS是一种基于互联网架构的消息中间件。消息服务和传输服务是金蝶iMTS消息中间件的核心组成服务。传输服务实现发送和接收消息的功能，安全连接方面可选择的基于证书的方式，同时还提供身份认证、数字签名、数据加密等。消息服务支持定义消息队列的消息组织形式。消息服务负责删除、配置创建、发送队列和接收队列。金蝶iMTS消息中间件在安全性方面支持证书管理、地址管理及消息中间件状态监控等齐全的监控管理工具。金蝶iMTS消息中间件提供消息API和控制API，具有简单易用的特点。天生具有出色的跨平台能力，是由JAVA实现。针对性能作为优化，支持GZIP压缩传输，能够实现大数据包的快速传输。发送端进行数据的压缩发送，接收端会自动进行解压缩还原操作，这些在金蝶消息中间件不需要额外的手工工作。定义数据传输时间，充分利用空余的带宽时间，基于配置的实时传输模式等[19]。

国内中间件服务厂商中，技术领域的开创者和领导者是东方通科技。其旗下的TongLINK/Q是基于消息队列或消息传送的消息中间件。设计目的是在应用程序之间提供可靠的消息传送，可以不同的网络协议、操作系统和数据库管理系统之间进行。用户可以将一个完整的操作定义为一种事件。[20]TongLINK/Q通过名字服务和应用管理，来维持一个传输方之间的高效传输队列。同时，根据TongLINK/Q请求包的名称字段调用预先配置的对应的系统程序进行请求处理；同时具有一定的负载均衡，线程优化：根据每类服务进程所面对的请求数，自动扩充线程数量，这样可以保证请求队列高效的处理请求。TongLINK/Q的日志文件系统记录包括：日常运行系统信息，传输的数据包或文件的概要信息、系统错误提示等，日志可以定义共5个级别，越高的级别越详细的输出信息。这样通过日之后可以对系统服务进行调优，监控系统所处于的运行状态，对系统出现错误的追踪和调试排除。也就是说，日志在应用系统在开发阶段、测试阶段和运维阶段都可以提供强有力的支持。而这些日志的配置够可以有配置项进行改变，从而改变信息的输出详细程度，这样在开发式分布式系统中，有一定程度上的性能提升[21]。

中创软件推出消息中间件产品InfoBus/Q，有多种传播方式，包括：同步、异步、分组广播，对性能优化功能支持包括：分层传输、消息压缩、消息加密、消息路由和断电续传等，这些就可以解决了大数据并发传输时面临的性能瓶颈。

InfoBus/Q可以配置大容量的消息缓存队列，实现了分布式环境下的可靠消息传输[22]。InfoBus/Q与传统消息中间件不同的地方是，采用CORBA内核，从而具有开发性和异构性的特点。安全性上支持基于SSL的结点认证和应用程序认证策略。消息池的内存化，交互技术的高效算法采用，无限制的缓冲在硬件上的

支持等。由于缓存的实现，网络发送故障的时候，应用系统也不需要为此做出处理实现。由于开放性的监控实现，可以在任何一个节点上监控其他节点的运行情况，同时进行统一的管理、监控和维护[23]。

黎明网络iSwitch实现的是分布式的C/S架构的消息中间件。iSwitch重要组成部分包括：信息交换引擎、信息交换管理和业务集成网关。iSwitch针对网关支持安全隔离和物理隔离两种方式。iSwitch消息中间件支持异构数据库，并为其提供了基于图形化的配置管理工具。在稳定性方面，iSwitch信息引擎提供

99.999%信息交互成功率，对容灾和容错的技术提供了全方位的支持。在安全性方面，包括第三方认证证书，身份认证，数据传送时加密与解密等方法与手段和网关所支持的安全隔离[23]。

#### 1.2.3.2 国外研究现状

MSMQ全称MicroSoft Message Queue，微软消息队列，是以异步传输模式实现的在多个不同系统之间通信，通信的系统不要求必须在同一台机器，分布在网络的任何节点上都可以高效的进行通信。微软消息队列实现原理是基于容器的：发送方将消息放入容器中，这是消息存储在一个系统级别的公共空间接收方在可以在本地也可以在异地从该容器中查找并获得消息，从而进行处理。MSMQ队列类型下面几个种类：公共队列、共有队列、管理队列和响应队列[25]。

（1）公共队列可以被网络中所有的节点访问，并且可以复制消息到其他队列中去。

（2）共用队列规定只有本地的节点可以访问，而在网络上的其他节点不允许访[18]。

（3）管理队列消息队列网络发送消息后，会产生回执消息，管理队列中就会存储这些回执消息。

（4）响应队列即向目标系统发送消息后，接收其返回的回执消息[26]。

MSMQ优点在于它良好的稳定性，消息具有优先级定义，支持脱机方式进行传输。同时在安全性方面，反故障机制可以有效保证消息传递和业务相关操作执行处理的可靠性。同时，MSMQ的缺点是实时性无法保证，对于大量请求发生的场景，存在响应延迟。

IBM的Mqseries设计之初就不是实现简单的消息中间件产品。ESB是，对于一般意义上的一对一的消息中间件，将其放在星型拓扑的网络中而实现相应的系统架构。MqSeries就是IBM的ESB产品。MqSeries设计目的是起到系统总线作用，其成为整个系统的中心，所有节点都直接连接到消息中间件，避免系统之间直接相互联接。这种架构方式的优点是减少系统应用之间的耦合性。消息路由和数据转换是MqSeries的两个核心功能。Mqseries支持根据消息相应

字段和定义的业务逻辑来判消息的传播方向。即这就是Mqseries实现的路由功能。MqSeries支持的消息中间件的方式包括：直接连接消息中间件、指定关联文件或目录、web service通讯等[28]。

（1）直接连接消息中间件，MqSeries直接连接消息中间件。

（2）指定文件或目录，配置MqSeries管理某个文件或目录，被监视方有改动就认为有新的消息抵达，通过读取文件内容获得消息[19]。

（3）web service通讯，利用web service与之建立联系的方式进行通讯。

MqSerice在兼容性方面做到最大程度上的支持，特别是过去标准的系统或是少数特殊通讯方式的系统[29]。

#### 1.2.3.3 存在的问题

如上所述，由于近几年，有关消息中间件技术和理论得以发展快速，许多产生了多种多样的成熟的覆盖使用范围广的消息中间件产品，尤其消息中间件在异构系统整合方面中优势越来越突出。但是出于历史背景和设计目的原因，这些消息中间件产品还存在许多问题和不适用证券委托交易场景中使用的方面因素[30]。

在证券行业的背景下分析，现有的消息中间件产品存在以下几个方面的问

题：

（1）系统产品设计之初是面向大规模实施的，规模配置是重量级。因此在功

能设计上相对证券委托交易业务有许多无用的功能点。即在功能需求的实现上，缺乏有针对和适用证券委托交易业务的设计与实现；在证券行业快速发展的今天，需求变更经常发生，这就需要对实现的易于变更，而现有的无论是商业产品还是开源产品都具有通用化的是设计与实现。

（2）商业中间件产品价格昂贵，并且维护费用高昂，而开源中间件产品由于相关文档缺失，产品的稳定性相对不稳定，这样在证券行业中使用会承担相当大的风险。

（3）现有的软件产品开发设计比较早，很少有对分布式环境平台。股票交易在上午9: 30到10:30位开始连续竞价阶段，如果前一天股价被严重低估，今天会大量请求，然后过后就交易量骤降。这样对证券委托交易系统要求扩展性和延展性[31]。

## 1.3 本论文的主要工作内容

由于目前现存的消息中间件产品没有完全适用于开放式分布式环境下的消息中间件。因此本课题的主要研究目标是在证券业务背景下，针对证券委托交

易需求，对在证券委托业务场景下的消息中间件相关技术进行研究和实现：针对证券委托交易应用环境进行详尽的需求分析，分析研究现在已经存在的消息中间件具有的优点，选择与引进技术方案与设计思路，通过一系列详尽的可行的技术方案间进行比较与选择，从而满足需求的消息中间件的设计与实现，最终自主开发一个具有商业价值和可以实际通入商业生产的消息中间件，同时也灵活性高、专业性强、可进行对未来业务进行持续开发的面向证券交易数据的消息中间件产品。

结合课题的研究意义和目前存在的问题，本文的工作内容如下：

（1）分析对比现有的消息中间件产品，研究其的通信模型、工作模式等技术要点，通过借鉴优秀的技术思想与设计方案，并结合分析证券交易业务数据交换通信协议的消息格式，提出面向证券交易数据消息中间件的系统架构和总体设计。

（2）借鉴层次结构思想，有效划分消息中间件的模块结构。

（3）研究现有产品的在信息传输加密和文件日志系统建立上面的设计。详细介绍其具体设计、实现要点和创新内容，并将其分别应用与消息中间件的消息压缩子模块和消息中间件日志模块。

（4）基于混合编程技术，通过C++实现多线程模块和网络连接模块，而通过

LUA构建完整的框架，支持面向未来证券业务的二次开发。本论文的内容主要分为以下几个部分：

论文第1章简要介绍了课题的来源及研究意义，以及与课题相关的国内外研究状况。主要对消息中间件定义和工作模式，通过对比国内外代表性消息中间件产品，分析其中不足之处。

论文第2章论述了消息中间件的系统的需求，包括功能性需求和非功能性需求。同时论述与消息中间件实现相关的理论与技术。

论文第3章简要论述证券委托交易系统的架构设计。针对消息中间件的概要设计。主要包括分析与客户端通信协议和与柜台的消息请求格式。最后进行接口分析和模块划分。

论文第4章论述详细设计。包括各个模块间的协作关系和模块的类图的详细描述。

论文第5章论述消息中间件的实现与测试。介绍消息中间件的实现，功能验证以及性能测试和分析。

论文最后是结论，总结了论文的研究成果，证券委托交易系统中的消息中间件的设计与实现，并对消息中间件的后续改进工作进行论述。

# 第2章 消息中间件需求分析与总体设计

本章主要根据前期市场调研收集的信息，经过公司多年参与运行证券业务，参考多名专家的专业意见，对产品进行了功能性需求分析，并定义非功能性需求。然后简要介绍证券委托交易系统整体架构，对消息中间件所处的环境有整体概念。接着详细论述消息中间件的架构和内部所处理消息结构。最后论述系统功能模块划分。

## 2.1 消息中间件的需求分析

### 2.1.1 系统角色

邮政综合服务系统主要涉及的用户角色分为：

（1）客户端程序：投资者进行证券交易委托的客户端，即在各种操作系统上的PC客户端和手机客户端。客户端需要为投资者提供页面展示和相关的证券业务数据，同时在交易过程中，需要展示对话框、确认框和警示框等。

