中图分类号：TP393 论文编号： 1028716 17-SZ024
学科分类号：081203

硕士学位论文

基于 DDS 的飞机协同设计数据分发系统的设计与实现

研究生姓名 钱哨
专 业 类 别 工程硕士
专 业 领 域 软件工程
指 导 教 师 陈 丹 副教授
庄 毅 教授

南京航空航天大学

研究生院 计算机科学与技术学院

二〇一七年三月

Nanjing University of Aeronautics and Astronautics
The Graduate School
College of Computer Science and Technology

Design and implementation of aircraft collaborative design data distribution system based on DDS

A Thesis in
Computer Science and Technology
by

Qian Shao

Advised by

Associate Prof. Chen Dan

Prof. Zhuang Yi

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Engineering

March, 2017

承诺书

本人声明所呈交的博/硕士学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得南京航空航天大学或其他教育机构的学位或证书而使用过的材料。

本人授权南京航空航天大学可以将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、汇编学位论文。

（保密的学位论文在解密后适用本承诺书）

作者签名：钱宁峰

日　　期：2017.3.28

摘 要

互联网技术的迅猛发展，以其极大的便利性不断融入各种生产和生活环境中，而协同设计尤其是当前学术研究领域的热点话题。协同设计等相关技术从早期就已经进入复杂的生产环境中，并凭借其便捷的设计部署方案与丰富的数据信息传递方式，使得多厂商之间协同工作完成大型工程项目。但是，大型项目生产环境具有特殊的复杂性与内在的不确定性，一般数据平台在对数据传输方式与信息的服务质量支持上存在明显缺陷。因此，国际对象管理组织(OMG)提出了基于数据分发服务(DDS)的一般性规范，对解决上述问题有深远意义。

本文基于数据分发服务的基础上，结合飞机协同设计相关背景，设计并实现了基于 DDS 的飞机协同设计数据分发系统。论文的主要工作如下：（1）结合飞机协同设计相关背景及数据，验证数据分发系统的可实施性。深入分析数据分发服务相关基础设施、标准、开发者文档和源码组织模块与体系结构。综合国内外研究现状，熟悉飞机协同设计环境下特点和适应性规则。（2）对数据分发系统进行需求分析，设计主要功能模块和改进模块。依据需求分析结果进行系统的总体架构设计和流程设计。（3）设计并实现基于 JSON 数据交换格式的飞机协同设计应用适配器。针对原始基于 XML 应用适配器的各项特点进行深入分析，为了改进其编解码与存储造成的过多性能消耗，提出了基于 JSON 的应用适配器来解决异构系统中数据传输方式不统一的问题。（4）设计并实现基于压缩布隆过滤器(CBF)和服务质量(QoS)兼容的自动发现算法。深入研究数据分发服务中发布订阅的自动发现机制，采用更适用于飞机协同设计环境下的 CBF，利用其替代原始过滤器，有效地降低了网络中传输的数据量，减缓了内存的消耗；运用更少的 hash 函数，降低了计算的复杂度。同时，可以获得更小的虚警率，提高了主题信息的匹配率。此外，添加 QoS 策略兼容性判定的设计与实现，极大地提高了主题的匹配率。（5）实现基于 DDS 的飞机协同设计数据分发系统的主要功能，其中包括增加、删除、修改和展示主题信息等基础功能和基于 JSON 的应用适配器组件与改进的自动发现算法模块。

现阶段，本文基本实现了针对飞机协同设计环境下，基于 DDS 的数据分发系统，并应用相关数据进行测试和实验。实验结果表明，在对原始系统各方面的优化与改善中都取得了良好的性能提升。

关键词：协同设计平台，DDS，数据传递方式，自动发现算法

ABSTRACT

The rapid development of Internet technology with its great convenience into a variety of production and living environment, and collaborative design, especially in the field of academic research hot topics. Cooperate with related technical design from the early has entered the complex production environment, and with its deployment scheme of data rich information transfer mode, makes the multi vendor to work together to complete large projects. However, the production environment of large-scale projects has their complexity, and the general data platform has obvious defects in the support of data transmission mode and information service quality. As a result, the international organization for object management (OMG) has proposed a general specification based on data distribution service (DDS), which is of great significance to solve the above problems.