（2）交易柜台：券商运行维护的服务器，提供所有投资者相关的数据。消息中间件需要主动连接交易柜台，客户端程序委托消息中间件从交易柜台中获得业务数据。

### 2.1.2 消息中间件功能性需求

系统总体用例图如图2-1所示，消息中间件的主要功能具体描述如下：消息中间件主要包含三类业务功能需求：数据传输、业务数据展示以及柜

台状态监测。数据传输主要包含请求路由、数据压缩、请求参数分离、柜台请求接收、柜台请求发送、数据加密、柜台请求生成和应答数据生成，其中柜台连接建立包括了身份认证。业务数据展示功能目前主要包括页面载入、交换控制和菜单项控制。柜台监测主要包括柜台状态监测和柜台异常处理。其中消息中间件中内置定时触发器主要执行日子里文件记录、柜台状态监控和中间件状态监测。

下面将详细介绍部分关键的系统用例，详细的用例描述如下：

（1）菜单项控制：投资者在登录验证通过后，客户端程序需要显示菜单项、各个菜单项的顺序和菜单项之间的父子关系。消息中间件根据请求参数中的客户端版本号，投资者的身份信息返回菜单项的定义文件。

（2）页面载入：消息中间件根据请求参数中客户端版本号和投资者的身份信息返回预定义页面包。投资者在点击菜单项后，根据菜单项控制文件信息选择载入预定义页面或查询模板界面。

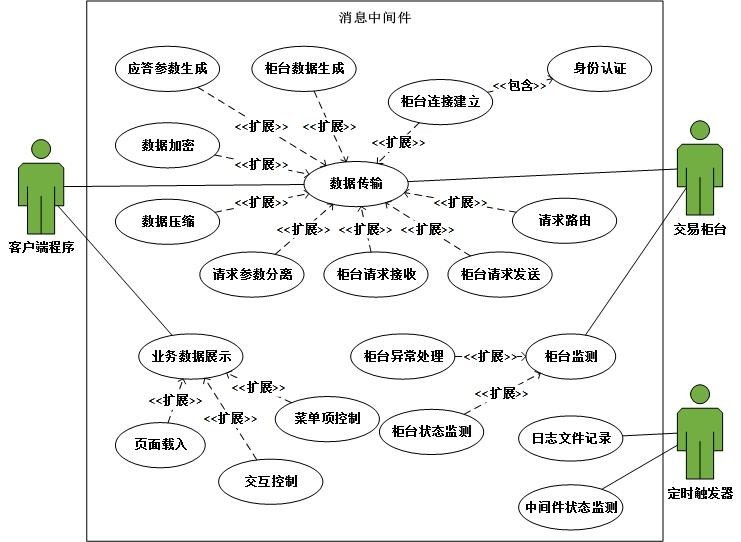


图2-1 系统整体用例图

（3）交互控制：这是消息中间件与客户端程序协定的通讯协议，根据应答客户端数据附带特定字段控制客户端额外显示提示框、警示框和确认框等。

（4）柜台状态监测：消息中间件主动的定时向交易柜台发送心跳请求，根据心跳包返回的应答信息确定柜台所处于的状态。交易柜台状态包括启动状态、服务状态、异常状态。同时如果一定数量的心跳包没有返回，认为网络状态已经不可用。

（5）柜台异常处理：在柜台状态处于启动状态和异常状态或网络状态已经不可用时，会进行发出异常消息，消息中间件内置的触发器会对异常状态进行异常处理。

（6）请求路由：分为静态路由和动态路由。静态路由是通过匹配配置文件中的每个路由信息中的可接受的请求名，决定转发的URL地址。动态路由是在柜台数据生成时，通过定义route\_no字段的值，来决定最终发向交易柜台的URL地址。

（7）柜台连接建立：消息中间件在启动时，通过配置文件对应柜台DLL，读取柜台定义字典文件，通过身份认证与交易柜台的可信的连接通信。

（8）身份认证：通过消息中间件中的身份证书，与交易柜台进行相互的身份认证。

（9）请求参数分离：对客户端程序发出的请求字符串，进行请求参数进行分离。其中请求字符串分为标准参数和扩展参数。

（10）柜台请求生成：根据请求参数分离的参数字段，组装出附带投资者交易委托数据的柜台请求。

（11）柜台请求发送：在柜台数据生成后，进行数据压缩与数据加密后，完成向交易柜台发送数据。

（12）柜台应答接收：接收柜台返回的数据，进行数据解压缩和数据解密，返回给应答参数生成，生成返回给客户端程序应答数据。

（13）数据加密：对柜台请求发送进行数据加密，同时对柜台应答接收进行数据解密。

（14）数据压缩：对柜台请求发送进行数据压缩，同时对柜台应答接收进行数据解压缩。

（15）应答参数生成：对柜台返回的应答数据，进行数据转换，组装成客户端程序的需要的响应字段。

（16）中间件状态监测：不断显示消息中间所处的状态，需要提供一个对外展示的GUI界面，同时需要提供一个配置端口，为后续的监测程序开发提供条件。不断显示中间件所处的状态，有利于运维人员进行不断监控整个证券委托交易系统处于正常状态。保证委托交易正常、稳定、可靠的运行。同时也有利于开发人员进行调优。

（17）日志文件记录：在实际的生成环境中，需要不断记录各种运行情况到文件中，方便出现问题可以进行分析查找问题。日志应该分层次生成，对于请求日志、异常日志和连接日志应该放在不同的文件中。这样有利于查找问题的方便性。

### 2.1.3 消息中间件非功能需求

消息中间件的非功能需求主要体现在响应时间、鲁棒性、可复用性、安全性和稳定性五个个方面，阐述系统的性能目标。

##### （1）响应时间在实际的证券交易，以股票为代表，集中在几个时间段内

进行，尤其是行情转变情况下，交易量巨大导致请求数激增。这就要求消息中

间件可以在规定的时间给出响应。依据尽快失败的思想，如果当前的系统处于不稳定的状态，也要在规定时间给出异常响应。

##### （2）健壮性在异常和危险情况下系统生存的关键。考虑到消息中间件自

身或者证券委托交易系统处于不稳定状态时，仍然可以保证部分基础功能正常运行，尤其是日志文件模块可以有效记录出现的异常情况和消息中间件当时正在处理的消息请求。

##### （3）可复用性任何一款软件在开发的设计和实现过程中都应当注重利用

模块化设计来降低耦合度和复杂性。模块化设计还有助于减少开发工作量，提高可复用性，从而降低软件开发成本。所以系统应对一些具有较高重用价值的模块进行封装，比如通用性较强、使用率较高的基础业务逻辑模块、基础通信功能模块和多线程模型等。

##### （4）安全性在证券委托交易过程中，数据安全是第一位的。要保证数据

的传输在加密的状态，不能再传输过程中被篡改。通信的双方是可信的，不能中途被伪造接收方。

##### （5）稳定性稳定性需求是指系统拥有长期稳定运行的能力，保证消息中

间件不因为长时间运行导致的崩溃，能够持续稳定的运行下去。通过长时间的测试模拟用户操作，尽可能保证消息中间件在实际运行时不会出现问题。

具体需要达到的硬性性能指标为。性能需求主要在于使用本程序时，每秒可以处理的请求数和数据量，参考CPU占用率，所以必须要满足：

（1）在系统运行过程中，每秒可以处理300个用户请求。

（2）在系统运行过程中，每秒可以处理30k左右的数据。

## 2.2 消息中间件的总体设计

### 2.2.1 证券委托交易系统拓扑结构

证券委托交易系统总体拓扑结构模型如下图2-2所示。投资通过客户端程序进入证券委托交易系统中，比如图中PC端、PAD和手机。PAD和手机通过互联网接入手机主站，手机的数据经过手机主站进行数据格式转换，通过网关进行与委托主站进行连接。PC客户端程序通过互联网直接与委托主站进行连接。委托主站把收集到的请求转发给消息中间件，消息中间件对请求数据进行数据格式转换发送给交易柜台，交易柜台经过自身的业务逻辑处理，返回应答消息。消息中间件再次进行数据格式转换发回给委托主站，由委托主站原路返回给消息发出方。

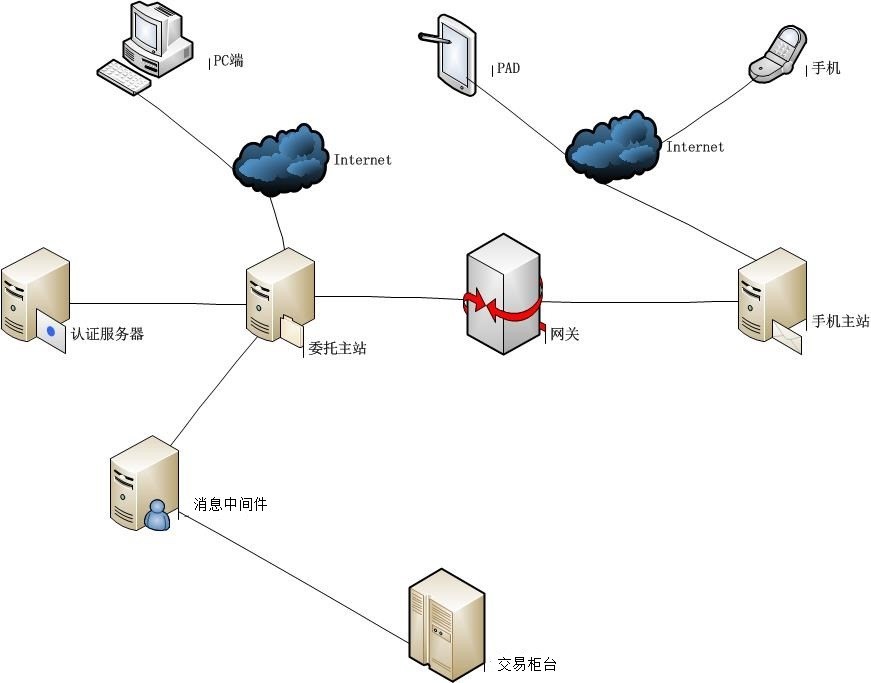


图2-2 系统总体拓扑结构图

### 2.2.2 消息中间件总体架构

消息中间件采用C++与LUA混合编程技术。以C++实现底层基本的网络通信、多线程实现和数据缓存；以LUA对处理逻辑进行实现。如图2-3所示消息中间件架构图。

整个消息中间件大体分为两层，LUA脚本框架层和C++框架层。LUA脚本框架层为实现消息中间件主要功能模块逻辑部分。对请求进行抽象可认为获得客户端数据表示控制、业务数据以及对柜台状态的判定都是请求。不仅仅定义请求为获取业务数据的请求。这样基于键值对映射的关系，定义一系列命令字符串，并对应其处理逻辑的方式构建整个LUA框架。柜台监测模块是设计之初预定义命令字符串，由消息中间件单位时间触发调用。相对的，其他两个模块的逻辑需要接到相应的命令字符串才会触发相应的逻辑。C++框架层是对LUA层的支撑作用，最重要的部分是线程池和数据存储区。数据存储区一方面存储认证请求时保存用户基本信息，另一方面缓存一定的查询类请求数据。线程池部分是C++框架层运行的根本。在线程池中建立三类线程，分别为（1）预处理线程，处理主站发过来的请求数据（2）工作线程，处理预处理线程处理的数据，具