In this paper, based on the data distribution service, combined with the background of aircraft cooperative design, we design and implement a data distribution system based on DDS. The main work of this paper is as follows: (1) to verify the feasibility of the data distribution system based on the background and data of aircraft cooperative design. In-depth analysis of data distribution services related infrastructure, standards, developer documentation and source code organization module and architecture. (2) the data distribution system needs analysis, the design of the main functional modules and improved modules. According to the results of the requirements analysis of the overall system architecture design and process design. (3) the design and implementation of an aircraft application adapter based on JSON data exchange format. According to the characteristics of the original XML application adapter based on in-depth analysis, in order to improve the performance of excessive consumption of decoding and storage by the proposed application adapter based on JSON to solve the problem of data transmission in heterogeneous system is not unified (4) the design and Implementation Based on the compressed bloom filter (CBF) and quality of service (QoS) automatic discovery algorithm compatible. Using CBF instead of the original filter, effectively reduces the amount of data transmission in the network; use hash function less, reduced the complexity of calculation. We can get a smaller false alarm rate and improve the matching rate of the topic information. In addition, the design and implementation of QoS policy compatibility decision is greatly improved. (5) implementation of collaborative design of the main functions of data distribution system of aircraft based on DDS.

At the present stage, the data distribution system based on DDS is realized in the environment of cooperative design, and the data are tested and tested. The experimental results show that good performance has been achieved in the optimization and improvement of the original system.

Keywords: collaborative design platform, DDS, data transmission mode, auto-discovery algorithm

目 录

图表清单

缩略词

缩略词	英文全称	中文全称
OMG	Object Management Group	对象管理组织
DDS	Data Distribute Service	数据分发服务
DLRL	Data Local Reconstruction Layer	本地重构层
DCPS	Data-Centric Publish-Subscribe	以数据为中心的发布-订阅
XML	Extensible Markup Language	可扩展标记语言
JSON	Javascript Object Notation	Javascript 对象符号
SDP	Simple Discovery Protocol	简单发现协议
BF	Bloom Filter	布隆过滤器
BFV	Bloom Filter Vector	布隆过滤器位向量
CBF	Compressed Bloom Filter	压缩的布隆过滤器
QoS	Quality of Service	服务质量
ACE	Adaptive Communication Environment	自适应通信环境
TAO	The ACE ORB	ACE 远程调用

第一章　绪　论

1.1 研究背景及意义

随着互联网技术近些年来的迅速发展，各领域的技术分支也逐渐细化，学科知识的专业程度也变得越来越高。为了适应各种应用程序复杂多变的需求，网络技术作为通信的基础条件，由硬件到软件的研究也极为全面且深入。其中，分布式网络用于通信不同地域，多个结点的数据交互，凭借其高可靠性，易拓展性获得了极为广泛的运用。如今，大规模精细化的设计与生产活动，尤其像飞机设计，各生产单位之间需要共享设计与生产的数据信息以便协调整体的生产活动得以有序，按时的完成。因此，基于分布式网络的多种应用方案为复杂场景下的精细化生产提供了良好的解决途径。

协同活动[1]是计算机科学技术中很重要的应用，是一种双方相互协作，进而深入了解，再高效协同的过程，即是发生活动的对象共同协作设计的过程。这种交互的过程本身实质是协作对象为了共同的目的而同意，妥协，赞同的迭代。同样的，在复杂大型的生产环境下，协调多方数据信息，增强交互效率以降低沟通成本，进而减少人员和事物的损耗都依赖于协同设计的有效应用。

飞机的设计与生产是一项极其复杂的工程。特别地，设计阶段各模块高效协同应用对后期生产工作起到决定性的作用。而且辐射范围也涵盖了整个飞机的生命周期。而且设计过程决定了飞机整个生命周期绝大部分的成本。以往的飞机设计与生产过程普遍应用串行的设计模式，使用这种模式是为了保障设计与生产过程严谨有次序的进行，飞机中的每个部件及部件的设计与生产单位之间存在流程上的顺序限制，在一项复杂任务中，一个单位必须等待另外一个单位完成任务后才能开始工作而且在任务上也存在一定的优先级，致使随着任务的复杂度增加，各单位之间的协调工作变的愈加困难。而且各单位之间都拥有自己的专业软件系统，在数据信息传递时交流成本很高，缺乏统一的传递与协作标准。此外，飞机中各部件设计与生产过程中涉及的多学科多领域的专业知识，单位相互之间处在不同的运行环境之下，应用不同的数据储存方式、本地专业操作系统和编程语言，这些根本上的差异致使数据信息交互成本再次提高。因此，过去滞后的开发操作方式带来了极大地不变，使得系统中各结点的互联互通更加困难且相互独立封闭，无法真正成为统筹角度上的网络中信息结点，种种原因导致开发流程延缓，数据信息交互滞后，开发流程紊乱等现象，最终使得整体飞机设计与生产过程周期变长，成本加剧。正因如此，现代飞机协作设计过程做出了许多改善，首先是设计各单位发挥自身优势，分享各自经验和资源知识，部门与部门，单位与单位之间促进友好交互关系。其次，制定统一标准供各协作部门实施执行。

飞机协同设计作为一种庞大的工程体系，其具有分布式地域、实时协作、多级学科知识等特点，系统中各结点相当于多领导小组的协作，用以高效地开发分布式系统产品的生产与

开发。相互协作，高效互通是其最主要的功能特点之一。现阶段针对各部门运行环境和物理条件的种种差异，最有效的方式是保持现有各专业软件系统的独立性，统一架构各结点的消息通信机制。目前，基于开源项目 OpenDDS(DDS：Data distribution service)进行二次开发实现的拓展系统用于提供多种数据访问与交互服务，而本文在飞机协同设计环境下，为了实现各部门内部专业软件系统的数据交互通信更加需要一种数据分发系统，因此深入理解 DDS 内部核心技术规范成为接下来工作的重心。

1.2 国内外研究现状

DDS[2-4]作为数据分发体系中的主要标准，相较于公共对象请求代理体系结构(CORBA：Common Object Request Broker Architecture)、Java 消息服务(JMS：Java message service)，对以数据为驱动的数据分发体系结构的研究分析与规范标准的拟定而言，存在一定的滞后性。在 21 世纪之前，基于数据分发服务的中间件在通信的灵活易用性，快捷方便性和可扩展性上都存在很大的改进空间。文献[5]中指明，CORBA 技术作为对象管理组织(OMG：Object management organization)组织的一种标准，是以服务及对象为主要目标，提供面向对象应用级别的体系规范。其中，CORBA 技术使用 C/S（客户端/服务器）通信模型，通信协议和内部协调机制较为复杂并且伴随着海量的数据流量访问容易出现服务器瓶颈等问题。因此，CORBA 更适合数据流量不算大且实时性要求不高的分布式处理服务。JMS 技术是 Java 消息通信服务，内部使用了发布/订阅的消息通信模型，但仅支持单一的编程语言，缺少应用级别的服务质量策略，使其也只能服务于低实时性，非紧急的任务。

DDS 技术一直以来缺乏强有力的标准和规范，因此，国际对象管理组织在分析了大量现有方案的缺陷后，依照 DoD 美国国防部信息技术中心的相关注册标准，开始进行大量数据分发相关知识技术的研究和实现操作。21 世纪初期发布了面向分布式实时系统的 DDS 标准 1.0 版本[2]。同时，我国由科学技术部颁布了数据分发服务指南与规范，旨在规范分布式网络数据分发服务的开发流程，制定数据分发共享数据的过程及需要遵循的标准。在随后的几年中 DDS 的 1.1[3]和 1.2[4]版本也相继问世。DDS 标准中提供了详尽的中间件使用指南和完备的第三方开发者文档。

当前，面向数据分发服务的研究主要分为两大部分。一方面是针对数据分发服务中间件技术的研究，并且已经产生了许多较为成熟的数据分发服务中间件产品，如 OpenDDS、OpenSplice DDS 等。另一方面，针对数据分发服务中自动发现算法的研究也长期成为学术研究的热点话题之一。

1.2.1 数据分发服务中间件技术研究现状

随着中间件技术的不断发展，国际上涌现出一批相对成熟的中间件技术产品，如 CORBA，COM/COM+和 J2EE。CORBA 作为国际对象管理组织内部推出的标准之一，其自身具有良好的对外开发性，也有部门产品对外开源，最大的优点在于其独立于特定的软硬件

平台，编程语言和网络环境。也因此其受到广大商业软件的青睐，推出了许多相对成熟的商业软件。COM/COM+主要应用于 Windows 平台，是微软公司利用其庞大的原始技术积累设计实现完成。J2EE 是采用 JAVA 编程语言实现的面向对象的中间件产品，因语言本身，所以具有跨平台等特点。中间件技术的不断发展涌现出多分支体系结构，尤其面向对象的中间件技术受到了学术界、商业和军事等领域极大地关注。基于分布式中间件的研究长期以来一直是学术研究的极大热点之一。多种中间件产品强调数据信息交互的高效性与可靠性，设计良好的数据交互方式是产品研究的重点之一。将具体的应用集成进中间件也是主要的开发拓展手段之一。IBM，Microsoft，Google，Amazon 等主要的中间件生产厂商越发强调信息交互与数据消息总线机制这一中间件核心技术的协调。此外，国内的中间件技术也逐步走向成熟，各企业集团为了建立起自身的技术积累与后期业务拓展更好的掌控都开发了自己的中间件。其中较为常见的有 A2E-DataIntegrator[6]、TongIntegrator[7]和 Inforbroker[8]。