体是调用LUA脚本框架层的逻辑进行处理（3）发送线程，将工作线程处理完的数据返回给主站。通过一定的配置项设置各类线程的数目，可以进行调优处理，保证三类线程都能进行一定程度上的满负荷处理状态。其中工作线程是整个消息中间件的集中逻辑实现部分，其他部分都是为其服务的，所以这部分也是设计的重点部分



图2-3 消息中间件架构图

为建立高可靠性和高安全性的消息中间件，需要在原有的需求分析基础上建立以下几个服务程序：

监听服务：设立一个单独的运行线程，不断监听中间件发出的事件，对各种事件进行匹配处理适配器，进行逻辑处理。

目录服务：目录服务允许客户端在整个分布式环境中寻找需要的服务。具体的实现包括两个独立的组成部分，柜台服务和中间件服务，柜台服务是对柜台的功能的进一步封装，而中间件服务是中间件自身提供的服务。两个服务提供了层次型的命名空间，并且归纳与一个目录服务下，方便在分布式系统中的各个组件调用。

线程服务：利用线程服务，客户端和柜台之间的消息交换就可以有多个控制流，从而实现内部的并行处理。

安全服务：认证，数据加密和校验组成。认证是在与柜台建立连接时，需要出示认证密码或认证证书，接收返回的系统号sysNo。在进行数据发送时，对数据逐字符流数据加密。校验是在加密之前对需要传输的字符流进行，使用CRC

（循环冗余校验）。

### 2.2.3 消息结构定义

消息中间件处理消息的模型即消息结构，当接收到客户端程序发送的消息请求命令字符串，首先分离出命令字符串，匹配对应配置表，配置表即有消息结构组成模块组成，每个组成模块根据所起的作用实现相应的逻辑，LUA框架层就可以统一按照相应的顺序进行处理消息结构组成模块，从而完成消息中间件所承担的各个需求功能。如表2-1所示消息结构组成模块表。

消息结构有三部分组成：请求头部分、预处理部分和交易柜台返回数据处理部分。请求头部分包括nor\_head和ex\_head。来源是客户端发出的消息请求字符串。其中nor\_head的组成是固定字段，这些固定字段是会被经常使用到的。而ex\_head是扩展请求头部分，是客户端程序针对当前消息请求字符串而外添加的字段。预处理部分是pre\_handle，是对请求头部分的字段生成柜台字段部分。这个过程可以通过定义函数的方式实现复杂的功能。最后一部分是柜台返回数据处理部分。交易柜台返回数据由单行返回和多行返回两种模式，其中单行返回由post\_handle和extend\_handle进行处理。多行返回统一放在table\_handle下，组成有table\_handle. head、table\_handle. handle、table\_handle. OTHERHANDLE 和

table\_handle. SKIPFUNC。最后userinfo\_handle是对消息结构的扩展，可以在处理交易柜台返回数据部分时，进行数据保存，以方便其他的LUA配置表进行使用。

每个处理模块都对应一个对应表，对应表有键值对组成，左边是发向交易柜台或客户端程序的字段，取决于键值对所处于消息结构模块组成部分。右侧可以直接进行赋值，也可以定义个对应表。对应表组成有def，为默认值；field，取得交易柜台返回字段或请求头字段；change，可以定义函数，用以field获得

值后调用函数进行逻辑处理。对于右侧是对应表的情况，处理的过程为获取field值，如果没有，取def值。对取的值执行change函数，如果没有，则不处理，最后return返回。

表2-1 消息结构组成模块表

|  |  |
| --- | --- |
| 组成模块名称 | 说明 |
| nor\_head | 标准请求头，组成字段都是固定 |
| ex\_head | 扩展请求头，由客户端程序放入 |
| pre\_handle | 处理请求配置表，组织柜台需要的业务数据 |
| post\_handle | 处理单行返回配置表（多行返回默认处理第一行） |
| extend\_handle | 处理单行的扩展请求返回数据配置表 |
| userinfo\_handle | 处理单行返回操作用户缓存数据配置表 |
| extern\_data\_handle | 处理标准请求使用扩展返回配置表 |
| error\_handle | 处理请求错误函数（非配置表） |
| table\_handle | 处理多行数据总配置表 |
| table\_handle.head | 多行数据返回数据结果集（这里可以当作是定义表头） |
| table\_handle.handle | 多行数据返回处理配置表 |
| table\_handle.handle.OTHERHANDLE | 多行数据单行汇总处理函数表 |
| table.handle.handle.SKIPFUNC | 多行数据单行过滤标准处理函数表 |

对于table\_handle为多行返回配置表，其中head表头部分，可以定义表头的数目、表头的长度、显示文字、类型、对齐方式等属性。默认的表头定义为{字段名，字段显示名，数据类型，风格，数据长度，小数点位数}。对于字符串、整数类型数据，小数点位置无效。除了字符串类型，其他类型的数据长度只是显示的长度，不影响存储的长度。

OTHERHANDLE是table\_handle. handle的多行辅助处理表。它负责对整行数据进行处理，作用和post\_handle等单行返回的数据处理很类似、要做到对多行数据的修改，就只能通过OTHERHANDLE。在进入到OTHERHANDLE处理函数中，即表示，从柜台返回的多行数据的一行，已经被转化为消息中间件中

的数据结果，并且可以对最终返回作出修改了。比如需要对其中一列的显示做数学操作，结果为第一列的数据加上第二列的数据求平均值。

SKIPFUNC是table\_handle. handle的多行辅助处理表。它负责过滤数据。在进入到SKIPFUNC处理函数中，即表示，从柜台返回的多行数据的一行，已经被转化为消息中间件中的数据结果，并且可以对最终返回作出修改了（和

OTHERHANDLE一样，可以对数据最终做出修改）。

### 2.2.4 功能模块划分

根据第2章需求分析的用例图划分功能模块，如上图2-4所示，本系统共划分为三大功能模块，业务数据展示模块、数据传输模块以及柜台监测模块。每个模块同时又包含许多小的模块，下面对各个模块进行描述。

页面与菜单载入模块：当客户端进行身份认证成功后，需要由登录窗口跳转展示主页面。这时客户端程序需要从消息中间件中读取菜单项控制文件和预定义页面包。该模块即根据客户端程序发出的客户端版本和柜台返回的投资者认证信息，根据LUA配置表定义的逻辑，返回对应的菜单项控制文件和预定义页面包。最终完成客户端程序的菜单项载入和各个页面的正确显示。

交互模式模块：是与客户端程序之间建立的协议，旨在根据业务需要，消息中间件返回提示信息和控制字段，客户端根据这些提示信息和控制字段进行展示UI元素，以实现符合业务场景的用户交互逻辑。客户端需要实现的的UI元素有警示框、对话框、提示框和协议展示框。同时支持在提示框下进行页面跳转和直接访问URL地址或发出命令字符串。



图2-4 系统模块划分图

柜台连接建立模块：在消息中间件启动时，需要与柜台建立可靠且相互信赖的数据链路，这就需要此模块具有身份认证的功能。同时为了尽可能发挥消息中间件的处理性能，需要根据配置文件决定具体建立多个与柜台连接的数据链路。

请求消息预处理模块：对收到的客户端程序发送的消息请求字符串，需要对字符串进行分解处理。首先分离出各个命令字段，生成命令字符从，标准请求头和扩展请求头把。把以上数据放入数据存储区中，以方便其他的模块进行取出使用。

请求消息处理模块：是消息中间件的主要模块，对所有收到的请求消息进行处理。主要包括以下过程：

（1）柜台请求收发，完成对柜台请求的发送和柜台应答的接受。它依赖于数据压缩与解压缩模块和数据加密与解密模块。只有经过以上两个模块，才能还原数据信息，交由配置表进行业务逻辑处理。

（2）柜台请求生成，由于交易柜台种类不止一个，所以不能有客户端程序直接发送交易柜台请求，这样客户端程序只给出生成交易柜台请求必要的参数，而由柜台请求生成模块生成交易柜台请求。

（3）请求路由模块：消息中间件与交易柜台之间的路由分为两类，动态路由和静态路由。静态路由是在配置文件中，直接定义交易柜台IP与端口号和可以处理的请求命令，这样在接到请求命令后匹配对应的柜台地址。动态路由是在配置表中定义route\_no字段，可以LUA框架层的与C++框架层接口部分指定交易柜台地址选择时，进行动态的根据route\_no进行选择。

（4）应答参数生成模块：在接到交易柜台应答数据后，对应答数据进行数据格式转换，成为客户端程序可以解读的字段名称。即客户端程序发出请求命令后，生成应答数据。

（5）数据压缩与解压缩模块：旨在降低消息中间件与各个组件之间的网络传输数据量，从而提高响应速度，对所有发送到交易柜台传输的消息进行压缩，同时对所有从交易柜台接受到的数据进行解压缩。

（6）数据加密与解密模块：旨在提高消息中间件的数据传输安全性，对发向交易柜台的数据，根据私钥进行加密，对收到的数据根据公钥进行解密。对加密和解密操作进行模块封装，方便其他模块调用。

柜台状态监测模块：由于消息中间件主动与消息中间件进行建立连接，这就需要不断监测交易柜台当前所处于的状态，是否可以正常处理客户端程序产

生的业务相关请求。如果交易柜台处于异常状态，发出事件，由柜台异常处理模块进行处理。

柜台异常处理模块：对接收到的柜台异常事件，通过调整消息中间件自身加以应对。

日志文件记录模块：以保证消息中间件可靠性为目的。日志分为请求消息日志，程序异常日志和连接状态异常日志。日志的分类记录可以保证后续的查找问题的方便性。

中间件状态监测模块：不断收集中间件中各个线程所处于的繁忙状态，同时也收集消息中间价已经处理的请求数和数据量，形成统计信息，为单独形成统一的监测管理提供可能。

## 2.3 本章小结

在本章中根据前期的客户调研和多年参与证券业务的相关经验得出了该系统的用户业务需求。并对消息中间件的业务需求做出了详细的分析，为了方便后续的设计开发工作，更好地细化和明确这些需求，本章采用UML面向对象建模语言，使用用例图的图例驱动分析的方式，进一步明确了消息中间件的参与者和用例，消息中间件与参与者之间的互动关系。最后根据客户的要求和系统实际的生成环境，分析了用户对系统的非功能性需求，主要包括稳定性需求和性能需求。接着介绍了证券委托交易系统的总体架构。然后叙述了消息中间件的总体架构和消息结构。最后根据需求分析进行模块划分。

# 第3章 消息中间件的详细设计与实现

第2章对消息中间件进行了需求分析和总体设计，通过上述内容将消息中间件分为三部分功能模块，业务数据展示模块、数据传输模块和柜台监测模块。其中由定时器和触发器调用的两个功能模块：日志文件记录模块和中间件状态监测模块，日志文件记录模块可以提高消息中间件的问题定位能力，从而进一步提高稳定性；中间件状态监测模块，不断收集中间件各个线程所处的状态，并提供对外接口以进行展示，从而通过优化参数进一步提高性能。本章将详细讲述这个四部分的详细设计。

## 3.1 业务数据展示模块设计与实现

业务数据展示主要控制客户端程序展示证券业务数据的样式和与投资者进行交互的方式的定义。业务数据展示主要包括页面与菜单项载入模块、交互模式模块两个功能模块。页面与菜单项载入模块通过请求参数分离模块分离出客户端程序的请求命名字符串中命令参数，根据客户端程序版本号和客户个人信息进行返回相应的控制文件。交互模式模块主要通过与客户端程序相互约定的协议，在返回给客户端程序的消息中，通过定义相应的字段，来控制客户端进行实现相应交互所需要的对话框样式。

### 3.1.1 页面与菜单项载入模块设计与实现

页面与菜单项载入模块主要包含以下2个功能点，页面载入和菜单项载入。客户端程序在登录界面跳转到主界面前，需要完全载入预定义页面包和菜单项控制文件，这样才能正确生成主界面，页面与菜单项模块划分图如下图3-1所示。



图3-1 页面与菜单项模块划分图

菜单项载入功能流程图如下图3-2所示。开始时，客户端程序处于显示登录界面，投资者在输入资金账号和密码后，选择一种加密方式对资金账号和密码进行加密，客户端发送投资者身份验证请求命令字符串。消息中间件在接到

身份验证请求字符串后，由请求参数分离模块进行请求命令字符串分离参数，如果身份验证失败，返回错误消息并结束。如果身份验证成功，通过在消息结构中的extern\_data\_handle组成模块定义xd\_func和xd\_func\_add字段，通知客户端程序完整菜单项控制文件和附加菜单项控制文件所处于消息中间件的相对路径，这样客户端程序发送这个相对路径给消息中间件，获得完整的对应的菜单控制项配置文件。



图3-2 菜单控制项载入功能流程图

一般来说，消息中间件中会定义多个func. xml和func\_add. xml文件，分别为完整菜单控制项文件和附加菜单项控制文件。完整菜单控制项文件定义了客户端程序必须展示的菜单项。附加菜单项控制文件是对完整菜单项控制文件的补充。完整菜单项控制文件有且只有一个，而附加的可以有多个。如下图3-3菜单项控制目文件结构图。

<?xml version="1.0" encoding="GB2312"?>

<XD\_Function>

<class name="股票" id="129">

<item name="查询" id="0" icon="184" icon2="184">

<item name="查询资金股票" id="518" id\_pos="165" replace="165"/>

<item name="查询资金股票" id="165" delete="1"/>

<item name="查询模板 1" id="xxx" icon="167" id\_pos="169" name\_pos="当日成交" type="cx\_template" reqtype="2304" extra\_cmd="cx\_template\_demo1"/>

</item>

</class>

<class name="开放基金" id="240">

<item name="申购" id="227" icon="161" class\_startpage="1"/>

</class>

</XD\_Function>

图3-3 菜单项控制文件结构图

每个菜单有以下基本属性：

name：菜单显示名称。

id：菜单id。

icon：菜单图标编号

icon2：菜单打开时的图标，一般子菜单只要指定一个就可以，而父菜单可以指定两个。

id\_pos：表示定位节点的菜单id，如果不为0，那么新节点插入到这个节点

后面

name\_pos：表示定位节点的菜单名称，必须完全匹配，如果不为空，那么

新节点插入到这个节点后面

pos：前面id\_pos和name\_pos都找不到时，默认添加为父节点的最后一个节点，如果该值为“first”，那么添加父节点的第一个子节点。

replace：如果设置了replace对应id，那么当对应id的菜单项最为起始页面时，会使用当前页面替换。

startpage：强制客户端程序以指定的页面作为起始页面。

class\_startpage：当标签切换时候，将作为默认菜单。

在实现时，如果在添加前能找到对应的菜单项，那么不再添加。如果id=0

会自动用name查找。

以上定义的菜单项都在预定义页面包中存在对应的页面。同时消息中间件支持定义查询模板。查询模板只能显示二维表格的页面，同时支持简单的时间范围定义和翻页功能。一下对查询模板中的字段进行说明：

type：菜单类型，有“ajax”和“cx\_template”。如果为空，默认是ajax方式，如果是查询模板则必须返回。

reqtype：请求类型，有扩展请求和已定以的查询请求。

extra\_cmd：扩展请求命令，查询模板页面实现的数据直接来自请求命令对应的配置表。

show\_date：是否显示日期控件。1表示显示，默认不显示。

qsrq：开始日期，如果show\_date=1则会出现。如qsrq=-30，表示当前日期减去30天为初始显示日期，如果值大于10000000，则显示指定日期。

zzrq：追加日期，表示当天加上数字的日期。

auto\_refresh：是否自动刷新，值为1时，表示切换到当前页面就会触发一个刷新动作。

html：当type=ajax时，可以再这个参数指定一个URL地址，或一个html

文件，这样客户端会取相应的页面并显示在客户端程序上。

extern\_data\_handle =

{

['EX'] = '2.0',

['hxui\_support'] = '1',

['xd\_func'] replydata)

=

{def = '', change = function(v, ans, userinfo, askdata,

local clientver = versionNoStandardization(askdata:getdata('clientver')); for key, value in pairsByKeys(versionToFuncFile) do

if clientver <= tonumber(string.sub(key,2)) then

return string.gsub(base64encode\_ex(ReadFile(value) or ''), '=',

'%%61');

end

end;

return all\_menu; end},

['xd\_func\_add'] = add\_menu,

},

这样就可以再LUA脚本层中的身份验证配置表中定义返回的菜单项控制配置文件。如图3-4菜单项控制核心代码片段图。

图3-4 菜单项控制核心代码片段图

“EX”和“hxui\_support”规定客户端程序是否需要支持页面载入。

Xd\_func\_add直接返回add\_menu变量，其中定义了func\_add的相对路径并且经过base64encode函数进行编码。xd\_func定义处理函数，首先获得请求客户端的版本号，然后依次遍历字典versionToFuncFile，其定义了各个版本范围内的客户端应该载入的菜单项控制配置文件的相对路径。如果没有找到，就返回默认定义的add\_menu变量定义的菜单项控制配置文件。

页面载入控制模块功能流程图如图3-5所示。



图3-5 页面载入控制模块功能流程图

载入完成菜单项控制文件后，客户端程序发送请求命令“ajax\_pkg”，消息中间件在捕获到请求命令后，匹配对应的配置表，调用LUA业务层脚本进行处理。首先获得请求头中param字段、客户端程序支持的核新版本号hexin\_ver和类型type。对参数param进行解析，获得需要读取的文件名称列表和对应md5值。逐个验证文件名对应的文件和解析出md5值是否相同。如果存在不同，说明是请求消息有误，结束过程。把文件名和预定义页面包名都放在pkg\_table表变量中。如果遍历结束，并且type类型为“hxui”，根据hexin\_ver变量值逐步递减的方式遍历pkg目录下的hxuiX子目录，X是hexin\_ver的版本号，如果存在停止遍历，生成相对路径地址，并对预定义页面包进行base64encode编码，并返回。这里在pkg目下始终存在hexin1/ggt. pkg预定义页面包，保证最终会返回一个预定义页面包。

Ajax\_pkg对应的配置表是自返回模式。预定义页面包在消息中间件本地，不需要到柜台去获取。如下图3-6自返回模式核心代码片段图。

pre\_handle =

{

['\_USE\_SELF\_DATA\_'] = '1', ['\_SELF\_DATA\_'] =

{

{'index'},

{'1'},

},

},

图3-6 自返回模式核心代码片段图

### 3.1.2 交互模式模块设计

交互模式是客户端程序与消息中间件之间约定的协议，旨在通过消息中间件返回提示信息和定义显示方式字段，客户端程序实现相应的UI元素，从而实现完整的用户交互行为的定义。需要指出的是需要客户端配置开启交互模式，否则仍然不会实现用户交互行为的定义。

详细的控制字段描述见下表3-1所示。交互模式可以实现模态和非模态两种对话框类型。同时支持简单的弹出窗口和大型的文本协议窗口。文本协议窗口用以显示在证券开户等场景中需要用户签署确认的协议。在对话框上可以添加是否两个按钮成为选择框，可以只添加确认按钮成为确认框。在大型的文本协议窗口中，可以添加复选框，并把复选框和是按钮和否按钮进行关联，形成在投资者在确认协议前，必须点击复选框，表示自己已经阅读了相关协议内容，然后才表示同意开户等业务相关操作的。

表3-1 交互模式控制字段描述表

| 编号 | 控制字段 | 描述 |
| --- | --- | --- |
| 1 | show\_modal | 为 1 时显示模态对话框 |
| 2 | show\_yesno | 为 1 显示“是/否”按钮，0 显示“确认”按钮， 对于 big\_msg 情况可返回“2”显示“同意/不同意” |
| 3 | action\_yes | 点“是”操作，如 next\_req,menu\_jum,pop\_url，  try\_again 等，默认 try\_again，即重发当前请求 |
| 4 | action\_no | 点“否”操作，用法通 action\_yes |
| 5 | next\_type | 下一个请求类型，是标准请求或扩展请求，默认2403，表示扩展请求 |
| 6 | next\_cmd | 下一个请求命令字符串 |
| 7 | menu\_jump | 将要跳转到菜单 id |
| 8 | pop\_url | 需要打开 URL 链接，这个值需要  base64+urlencode 编码 |
| 9 | big\_msg | 使用长 retmsg 模式，客户端程序使用用户协议对话框显示 |
| 10 | retmsg | 需要返回给客户端程序提示信息，如果是  big\_msg=1,需要 base64+urlencode 编码 |
| 11 | yes\_text | 确认按钮显示文本 |
| 12 | no\_text | 取消按钮显示文本 |
| 13 | check\_text | 复选框按钮显示文本 |
| 14 | show\_check | 是否显示复选框 |
| 15 | check\_relation | 复选框关联控件，关联时 uncheck 状态，控件为  disable，如果 1 表示关联 yes, 2 表示关联 no，  0 表示无关联 |
| 16 | check\_action | 为 no\_showagain, check 状态提交时，可以控制本次登录不再显示该对话框 |
| 17 | caption | 对话框标题设置 |

需要指出的是，action\_yes和action\_no支持设置多个操作，中间用“；”分隔。当客户端程序与接口中间需要多次交互的时候，接口必须返回next\_flag记录，然后通过next\_flag字段设置接下来需要flag字段的值，也可以通过flag字段直接设置值。

## 3.2 数据传输模块设计

数据传输模块目前主要功能，包括：柜台连接建立模块、请求消息预处理模块、请求消息处理模块、应答消息返回模块、日志文件记录模块和中间件状态监测模块。数据传输模块是消息中间件基本模块，其他模块都依赖这个模块的部分子模块。