OMG 拟定的 DDS 逐渐成为数据分发服务体系中的核心标准，正极大地运用于各种商业和军用的分布式软件系统之中。其中最具影响力的应用是美军 DoD 国防部的开放式架构计算环境 OACE（Open Architecture Computing Environment）以及舰艇自卫系统 SSDP（Ship Self-Defense System），前后者都依靠 DDS 规范建立起强大的中间件系统，同时被作为美军未来军事领域主要的战略部署[9]。

在商用领域中，美国 RTI 与法国 THALES 集团以 OMG 的 DDS 标准研发出了多种数据分发服务中间件产品且普遍应用于航空航天、工业自动化控制、通讯、军事及智能交通等多领域中。但我国国内的分布式实时通信系统环境相关的数据分发服务的研究与探寻仍旧处在萌芽阶段，其中基于设计相关的中间件系统平台及产品的研制仅处于初期[10]。文献[11]中指明我国对数据分发服务的研究仅停留在对已有成熟开源的中间件产品的再次开发和业务拓展，同时各大厂商都根据自己的规范来实施开发，因此业界开发的实施标准难以统一。

国内军事领域的 DDS 产品也相对成熟，文献[12]指明国内对军事战术信息相关的数据分发服务技术已经进行了深入的研究且在数据信息的发布/订阅、数据信息的检索与数据信息的智能过滤都取得一定的科研成果。文献[13]中对原有数据分发服务中间件的基于动态服务的自动发现算法和容错机制的不足进行了分析和改进，使得现有发布/订阅机制中的数据信息集成的集中存储结构转换为 P2P 的分布式体系架构，同时通过双层 Chord[14]将发布/订阅机制动态化，最终使数据分发服务中间件能够自适应主题数据信息的匹配。

当前，依据 DDS 开发且相对成熟的开源项目主要有 OpenSplice DDS[15]、Java-based DDS[16]、CoreDX[17][18]和 OpenDDS[19]。OpenSplice DDS[15]，该产品是由 PrismTech 公司提供，产品本身是对 OMG 拟定的 DDS 规范的完全兼容，其内部的实现机制提供了所有对数据本地重构层(DLRL：Data Local Reconstruction Layer)内部对象的描述和基于数据驱动的发布/订阅(DCPS：Data-Centric Publish-Subscribe)的相关描述。OpenSplice 的核心模块提供了发布/订阅功能的接口函数，其中要求实时的信息传递且为重复发布者提供了基础的功能支持。

文本订阅和持久性方面提供了附加的数据信息管理特性，以确保高效地数据信息传递及高度的数据有效性和强劲的过滤与查询功能。在 DLRL 层上对相关主题集合采用面向对象的角度建立视图，使其具有面向对象的特性。在生产力工具组件上，是基于 Eclipse 的高效功能组件，支持多级应用的跨平台交互操作、多种编程语言对象和信息建模。在系统组织架构方面，OpenSplice 具有高拓展性和很强的伸缩性和适应性，内部采用共享内存的方式来联通网络中多节点中的多种应用程序，其多种服务提供了热插拔的功能。

Java-based DDS[16]，由法国格勒诺布尔大学教授根据教学实践中的经验总结，利用 JAVA 编程语言编写的数据分发服务。其基于 ORB 技术中的事件驱动的服务开发完成。CoreDX[17][18]，是 TOC 公司基于 C 编程语言开发完成。是专门为嵌入式系统设计开发的 DDS 体系结构。其更强调提高通信质量和性能，采用最新的网络协议，以期提高系统的吞吐量和网络的时延。

OpenDDS[19]，由美国公司根据 DDS 标准规范的 1.0 版本开发完成。其内部是基于 C++ 编程语言编写，依据对象管理组织的规范完成的数据分发服务。OpenDDS 基础架构采用 ACE/TAO(Adaptive Communication Environment/The ACE ORB，是基于 ACE 基础上的 CORBA 实现框架)这种自适应通信交互环境(为跨平台与可移植提供保障)，在使用时只需要简单配置其环境变量和编译参数，便可以利用其中主要的发布/订阅等功能。开源项目 OpenDDS 在随后的版本中更新了对 OMG 的数据分发服务 1.2 等版本的支持，其核心功能发布/订阅也做出修改，支持高效通信的实时发布/订阅通信协议的 DDS-RTPS 规范[20]，而且提供了对应 JAVA 消息服务的接口函数。