各个模块之间的关系如图3-7所示。



图3-7 数据传输子模块间关系图

首先由柜台连接建立模块建立与柜台之间的可信赖网络通道，同时完成消息中间件的初始化。然后判断消息中间件是否处于运行状态，如果是一直循环下面处理过程。如果接受到客户端程序发送的消息请求命令，由请求预处理模块进行消息请求命令的解析，解析工作完成后由请求处理模块进行处理，处理完成后应答消息返回模块生成返回给客户端程序的应答消息，最后日志文件记录整个处理过程所产生的日志信息。如果没有接受到客户端程序的消息请求命令，则监测消息中间件是否在启动中。即如果消息中间件一直在运行状态，消息中间件就会不断检查是否收到了消息请求命令，而在运行状态中，中间件状态监测模块一直会收集消息中间件的运行信息，展示到消息中间件的相关界面上。

### 3.2.1 数据传输模块设计

数据传输模块作为消息中间件的基础模块，对其设计覆盖了消息中间的所有类，其他模块是在其基础上，在LUA脚本中定义了逻辑实现。如图3-8



图3-8 数据传输模块涉及主要类图

CTransMidApp作为界面启动按钮的响应类，有OnStart方法进行整个消息中间件启动。CTransMidDoc作为整个消息中间件的抽象，统一前端界面显示和后端逻辑处理的协调调用。CTransMidView是前面界面显示的响应类，主要各种统计数据、日志信息和各类线程所处状态在用户界面的展示。CLogHandle是日志工具类，CTransMidView 对于日志文件的各种操作都依赖于这个类的协作。

CTMHandle是整个消息中间件后端逻辑的控制中心，所有的调用关系定义、数据整合都有这个类负责。CMulitProperty是配置文件的操作工具类，可以快速获得配置项对应的配置值。\_ThreadInfo是结构体，定义了线程的结构。

CGuiTaiInterface是LUA框架与C++框架的接口，C++通过这个类与LUA框架

进行数据交换。CGCommData是C++框架为LUA框架提供的支持工具类，使得

LUA框架可以快速的获得所有公共数据信息，包括C++动态产生的数据项。下面分三个部分叙述数据传输模块：以CTransMidView为中心的前端界面

数据支持，以CTMHandle为中心的后端逻辑支持和以CGuiTaiInterface为中心的LUA框架支持。

##### （1）以CTransMidView为中心前端界面数据支持

如下图3-9前端界面数据支持类图所示。CTransMidView作为用户界面显示的响应类。最主要的方法：Dump方法是从缓存区中获得数据的方法，传入的参数是消息中间件启动后获得缓存区的引用；OutPutLog方法是不断向请求明细界面输入请求日志，参数可以控制显示的级别和样式；OutPutLog2方法是向文件中追加写入请求日志，参数控制写入的详细程度。



图3-9 前端界面数据支持类图

CTTable是用户交互界面中四个标签界面的父类，只定义公共方法GetData，用以交互组件调用以获得需要显示的数据。其子类分别为：CTheadTable是线程状态标签页获得数据的响应类，CRequestTable是请求明细标签页获得数据的响应类，CLinkTable是连接信息标签页的响应类。CMonitorTab是运行日志标签页的响应类。

##### （2）以CTMHandle为中心的后端逻辑支持如下图3-10 后端逻辑支持类图所示。



图3-10 后端逻辑支持类图

CMultiProperty提供一个表结构的属性对，应以对后端实现时属性获得改变的支持。重要的方法：AtomChangeProperty提供原子性操作修改属性值，这种修改只会影响消息中间件内存中维护的属性表结构值，对于由文件定义的属性值并不会有影响；GetProperty获得属性值，可以获得内存中维护的属性表中的某项属性值。

CTMHandle是后端逻辑控制类，同时也是运行状态统计类。它分为两类方法：一类是统计信息方法，另一类是启动各类线程和队列方法。

1.统计信息方法

GetAllWaitWorkRequest方法获得所有等待工作线程进行处理的请求数目。

GetBusyRate方法获得繁忙率，是以处于繁忙状态的工作线程除以总的工作线程数。

GetWaitRequest方法获得由委托主站发送的，等待预处理线程进行处理的请求数目。

2.启动各类线程和队列方法

CheckThread方法用以检查各类线程是否正常启动。这是一个公共方法，实际上在内部调用一系列私有方法以完成检查。

CreateWorkThread用创建检查工作线程。

CreateInterface用以启动LUA框架，并返回CGuiTaiInterface指针。

CreateWorkList用以创建工作队列。

CreateSessionNotifyRequest用以创建一个事件监听器，用以捕捉消息中间件在运行过程中产生的各种事件。

DoPreHandle用以预处理线程的完整创立过程。

DoWorkHandle用以工作线程的完整创立过程。

DoSendHandle用以发送线程的完整创立过程。

GetIniFile方法用以完成消息中间件启动时的所有字典文件导入数据缓存区

中。

OnReceiveData，当发生柜台返回数据事件时，监听器就会触发调用此方法

进行处理。

OnSessionNotify，默认的事件监听处理函数，对不没有定义的事件类型进行处理。

OnTime，消息中间件定义的定时器，会定时触发在LUA框架中定义的

OnTime函数中定义的处理逻辑。

PushReq2WorkList，将请求推入到工作队列中去。

PushToSendList，将请求推入到发送队列中去。

SendReply，依赖柜台实现类，进行消息的发送函数。

SetReplyErrorMsg，从柜台返回的消息中提取出应答错误信息。

StartWorkThread，用以启动工作线程，包括LUA框架脚本导入工作线程并启动。

WriteLog，对请求消息写入日志文件中，可以定义写入的类型和具体的格

式。

GetThreadList可以获得所有队列指针所组成的链表。

CThreadList是队列类，请求队列、预处理队列和发送队列都是其实现的。

其中只有一个公共方法CheckTimeOut用以检查队列中是否有请求已经超时，如果超时从队列中去掉，并返回异常信息。

CUserMap是对预处理队过程分离出的用户信息进管理，主要包括删除用户信息方法DeleteUserInfo和获得用户信息方法GetUserInfo。

\_ThreadInfo是对线程信息的进一步封装，方便CTMHandle进行统计，也方便用户交互界面进行数据展示。

##### （3）CGuiTaiInterface为中心的LUA框架支持如下图3-11 LUA框架支持类图所示。



图3-11 LUA框架支持类图

CGuiTaiInterface类是C++与LUA框架连接类。C++底层接受到的数据通过它与LUA框架进行数据交换，同时可以提供一些高性能方法支持LUA框架调用。

HandleRequest方法是LUA获得C++提供底层支持与C++与LUA进行数据交换的入口方法。

Init\_config方法是设置LUA字典并建立在内存中的字典映射。

Global\_init方法完成全局初始化工作，然后通过global\_clone方法快速建立克隆对象，加快多个工作线程建立过程。

IsOk方法可以为工作线程建立完成后判断LUA框架是否完全载入正常提供判断。

PushSendData方法是C++向LUA框架中推送数据的方法。

RegisterFunc可以提供C++实现的函数给LUA框架直接使用，这样就可以避免LUA中实现的方法有性能的要求而无法实现的问题。

RunMySelfRequest方法为自返回请求提供处理方法。因为自返回请求并不需要与柜台进行交互，所以在C++底层实现时细节与普通处理并不相同。

### 3.2.2 柜台连接建立模块实现

柜台连接建立模块首先要完成消息中间件的初始化工作，然后通过认证文件与交易柜台进行身份认证，从而与交易柜台建立可相互信赖的网络通道，为接下来的数据交换打下基础。即柜台连接建立模块有两个之功能，如图3-12柜台连接建立模块划分图。



图3-12 柜台连接建立模块功能划分图

CtransMidDoc中方法OnStart是启动消息中间件。随之的处理逻辑由图3-13

所示。



图3-13 消息中间件启动流程图

用户在消息中间件点击启动按钮后，首先通过类CmultiProperty读取字典文件和配置文件，然后在数据存储区中建立字典文件和配置文件的内存实体，即

Map结构实体。然后检查当前消息中间件是否处于运行状态，如果是即结束启动逻辑。如果没有实例化CTMHandle，调用其OnStart方法开始消息中间件启动逻辑，并且设置消息中间件状态为运行中状态。

首先初始化预处理队列，通过调用CTMHandle类中DoPreHandlePro方法，用以存放接收到的客户端程序请求消息命令，然后初始化预处理工作线程，具体启动的预处理工作线程数目通过配置字典实体CMultiProperty类的

GetProperty方法中获得。调用CTMHandle类的DoWorkHandlePro初始化工作

队列，存储预处理线程处理完成的请求实体的指针。建立柜台连接，具体建立的柜台连接个数由配置文件实体中获得。CTMHandle中成员变量b\_plink指针指向一个CTMLink类型的指针，通过CMultiProperty类实例GetProperty方法获得柜台类型，从而实例化CTMLink类，并赋给b\_plink。初始化工作线程，启动的工作线程数据由配置文件实体获得，如果有工作线程没有启动成功，调用方法setRunningWithThread设置消息中间件为非运行状态并随即结束。如果全部工作线程启动成功，初始化发送线程，用以把应答发送给客户端程序；调用

CTMHandle类DoSendHandlePro方法初始化发送队列，存储工作线程处理完的请求后生成的应答信息，等待发送线程进行发送。

如下图3-14消息中间件多线程工作模型。



图3-14 消息中间件多线程工作模型

三种队列：预处理队列、工作队列和发送队列是为缓存上一个线程的处理结果，下一个线程获取数据位置。在CTMHandle类中分别定义成员变量

m\_nThreadHandles和m\_nThreadList存储队列和线程池的链表开始指针。消息中间件多线程工作模式为，委托主站的请求被接收后，放入请求队列进行缓存排队，当有空闲的预处理线程时会进行预处理操作，预处理完成后放入预处理队列中。空闲的工作线程就会从预处理队列中获得请求，按照工作线程的LUA脚本进行处理，处理完成后放入发送了队列中，等待发送。发送线程将会逐个发送给委托主站。

其中柜台连接建立功能的流程如图3-15所示。

首先从配置文件实体中获得需要建立的柜台连接数，然后不断循环建立。先建立一个与柜台通信的监听端口，然后彼此通信出示身份认证文件，如果认证失败，则直接关闭消息中间件。如果认证成功，则把这个连接线程放入线程池中，循环变量i加1，直到所有的柜台连接数建立完成。



图3-15 柜台连接建立流程图

### 3.2.3 请求消息预处理模块实现

请求消息预处理模块，完成对接到的请求消息命令字符串，分解出各个字段，并生成标准请求头和扩展请求头。如图3-16请求消息预处理模型流程图所示。



图3-16 请求消息预处理模块流程图

首先获得预处理队列的指针，方便操作预处理队列。判断消息中间在运行状态，如果否直接结束。如果是从预处理队列中获得请求包，对请求包进行分解操作，从而生成标准请求头和扩展请求头。从数据存储区中根据资金账号获得对应的用户信息缓存，如果标准请求头中的信息与数据存储区中的数据不匹配，则更新数据存储区中的用户信息。然后更新线程统计信息，把生成的标准请求头和扩展请求头写入请求日志中。