在多样的分布式应用程序中，为了高效的分发数据信息，DDS 内部制定了两种具体的数据模型，即平台无关模型与将平台无关模型映射到 CORBA IDL 实现的平台相关模型。而平台相关模型又被分为两种接口层，以数据驱动的发布/订阅接口层和数据本地重构接口层。前者通过主题数据信息、发布者和订阅者提供的对应数据信息，实现发布者与订阅者的数据交互与通信，即是将发布者发布的数据信息高效地传递给订阅者。后者则更加强调本地数据对象的访问，目的是运用远程局部对象访问本地数据集合以实现分布式的数据共享。开源实现 OpenDDS 的最新版本只是实现了基于发布/订阅的模型，对于数据本地重构的开发还处于初期阶段。

OpenDDS 最大优势在于配置过程简单清晰，包含基础功能全面，开发者文档与论坛详尽，在此基础上进行的二次开发会隐藏许多底层的实现细节减缓开发难度。但其代码规模庞大，项目分支繁多，想要从源码上整体把握 DDS 规范会比较困难，并且项目中使用技术种类繁多，想要深入理解其中构成需要耗费大量精力。此外，尽管 OpenDDS 实现功能全面，但应对复杂业务的开发需要仍然需要进行大量的修改。最后也是最重要的，在 OpenDDS 中基于发布/订阅的自动发现算法实现方式过于落后，不能够满足高速发展的大数据时代实时网络下比较高的数据传输与应用效率。

目前，针对分布式中间件开发的数据分发系统主要应用于军事、航空航天等封闭式环境中，现有可应用的系统都存在各种版权或功能上的限制，所以更加需要针对某种特定平台环境下定制相应的数据分发系统，而 OpenDDS 在多方面的功能支持都有利于分发系统的构建，其中包括为典型的发布/订阅功能提供支持，兼容异步多节点之间的实时高效通信手段，对大型分布式环境下节点高效交互的高保障性和高可靠性等。因此，基于 OpenDDS 的研究逐渐成为本文学术活动的重点之一。

1.2.2 数据分发服务算法研究现状

当前 DDS 的规范及其内部核心的实现机制仍然引起国内外众多学者的研究与探讨。在众多研究中不乏针对数据分发服务核心机制自动发现机制的探寻。文献[21]中指出 DDS 标准中简单自动发现协议(SDP：Simple Discovery Protocol)要求使用者预先静态配置网络中的各节点信息，不能够很好的支持大规模动态网络。所以，为了适用于大规模动态网络，文章提出了一种自适应动态网络服务发现机制。主要实现数据分发服务的实体数据信息与共享数据信息高效地联系，并且具有可见的通信标识及与标准模型无缝对接。但是存在不能预先判定主题信息是否相同等缺陷。

文献[22]发现当 OMG 组织拟定出 DDS 规范之后，网络中各系统节点在数据交互前期会有一种自动发现协议来定位并获取远程 DDS 实体及其属性。尤其当 DDS 发现匹配数据写入者和数据读取者实体位于不同的网络节点。其中 DDS 说明没有明确指定这种发现在网络中的具体实现过程。为了提供不同 DDS 实现之间的互操作性和透明性，OMG 组织标准化 DDS 互通协议(DDS-RTPS)。任何兼容 DDS-RTPS 的实现必须支持至少简单发现协议。这种 SDP 一般工作于相对小的或中型的网络环境之下，但是它不能随 DDS 端点数量的增加而拓展。即 SDP 不利于大规模环境且不易于系统拓展维护。文章基于布隆过滤器（BF：Bloom Filter）的匹配方式提出了一种自动发现算法，简称 SDPBloom，即运用 BF 来增加 DDS 的可扩展性。BF 为空间高效概率数据集合表示运用 hash 函数。经过实验的分析和论证，此种方式就网络加载和节点资源消耗而言能够有效地改善自动发现的处理。

文献[23]中为 DDS 提出了改进的 CBF（Counting Bloom Filter）[24]，这是一种新颖的自动发现协议。在现有的自动发现协议，所有的参与者发送它的端点数据给网络系统中其他参与者，并且从其他参与者获取端点数据。即便能够有效地降低内存使用消耗，但 BF 本身含有虚警率的缺陷。在大规模网络中，如海军军舰，需要大量的内存来存储所有的军舰各项数据。大多数情况下，大部分数据存储不需要通过端点，但是占用大量内存。为了减少数据发送和存储的大小，文章提出了一种结合 CBF 的自动发现过程，算法内部引入了 hash 表，旨在消除原始的虚警率。文章为过滤器的构建和在海军军舰网络拓扑总的发现时间提出了延迟时间的衡量指标。实验结果表明该方法获得了更低的延迟时间和少量的虚警率。但该方案主要不足在于不能预先判别 QoS 策略是否相互兼容。