### 3.2.4 请求消息处理模块实现

请求消息处理模块是整个消息中间件的核心逻辑，主要是依据预处理工作线程生成的标准请求头和扩展请求头，生成柜台请求，然后通过加密和压缩柜台请求进行发送到交易柜台中，收到交易柜台的返回应答后，在解密和解压缩，生成应答信息，放入发送队列中。

如图3-17工作线程运行逻辑流程图所示。



图3-17 工作线程运行逻辑流程图

首先获得工作队列指针，方便后面对工作队列进行操作。检查消息中间件是否运行，如果没有运行，直接结束。如果运行则试着从工作队列中获得请求头的指针，如果没有获得，则调用OnTime方法，进行柜台状态监测模块的触发。

如果获得指针，则通过指针到数据存储区中获得标准请求头和扩展请求头，生成柜台请求。检查配置表是否为自返回模式，如果是则直接生成客户端应答消息；如果不是对柜台命令进行压缩和加密，从线程池中获得一个发送线程，发向交易柜台。当接收到交易柜台的应答消息后，对应答消息进行解密和解压缩，然后在生成客户端程序应答，把客户端应答消息放入发送队列中，然后更新线程统计信息和写入请求日志的，重新获得指针。

### 3.2.5 应答消息返回模块实现

应答消息返回模块，是对已经产生的客户端应答消息发送个对应的客户端程序。如图3-18应答消息返回模块流程图所示。



图3-18 应答消息返回模块流程图

首先获得发送队列指针，方便操作发送队列。然后检查消息中间件是否处于运行状态，如果不是则直接结束。如果是则从发送队列中获得应答请求，对应答请求设置删除标志，防止应答请求发回委托主站后，委托主站又重新发回消息中间件，导致无法进行处理。然后发送到委托主站，同时更新线程统计信息和写入请求日志，重新获得应答请求。

### 3.2.6 日志文件记录模块实现

日志文件记录模块是对消息中间件出现日志分类进行记录的。对于由于程序设计或实现导致异常错误，则把异常错误信息写入异常错误日志中。对于建立柜台连接和断开柜台连接时，需要记录到连接日志中，终点是实现。对于客户端程序和交易柜台之间的请求，需要写入到请求日志中，同时可以显示在消息中间的对应界面上。

### 3.2.7 中间件状态监测模块实现

在上面的请求消息预处理模块、请求消息处理模块和应答消息返回模块设计中，都有更新线程统计信息，也就是各个线程不断的更新一份统计信息，这样消息中间件就可以再界面上进行展示了。

## 3.3 柜台监测模块设计

由于消息中间件需要主动连接交易柜台，所以消息中间件自身需要保证与交易柜台的连接可靠。柜台监测模块就是消息中间件中定时触发的监测交易柜台单位时间内说出的状态，一旦确认交易柜台当前处于异常状态，这时就需要进行异常处理，通过调整消息中间件自身，来保证整个证券委托交易系统的消息传输可靠性。柜台监测模块包括：柜台状态监测模块和柜台异常处理模块两个部分。

### 3.3.1 柜台状态监测模块设计

柜台状态监测模块，是消息中间件不断确认与交易柜台之间建立的通信通道处于连接状态，并且通过询问交易柜台所出的状态来确定交易柜台是否处于异常状态。

如下图所示3-19，柜台状态监测模块流程图。

定时器触发心跳包，调用LUA框架中OnTime函数。OnTime中默认5秒钟没有请求发柜台后，调用心跳包配置表生成并发出柜台状态监测心跳包。心

跳包是一个预定义的柜台请求，生成心跳包后发向柜台。默认心跳包3秒没有返回后，就认为网络已经不可达了，随即发出“网络阻塞”异常信息。如果柜台成功返回了应答信息，如果异常号码字段存在，则生成对应的异常信息抛出。如果没有异常信息，则正常结束。



图3-19 柜台状态监测流程图

### 3.3.2 柜台异常处理模块设计

如果捕获到异常消息，就会触发异常处理模块。这里设计之初预先定义了异常名称，和对应的默认的异常处理行为。具体如表3-2异常名称与预定义处理方式。

表3-2 异常名称与预定义处理方式

| 异常名称 | 异常说明 | 预定义处理方式 |
| --- | --- | --- |
| 网络阻塞 | 网络阻塞或不可达 | 消息中间件自动重启 |
| 交易柜台重启 | 交易柜台正在重启 | 关闭自身，30分钟后重启 |
| 交易柜台繁忙 | 交易柜台处于繁忙状态 | 默认不处理 |
| 交易柜台临时中断 | 交易柜台由于数据核查等原因，暂时停止处理 | 默认不处理 |
| 交易柜台功能失效 | 交易柜台部分交易功能失效，无法正确处理消息请求 | 关闭自身 |

消息中间件的异常针对的交易柜台的情况进行处理。当消息中间件和交易柜台的网络链路不可达或者网络处于繁忙时，消息中间件的心跳包不能到达交易柜台，这时候消息中间件自身会进行重新启动。当消息中间件重新启动时，如果与交易柜台无法建立连接后，消息中间件会自动关闭的。如果交易柜台处于重启状态，一般交易柜台重启需要20分钟左右。所以消息中间件首先关闭自

己，阻止接收任何客户端程序的请求，然后延后30分钟后进行重启。如果柜台处于繁忙状态，默认实现时不会进行处理的，有的券商也会提出让消息中间件主动丢失一些委托请求。如果交易柜台临时中断，一般情况下是因为柜台核查系统发现有异常错误数据，决定暂时停止处理新的委托交易命令，这种情况也是默认不会处理的。如果交易柜台反馈为柜台功能失效，那么说明交易柜台现在已经不能有效处理请求消息了，那么消息中间件自身关闭，停止接收任何客户端程序发送的请求消息。

## 3.4 本章小结

本章根据第2章的需求分析为基础，根据第3章的总体设计，按照模块划分进行逐一的设计与实现。重点是数据传输模块的设计，首先是数据传输子模块之间的关系，然后分别对柜台连接建立模块、请求消息预处理模块、请求消息处理模块、应答消息返回模块、日志文件记录模块和中间件状态监测模块的设计与实现进行详细的叙述。

# 第4章 消息中间件的测试

消息中间件作为证券委托交易系统的重要组成部分，它的运行准确性和可靠性对证券委托交易系统起着很大的影响作用，因此消息中间件在上线商用之前需要进行严格的测试。本章将根据需求分析中提出的功能需求和非功能性需求做出相应的测试，并且对测试后的结果进行分析与评价。

## 4.1 功能测试

[功能测试](http://baike.baidu.com/view/651435.htm)为对消息中间件的各项功能进行验证，通过下述的测试用例逐项检测系统功能，检查本系统是否满足用户的需求。

### 4.1.1 测试环境

功能测试在系统正式上线前进行，使用公司内部网络进行测试，不开放外网访问的IP及端口。同时对于证券委托交易系统依赖的柜台环境，公司选择恒生柜台为主，主要考虑到恒生柜台在券商生产中使用广泛，测试环境比较好获得。其中其他组件服务器中运行的是客户端程序和委托主站程序。环境如下表4-1所示：

表4-1 消息中间件环境信息表

| 设备名称 | 硬件要求 | 数量 |
| --- | --- | --- |
| 交易柜台 | Dell R220 1 CPU 3.2GHZ 8G 内存 2T 硬盘 | 4 |
| 消息中间件服务器 | Dell R920 2 CPU 3.2GHZ 4G 内存 600GB 硬盘 | 1 |
| 其他组件服务器 | Dell R920 2 CPU 3.2GHZ 4G 内存 600GB 硬盘 | 1 |

### 4.1.2 测试用例

系统将根据不同的测试用例对每项功能进行测试，系统功能测试主要分为由开发部门负责的系统测试与由测试团队负责的集成测试两部分。系统测试会从正常场景和异常场景两种环境进行测试，力求覆盖系统每个模块的功能。确保本系统各个模块的功能覆盖率，最终满足用户的业务需求。

本小节列出部分核心功能的测试用例及测试结果，证明消息中间件已经满足用户的基本功能需求，达到上线商用的要求。页面与菜单项载入模块功能测试用例如下表4-2所示：

|  |  |
| --- | --- |
| 用例编号 | THSB2B-014 |
| 测试内容 | 完成客户端登陆后主页面正常载入测试 |
| 测试功能 | 页面与菜单项载入功能 |
| 测试步骤 | 1. 客户端程序输入资金账号和密码 2. 选择一种对传输资金账号和密码的加密方式 3. 点击登陆按钮 4. 客户端程序正常显示主页面 5. 更改消息中间件中 func.xml 中的菜单项名称 6. 重新进行登陆，查看主页面对应菜单项名称 7. 逐个点击菜单项，观察每个页面是否显示正常 |
| 预期结果 | 对应菜单项名称已经更改，每个页面显示正常 |
| 实际结果 | 客户端可以正常各个页面，菜单项名称正常受到控制 |
| 测试结果 | 符合测试预期，测试通过 |

表4-2 页面与菜单项载入测试用例

通过上述测试用例证明页面与菜单项载入功能满足用户需求，能够成功获取预定义页面包，并且客户端的菜单项接收消息中间件的控制，可以有效的调整客户端的菜单项展示。

交互模式用例如下表4-3所示：

表4-3 交互模式测试用例

| 用例编号 | THSB2B-016 |
| --- | --- |
| 测试内容 | 完成验证交互模式有效性 |
| 测试功能 | 交互模式功能 |
| 测试步骤 | 1. 客户端程序点击“测试交互模式”菜单项  2. 点击“简单警示框”按钮  3. 点击“简单确认框”按钮  4. 点击“简单协议框”按钮  5. 点击“携带跳转警示框”按钮  6. 点击“携带复选框协议框”按钮 |
| 预期结果 | 消息中间件在实现了各个交互模式对应的请求命令的配置表，并在配置表中使用协议规定的控制字段，控制客户端显示相应的交互 UI，完成可能发生的与用户在场景中发生的交互 |
| 实际结果 | 简单弹框和协议框都能正常显示各个 UI 组件，并且跳转功能可以实现 |
| 测试结果 | 符合测试预期，测试通过 |

通过上述测试用例证明交互模式功能满足用户需求，能够成功应对由于需求变更，对于用户进行的交互方面的细节调整加以应对，完成基本交互场景中的提示、警示、确认和相关协议展示与签署的作用。

迁址查询功能测试用例如下表4-4所示：

|  |  |
| --- | --- |
| 用例编号 | THSB2B-017 |
| 测试内容 | 完成数据压缩与解压缩测试 |
| 测试功能 | 数据压缩与解压缩功能 |
| 测试步骤 | 1. 配置 XDTest 工具发送测试命令。  2. 检测接收到的数据量大小  3. 估算测试命令发送的数据量  4. 显示接收到的数据 |
| 预期结果 | 接收到的数据量应该小于估算的数据量，并且在多次调整测试命令发送  的数据量时，大小关系仍然不变。并且接收到数据正确无误 |
| 实际结果 | 测试命令中使用自返回模式，在 retmsg 中写入 30k 左右的数据，使用  big\_msg=1 模式，流量监测到数据量大小为 4k 左右。接收到数据正确无误 |
| 测试结果 | 符合测试预期，测试通过 |