文献[25]中提出的基于服务力向量（SAV: Service Ability Vector）的自动发现算法，考虑

到 SDPBloom 自动发现算法不能够在参与者发现阶段预先对各节点的 QoS 策略进行兼容性判定，致使网络系统中大量节点的 QoS 不兼容，因而消耗了大量的网络带宽资源。基于上述各种缺陷，文章提出了一种服务力向量，基于发布/订阅机制来构建 BF 向量和 SAV，利用两种向量对网络系统中各节点的主题名数据信息和 QoS 兼容性策略进行匹配，从而降低了网络带宽的损耗，在一定程度上处理了因 QoS 不兼容带来的问题，初步提升了主题的匹配率。但文章只是对 QoS 进行粗判，无法精确匹配，并且没有降低 BF 带来的虚警率问题，所以仍然存在改进的空间。

综上所述，遵循 OMG 组织制定的 DDS 规范，应用开源实现项目 OpenDDS 进行二次开发，改进规范中原有的自动发现算法，同时采用丰富的 QoS 兼容性策略来定制不同业务需求的功能，使得系统能够更好的适用于多种协同工作与环境之中是本文研究工作的又一重点。

1.3 论文主要工作

本文主要研究工作是在飞机协同设计环境下，设计并实现基于 DDS 的数据分发系统。需要对数据分发服务的相关标准、代码，飞机协同设计环境下的各模块特点与功能，异构系统中如何协调网络对端的数据传输方式以及基于 Bloom filter 的自动发现机制等方面都进行全面且深入的研究。因此，本文主要研究工作分以下几个方面：

（1）结合飞机协同设计开发环境并应用相关数据，初步验证基于 DDS 的飞机协同设计数据分发系统的可行性。深入理解与 OpenDDS 相关的基础组件，查阅开发者文档，分析数据分发服务源码梳理逻辑结构。权衡比较国内外在不同适用场景不同维度下，数据分发服务的适用特性与应用性。

（2）对基于 DDS 的飞机协同设计数据分发系统进行需求分析，分析其主要功能模块和可能的改进环节，设计并规划实现方案。根据需求分析结果对数据分发系统进行总体的架构设计和数据通信过程中的流程设计，深入探讨各功能模块的具体实现细节。

（3）设计并实现了基于 JSON 数据交换格式的飞机协同设计应用适配器。在综合探究系统各部分组成结构与功能的前提下，考虑到飞机各部件生产单位因部件个体差异性，导致各自使用的资源方式的不同，为了使系统各节点高效地共享生产数据，也即网络通信对端获得统一的数据传输方式，设计并实现了基于 JSON 数据交换格式的飞机协同设计应用适配器，同时也比较了其与原始 XML 方式的各种不同。在确定前者简化了封装，解析和存储等优势后，通过基于飞机协同设计的主题数据进行了实验的论证。实验表明，本文提出的适配器应用方案获得了更优的编解码和存取性能。

（4）提出并实现了基于 CBF 和 QoS 兼容的自动发现算法。在理解了原始基于 Bloom filter 的发布订阅自动发现机制后，深入探索了近些年来针对此问题的相关改进工作，采用了更加适用于飞机协同设计环境下的 Compressed Bloom Filter。CBF 的应用相较原始过滤器在网络中的数据传输效率更高，传递数据量更少。也因此可以用更少的 hash 函数来获得比

以往更优的错误率。同时，在 CBF 的基础上，提出了基于 QoS 判断机制，设计了网络中通信对端支持的 QoS 判别规则表，依据规则实现了预先甄别方法，并针对飞机协同设计的主题数据进行了实验论证。实验表明，基于 QoS 的方式极大地提高了主题的匹配效率。

（5）根据当前可应用的飞机各部件主题发布订阅的交互数据，模拟真实生产和作业环境，本文实现了基于 DDS 的飞机协同设计数据分发系统。其中包括增加、删除、修改和展示主题信息等基础功能，也包含基于 JSON 的应用适配器组件和改进的自动发现算法模块。

本文的新颖之处在于，从更加底层的数据传输格式出发，考虑到飞机协同设计环境的因素，选用更加轻量的 JSON 数据交换格式，替代原始 XML 格式，在本质上提升了网络中数据编解码的性能。同时，考虑到原始过滤器不能高效地传输数据且缺乏对服务质量的支持，在权衡多种解决方案后，选用 Compressed Bloom Filter 用以降低网络中实际需要传输的数据流量，可以运用更少的 hash 函数，从而降低在构建过滤器的过程中的计算量，并能获得更优地错误率。基于以上研究结果，本文又提出了基于 QoS 的设计与实现，充分匹配服务质量选项，从而提高了主题的匹配率，再次降低了网络数据流量。

1.4 论文组织结构

本文主要是在飞机协同设计的环境下，基于数据分发服务中的核心部分，如发布订阅的自动发现机制等。设计并实现数据分发系统。并通过相关数据对主要的功能部分进行测试，以期达到优化原始设计的目的。全文共六章，具体安排如下：

第一章：绪论。首先简要介绍协同设计当前研究的趋势。之后从分布式中间件和数据分发服务两个方面介绍了国内外相关的研究状况。最后，具体阐述了本文主要的研究方向以及每个阶段的工作重心。

第二章：基于 DDS 的飞机协同设计数据分发系统的设计。首先阐述了基于数据分发服务相关的若干关键技术。之后对基于 DDS 的飞机协同设计数据分发系统进行需求分析，在深入分析数据分发系统的功能需求和相关性能需求的基础上，设计规划基于 DDS 的飞机协同设计数据分发系统的总体架构和流程。

第三章：基于 JSON 数据交换格式的飞机协同设计应用适配器。首先介绍当前 XML 的主流地位及两种解析方式，并引入 JSON 的相关简介。在理论分析的基础上，详细展开了基于 XML 的飞机协同设计适配器模型，充分阐述了 XML 格式与解析流程。然后，引入基于 JSON 的飞机协同设计适配器模型，其中对两种格式进行了全面且深入比较，也给出了改进后的示例模型和解析过程。最后使用飞机协同设计相关数据进行测试验证。通过封装，解析，存储与压缩实验数据表明，JSON 相较 XML，在飞机协同设计环境下对系统整体存取时间性能平均有 15.68%的提升，同时，在压缩率上也有一定的优化。

第四章：基于 CBF 和 QoS 兼容的自动发现算法。首先介绍了发布/订阅功能在数据分发服务中的核心地位，从而引入自动发现机制这一重要内容。之后阐述了原始基于 Bloom filter 的自动发现机制的工作原理与实际应用。在此基础上，引入 Compressed Bloom filter 的推导

过程，给予性能优化的可行性证明。然后，提出了基于 CBF 和 QoS 兼容的自动发现算法，设计了网络通信对端的规则协议表，实现了用于规则判断的相关逻辑代码。最后使用飞机协同设计相关主题数据进行测试验证。通过算术编码压缩、hash 函数对比和基于 QoS 的预先匹配实验数据表明，本文提出的自动发现机制在减少网络中传输数据量的同时，可以使用更少的 hash 函数来达到更优的错误，QoS 的设计能够极大地提高主题的匹配率。

第五章：基于 DDS 的飞机协同设计数据分发系统的实现。综合飞机协同设计应用适配器和基于 CBF 和 QoS 兼容的自动发现算法等核心模块，实现了基于 DDS 的飞机协同设计数据分发系统。基本满足需求分析阶段期望达到的相关功能，并且对原始模块的各种性能优化也取得了预期的效果。