表4-4 数据压缩与解压缩测试用例

通过上述测试用例证明数据传输模块满足用户需求，能够成功压缩数据。接收到数据正确无误，表明可以进行解压缩操作，还原数据内容。

数据加密与解密测试用例如下表4-5所示：

|  |  |
| --- | --- |
| 用例编号 | THSB2B-018 |
| 测试内容 | 完成线路采集测试 |
| 测试功能 | 1. 网点查询中网点地图查询功能 |
| 测试步骤 | 1. 配置 XDTest 工具发送测试命令。  2. 检测接收到的数据，并直接展示  3. 显示接收到的数据 |
| 预期结果 | 监测工具提取的数据是乱码，而通过 XDTest 获得返回数据可以正常显示 |
| 实际结果 | 在测试命令中使用只返回模式，ret\_msg 中写测试提示内容。监测工具提取的数据是乱码，而通过 XDTest 获得返回数据可以正常显示 |
| 测试结果 | 符合测试预期，测试通过 |

表4-5 数据加密与解密测试用例

通过上述测试用例证明数据加密和解密功能能够满足用户需求，能够成功保证数据传输的安全性，有效防止恶意修改证券委托交易中产生的数据的威胁性。

数据传输模块试用例如下表4-6所示：

|  |  |
| --- | --- |
| 用例编号 | THSB2B-019 |
| 测试内容 | 数据传输测试 |
| 测试功能 | 1. 请求参数分离模块功能  2. 柜台请求收发模块功能  3. 柜台连接建立模块功能  4. 柜台应答参数生成模块功能 |
| 测试步骤 | 1. 点击“数据传输测试”选项。  2. 查看返回数据。 |
| 预期结果 | 测试数据模拟用户委托一笔股票买入操作，返回委托编号数据 |
| 实际结果 | 模拟用户买入 002767 先锋电子，100 股， 返回委托编号 23 |
| 测试结果 | 符合测试预期，测试通过 |

表4-6 数据传输模块测试用例

通过上述测试用例证明数据传输功能满足用户需求，能够把客户端提交的请求命令字符串进行参数分离，然后组装成柜台可以接受的柜台命令，最后接受交易柜台返回的数据，最后应答参数生成模块进行生成应答数据。接受到柜台确认委托生成的委托编号23。

固话缴费功能测试用例如下表4-7所示：

|  |  |
| --- | --- |
| 用例编号 | YZZHFW-040 |
| 测试内容 | 请求路由模块测试 |
| 测试功能 | 请求路由模块测试 |
| 测试步骤 | 1. 配置消息中间件，建立两个柜台地址，分别接受不同的请求参数  2. 分别点击“测试柜台地址 1”按钮和“测试柜台地址 2”按钮  3. 查看返回内容 |
| 预期结果 | 分别发送 T-5 请求和 L-rzrq-zhrz 请求，配置两个柜台地址分别接受请求， 柜台返回的 sysNO 不应该相同。 |
| 实际结果 | T-5 返回 sysNO 为 3，L-rzrq-zhrz 返回 sysNO 为 4 |
| 测试结果 | 符合测试预期，测试通过 |

表4-7 请求路由模块测试用例

通过上述测试用例证明请求路由模块功能满足用户需求，能够成功请求命令参数进行路由发送，T-5应该发送给普通交易柜台，L-rzrq-zhrz应该发送给融资融券柜台。

日志文件记录模块功能测试用例如下表4-8所示：

|  |  |
| --- | --- |
| 用例编号 | YZZHFW-047 |
| 测试内容 | 完成日志记录测试 |
| 测试功能 | 1. 日志文件记录模块 |
| 测试步骤 | 1. 开启 debug 模式。  2. 清空所有日志。  3. 启动消息中间件，查看连接状态日志  4. 清空已经产生请求日志  5. 客户端程序“日志记录测试”按钮  6. 查看请求日志内容 |
| 预期结果 | 消息中间件启动后，会在连接状态日志记录连接时间和连接状态，客户端发送的请求，在开启 debug 模式下，会记录在请求日志中 |
| 实际结果 | 连接状态日志被新建，记录 Start /propid=3644;prownd=3212272;  [E662011.11.17] 等信息，请求日志记录了完整的 T-5 请求 |
| 测试结果 | 符合测试预期，测试通过 |

表4-8 日志文件记录测试用例

通过上述测试用例证明日志文件记录功能满足用户需求，能够成功记录消息中间件的连接状态和完整的请求过程。这个功能在正式的生产环境中是关闭的，只有在调试时打开，请求日志也是该5天后会自动删除的。

如图4-1消息中间件运行日志界面。



图4-1 消息中间件运行日志界面

运行日志界面中会展示消息中间件最基本的信息。预处理线程、工作线程和发送线程是否正常建立，柜台连接是否正常建立。在运行期间，如果出现错误信息，在这个界面上会以红字展示出请求字符串和错误信息的内容。在状态栏上，会显示监听是否正常，已经经过的用户数量，等待请求对应显示的是请求队列、预处理队列和发送队列在等待请求的数量，总共经过的请求数量和每秒已经处理了多少请求。

如图4-2消息中间件请求明细界面。这个界面在非debug模式下会简要的显示时间，处理的请求命令名称，命令字符串和登录账户等信息。而在debug模式下，会完整显示每个请求的发向柜台的请求命令，柜台返回的应答数据和核新处理后的需要返回给客户端的数据信息。如果在脚本中有调试命令，也会显示一些调试信息。当出现错误时，这个界面同样会出现红字字体的错误信息显示。



图4-2 消息中间件请求明细界面

如图4-3消息中间件线程状态界面。



图4-3 消息中间件线程状态界面

显示预处理线程、工作线程和发送线程的运行状态。展示线程是否繁忙，已经处理的请求数，运行状态，总的处理时间和平均每个请求处理时间，处理的数据量和一些提示信息，表示这个线程当前所处的状态。

## 4.2 非功能测试

非功能测试是对消息中间件进行并发性与响应速度的性能和稳定性的测试。系统性能测试旨在检测消息中间件对于高并发、高请求量的系统负载与系统响应速度进行测试，查找系统性能瓶颈，分析可能导致性能瓶颈的原因，提出优化方案并比对方案实施效果。

消息中间件是证券委托交易系统的数据通道，维护客户端程序和交易柜台之间的数据交换；系统间的交互是由具体业务、行情构成的，如果依靠自然请求对消息中间件进行性能测试无法满足压力测试的需求，因此采用浙江核新同花顺网络信息股份有限公司开发专门测试工具XDTest。如图4-4所示XDTest界面。



图4-4 XDTest测试工具界面图

XDTest是基于客户端模型，即本身就是模拟客户端发送请求和接收数据，只不过它并不显示数据的返回结果，只是确认是否数据返回。所以在大量连接的情况下，单个程序的执行效率可能会较低，需要启动多个测试客户端同时进行。

XDTest有两种模式，登录测试模式和压力测试模式。登录测试模式是使用动态口令和通讯密码的发出身份认证请求，测试时每次发送一个请求之后重新连接，重新认证。这样就能测试消息中间件的稳定性。压力测试模式是身份验证成功后，重复发送已经设定好的请求命令字符串。需要注意的是在这种模式下，前一个请求回来之后再发送下一个请求，避免请求积压造成XDTest测试程序崩溃。这种方式能测试出后台的最大处理能力，但是不能通过配置的方式控制请求速度，所以同样需要启动多个XDTest 测试程序，并且把委托主站和

XDTest放在不同的机器上，因为XDTest本身占用的资源也是比较大的。

### 4.2.1 稳定性测试

利用XDTest工具的登录测试模式，使用编写测试脚本方式对系统的稳定性进行测试，通过长时间的运行观察超时请求数和失败请求数。一般股票委托交易时间为上午9: 30至11:30，下午13: 00至15:00，总共进行4个小时的交易。

我们制定24小时测试时常，完全可以保证稳定性的可靠。

测试结果如下表5-10所示：

|  |  |
| --- | --- |
| 测试项目 | 测试结果 |
| 测试模式 | 登录模式 |
| 测试总时长 | 24 小时 |
| 启动 XDTest 数量 | 1 |
| 发送身份认证请求总数量 | 25920264 |
| 成功请求数量 | 25920112 |
| 超时请求数量 | 54 |
| 失败请求数量 | 98 |
| 成功数占比 | 99.9994% |

表4-9 稳定性测试结果

对消息中间件运行24个小时，共发送25920264个请求，超时请求54个，失败请求98个，成功数占比超过99.999%。上述测试结果证明消息中间件在时间运行下，仍能保证请求消息的成功处理。消息中间件的稳定性满足用户需求，在证券委托交易的过程中可以保证绝大部分的请求成功处理。

### 4.2.2 性能测试

性能测试主要为在使用消息中间件的过程中，监视程序的CPU占用率及内存占用大小，以每秒可以处理的用户请求数和数据量。测试时启动的柜台连接线程数为130个，预处理线程数为20个，不同的是工作线程的启动数量不同。测试的请求命令字符串覆盖股票交易的买入、卖出、查成交、查委托、查资金流操作。测试的请求覆盖了证券委托交易的常见柜台命令类型，同时也是客户在进行委托交易过程中经常会触发的柜台命令。可以即提供了需要生成柜台请求的操作，也提供了需要大量接收柜台返回数据的操作。尽量模拟在实际生产环境下消息中间件处理的客户端委托交易请求命令。经过测试环境中进行系统测试，具体测试结果如下表5-10所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 最大值 CPU 占用率 | | 每秒处理请求数 | 每秒处理数据量 |
| 单工作线程 | 7% | 80 | 5K |
| 双工作线程 | 24% | 180 | 13K |
| 三个工作线程 | 75% | 300 | 30K |
| 四个工作线程 | 100% | 352 | 40K |

表4-10 性能测试结果

通过上述测试结果证明消息中间件的性能能够达到用户的基本需求。在启动三个工作线程的基础上，最大CPU占用率在75%左右，每秒可以处理300个请求数。同时可以发现，制约消息中间件性能是CPU的处理能力，而不是内存。这就需要接下来的进行进一步的降低CPU的占用率。

## 4.3 本章小结

本章主要对系统进行了详细的测试，其中主要包括功能测试和非功能测试两部分。在功能测试中，列出了主要功能点的测试用例及测试结果，以证明系统满足用户的需求。在非功能测试中，主要测试了系统的稳定性、性能。经测试证明，系统已满足需求分析中的各种功能及非功能性需求，满足券商业务生产的要求。

结论

论文首先对消息中间件的现状与发展进行了阐述与研究，并针对于消息中间件的需求进行分析，设计整体消息中间件的总体架构，采用C++进行底层基础部分开发，使用LUA脚本进行框架构建，以支持实现大部分模块和功能点。紧接着论文对于消息中间件主要功能模块的设计与实现进行了详细的描述。最后通过测试证明了本系统满足需求中提出的内容，达到券商在业务生产的实际要求。

本论文的主要工作包含以下几个方面：

（1）本论文通过分析得出结论：开放式分布式已成为当今证券委托交易系统的必要架构模式，所以消息中间件作为开放式分布式系统中的地位越来越重要。分析了国内外比较成熟的消息中间件产品，对于其各个消息中间件的优点和不足都做了论述。

（2）随即进行详细的需求分析，叙述了功能性需求和非功能性需求。

（3）在总体设计中，首先对证券委托交易系统拓扑结构进行分析，然后进行消息中间件总体架构，定义消息结构和模块划分。

（4）根据总体设计中的模块划分，逐一对模块进行设计与实现说明，重点是数据传输模块，包括其自模块之间的关系。

（5）最后分别对消息中间件进行功能测试与非功能测试，以验证其达到需求标准。

本文所设计实现的消息中间件已经在浙江核新同花顺网络信息股份有限公司运行，目前为多家券商提供证券委托交易的消息中间价按，已经取得良好的效果，随着2014年沪港通的业务上线，已经证明本文所设计实现的消息中间件可以满足证券委托交易的实际需求。

下一步的工作内容主要为对原有上线的证券业务配置表进行维护，对证券业创新的业务模式进行支持，不断优化消息中间件的设计与实现，进一步降低

CPU的占用率并提高每秒处理消息数的能力。

参考文献

[1] 王宏斌. 消息中间件的研究和实践[D]. 杭州: 浙江大学, 2004.15-26

[2] 王波. 基于Java消息服务的消息中间件的研究与设计[D]. 成都: 电子科技大学, 2007 12-25.

[3] 吴一男. 分布式交易平台中消息中间件的设计与实现[D]. 杭州: 浙江大学, 2008 28-49.

[4] BIRRELL A D, NELSON B J. Implementing remote procedure calls[J]. ACM Transfer Computer System, 2004, 2(1): 39-59.

[5] BANAVAR G, CHANDRA T D, STROM R E, et al. A case for Message-Oriente Middleware[C]. Proceedings of the 13th International Symosium on Distributed Computing. London: Springer-Verlag, 2005: 1-18.

[6] EUGENE A. Heterogeneous Middleware Structures for Mygrocer Project[C]. Proceedings of the M-BUSINESS International Conference on Mobile Computing, Athens, Greece, 8-9 July 2002 15-25.

[7] 潘慧芳, 周兴社, 杨刚. 基于混合通信模式的消息中间件设计与实现[J]. 计算机工程, 2006, 32(3): 116-118.

[8] HORSTMANN C S, CORNELL G Core Java 2, Volume 2[M]. 5th ed. Saddle River, New Jersey: Prentice Hall, 2006 24-56.

[9] CARDELLI L, WEGNER P. On Understanding Types, Data Abstraction, and Polymorphism[J]. ACM Computing(CSUR), 2013, 17(4): 471-523.

[10] David S. 数据压缩原理与应用[M]. 吴乐南译. 北京: 电子工业出版社, 200335-65.

[11] P. Maheshwari, Hua Tang, Roger Liang. Enhancing Web Serice with Message-Oriented Middleware[C]. Proceedings of the IEEE international Conference on Web Service. 2004 36-39.

[12] Margo Seltzer, Michael Olson. LIBTP: Porable, Modular Transaction for UNIX[C]. Proceeding of the Winter 1992 USENIX Conference. 2003 25-56.

[13] Michael Stonebraker. The design of the POSTGRES storage system[C]. Proceeding of the 13th Conference on Very Large Databases. 2001 26-46.

[14] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, R. Panigrahy. Consistent Hashing and Random Trees: Distributed Caching Protocols for Reliving Ho t Spots on the World Wide Web[C]. 29th Anuual ACM Symposium on the theory of Computing. 2000 36-46.

[15] 杨文涛, 刘卫忠, 郑立新, 等. 多阶上下文自适应二进制算术编码实现[J]. 华中科技大学学报: 自然科学版, 2007, 35(3): 42-45

[16] Xia Meifeng, Shi Hongbao. Secure Model and Design of Network Communication System Based on Data Encrypton[J]. Comuter Engineering, 2001 35-67.

[17] W. Richard Steven. Unix Network Programming Volum 1 Networking APIs: Sockets and XT[M], Second Edition. 北京: 清华大学出版社, 1999 13-15.

[18] N Vun, Y H Ooi. Implementation of an Android Phone Based Video Streamer[J]. IEEE, 2010: 912-915

[19] 尚海忠, 朱堵彦, 等. 操作系统抽象层: 一种支持跨平台的新技术[J]. 计算机工程, 2002, 28(2): 27-28.

[20] SteveSouders. High-PerformanceWebSites[J]. Communicationsofthe ACM. 2008, 51(12): 36-41.

[21] Eugster P T, Pascal A Felber, Rachid Guerraoui, et al. The many faces of publish/subscribe[J]. ACM Computing Surveys, 2003, 3, 5(2): 114-131.

[22] 中科院软件所软件工程技术研究开发中心, 消息队列中间件ISMQ的设计和实现[Z]. 2000 36-48.

[23] X Zhang, et al, A PerformanceStudy of Monitoring and Information Services for Distributured Systems[C]. Proceedings of HPCD, 2003-12.

[24] BERNSTEIN P A. Middleware: a model for distributed system services[J]. Communications of the ACM, 2000, 39(2): 86-96.

[25] RAO B R. Making the most of middleware[J]. Data Communications, Computer Standareds & Interfaces, 2007, 3, 30(3): 1-9,

[26] T. H. Harrison, The Design and Performance of a Real-time CORBA Event Service, in Proceedings of OOPSLA'97, ACM, October 2013

[27] Frederic Rossi, Asynchronous Events on Liunx, Open System Lab System Research report, Ericsson Research Canada, 2013 34-56.

[28] 徐金悟等译, 基于C++CORBA 高级编程[M]. 北京: 清华大学出版社, 200235-79.

[29] 吴乐南. 数据压缩[M].2版. 北京: 电子工业出版社, 2014 24-26

[30] 杨志明. 几种常见软件体系结构模型的分析. 计算机工程与设计, 2014, 25(8): 1326-1328.

[31] 曾冠东. 基于MINA构建简单高性能的NIO应用[J]. 程序员, 2013(2): 120-123.

[32] 潘国伟, 宋玮, 消息中间件在电力调度自动化系统中的应用[J]. 电气应用. 2014(10) 14-36.

[33] 于江. SmartMessageQ消息中间件设计与实现[D]. 成都: 成都理工大学, 2005 14-124.

[34] 周荣林. 消息中间件的研究与应用[D]. 北京: 中国地质大学, 2007 25-47.

[35] 庞瑾, 试论金融证券交易系统设计与应用[J], 计算机光盘软件与应用, 2012(19) 34-36

[36] 张晓京, 饶一梅, 证券交易与信托投资系统的实现方法[J], 系统仿真学报2010(4) 23-25

[37] 候经川, 吕想, 上海证券交易系统的建设进展与对策研究——基于上海建设国际金融中心的战略视角[J], 中国软科学, 2012(5) 23-25

[38] 陈德栋, 消息中间件可靠传输机制的实现[D], 电子科技大学, 2012 23-35

[39] 李易民, JMS 安全消息中间件的设计与实现[D], 北京工业大学, 2011 12-23

[40] 刘克熊, 基于JMS的消息中间件模型的研究与实现[D], 西安电子科技大学, 2011 14-23

[41] 宋雨, 王莉莉, 基于 JMS 消息服务技术的研究及应用[C], 第一届中国高校通信类院系学术研讨会论文集, 2007 23-13

[42] FREEMAN E, BATES B, SIERRA K. HEAD First Design Pattens[M]. 南京: 东南大学出版社, 2006.

[43] BernsteinP. Middleware: A ModelforDistributedSystemServices. Commnuications of the ACM, 2006, 39(2): 86-98.

[44] Verissimo P, Rodrigues L. Distributed Systems for System Architects. Kluwer Academic Press, 2001 24-35

# 哈尔滨工业大学学位论文原创性声明和使用权限

学位论文原创性声明

本人郑重声明：此处所提交的学位论文《证券委托交易系统之消息中间件设计与实现》，是本人在导师指导下，在哈尔滨工业大学攻读学位期间独立进行研究工作所取得的成果，且学位论文中除已标注引用文献的部分外不包含他人完成或已发表的研究成果。对本学位论文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明。

作者签名：日期：年月日

学位论文使用权限

学位论文是研究生在哈尔滨工业大学攻读学位期间完成的成果，知识产权归属哈尔滨工业大学。学位论文的使用权限如下：

（1）学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文，并向国家图书馆报送学位论文；（2）学校可以将学位论文部分或全部内容编入有关数据库进行检索和提供相应阅览服务；（3）研究生毕业后发表与此学位论文研究成果相关的学术论文和其他成果时，应征得导师同意，且第一署名单位为哈尔滨工业大学。

保密论文在保密期内遵守有关保密规定，解密后适用于此使用权限规定。本人知悉学位论文的使用权限，并将遵守有关规定。

作者签名：日期：年月日

导师签名：日期：年月日

致谢

实习实训期间，对于证券业务有了深入的了解，开发技术也得到了极大提高。感谢刘远超老师的悉心指导，每次在论文撰写方面遇到问题的时候总能及时的给予建议以及解决方案，使我在撰写论文方面受益非浅，也进一步激发了我对专业知识学习的兴趣。

在论文的开题、中期以及最后的撰写中，刘远超老师一直耐心的给予解答和指导。包括论文的选题，研究方向方法，论文的内容都十分的关注，教会了我很多，从细节中做好每件事情。

同时感谢我的校外导师石伟新，公司同事王立峰、陈驹、周琦等。他们在实习的工作生活中帮助了很多，让我非常快的融入到了团队中。在实习工作初期，和项目组一起参与项目的开发，石伟新教会了我公司的工作流程、开发技术和人际关系协调，让我受益良多。组内的其他同事也给了很多帮助，在和大家一起工作中提升了自己的技术能力，沟通交流能力。

感谢我的家人，在学习和工作中一直默默的支持着我。

最后，感谢学习工作中的每一个朋友和老师，是你们的关怀让我度过了两年收获的研究生时光。

# 个人简历

王云龙，男，汉族，1989年4月18日生于黑龙江省同江市。

2008年考入黑龙江大学软件学院软件工程专业。

2012年7月获得工学学士学位。

2013年9月通过全国统招考试考入哈尔滨工业大学软件学院软件工程专业，就读硕士研究生。

2014年6月—2015年5月于浙江核新同花顺网络信息股份有限公司，实习岗位为软件开发工程师，参与项目证券委托交易系统。