第六章：总结与展望。整篇论文内容的梳理与归纳，深入分析各改进细节，并就可能拓展的研究方向给出详细的说明。

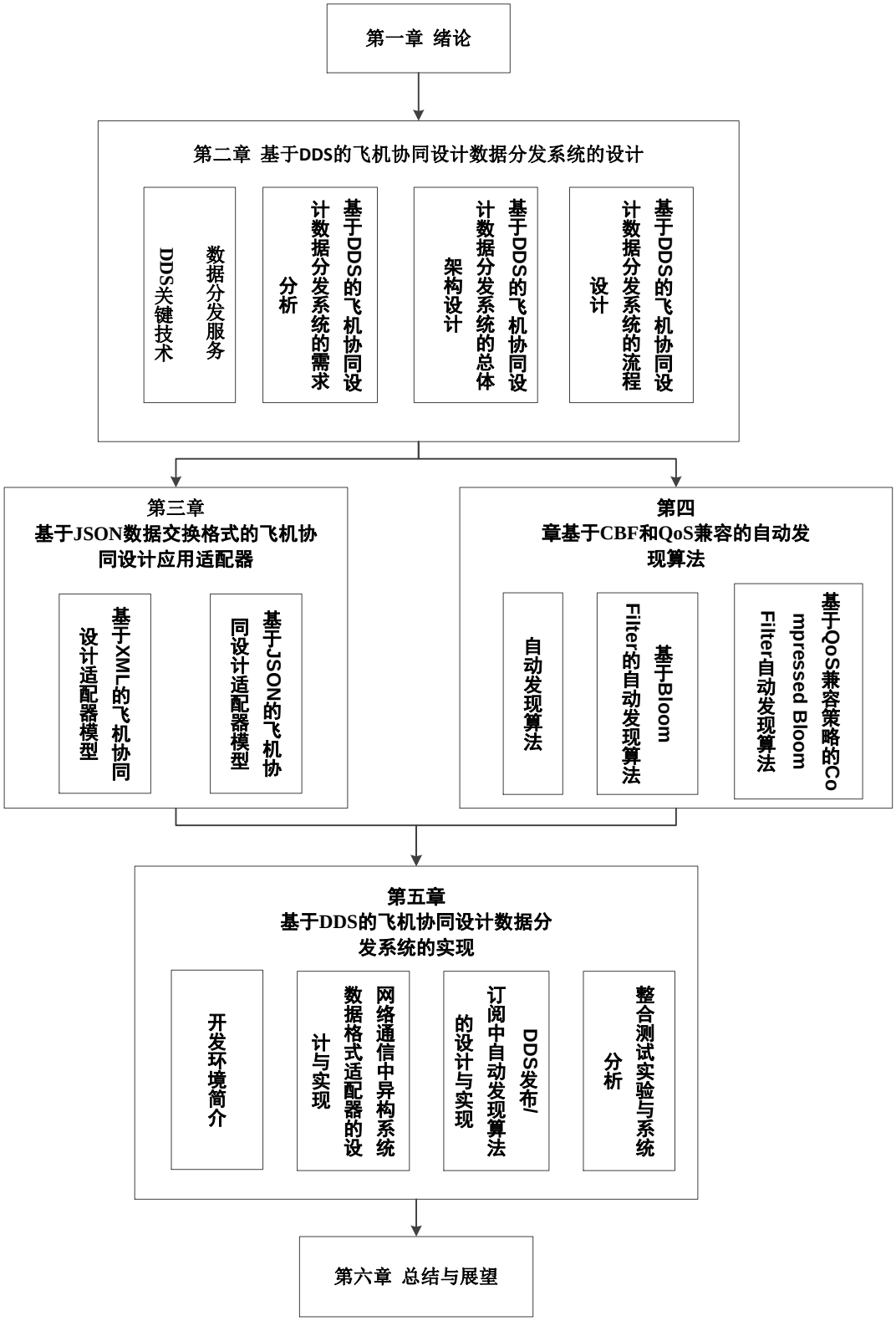

图 1.1 论文组织结构

第二章　基于 DDS 的飞机协同设计数据分发系统的设计

本章首先对基于 DDS 的飞机协同设计数据分发系统进行需求分析，阐述了飞机协同设计使用数据分发系统的必要性；根据数据分发系统的各项功能与模块，给出了数据分发系统的总体架构设计及流程设计；最后，论述了数据分发服务在飞机协同设计背景下的应用及 DDS 中涉及的几项关键技术进行分析。

2.1 基于 DDS 的飞机协同设计数据分发系统的需求分析

飞机协同设计实质是设计者之间相互通信和数据共享的过程，因为其设计过程的特殊性，飞机各部件要保持一定顺序设计，所以数据交互的高效性显得尤为重要。同时，各飞机部件制造厂商只熟悉当前自己单位所专业的部件知识和制作工艺，对于其他厂商的情况不甚了解，事实表明飞机协同设计同时涉及多学科知识和多技术范畴的协同过程。飞机协同设计是协同设计的子集，所以他们之间会存在许多共同的特点。

首先显而易见的，飞机协同设计是大规模分布式的协作开发过程，要求同属于统一地区或者不同地区的飞机部件厂商协同工作。协同设计过程中强调信息数据的高效互通，实时动态地传递和呈现各飞机部件的研发进度和任务流程等多方面的数据信息。此外，对于单个任务各部门之间的衔接具有一定的时效性，任务完成时需要整理信息并重建新的任务群组。最后，协同开发过程中，开发过程往往会存在互相冲突和资源的竞争问题，各部门的合作协调以及任务的优先级分配是解决问题的关键所在。

现有的各飞机部件制造厂商存在数据交互效率不高等问题，致使飞机协同设计开发流程紊乱，开发周期延长。针对以上问题设计飞机协同设计数据分发系统，功能需求如下：

1）信息交互域范围配置：因为飞机协同设计过程涉及多部门之间的信息协调，各个部门可能存在物理地域上的不一致，并且某些部分只需要与特定部门之间进行数据交互，而不用将自身数据信息广播给网络中的其他部门，所以，信息交互域的范围控制是当前的需求之一。

2）异构系统数据适配器：在飞机协同设计中，各厂商间使用的专业软件系统不尽相同，又因各自需求和生产环境等因素的不同，造成使用的操作系统、编程语言、数据库等方面存在差异。为了解决异构系统间数据格式不统一且提升原始数据适配器的存取时间性能，优化现有适配器，提升系统整体性能降低损耗显得尤为迫切。

3）实时的信息交互共享：飞机协同设计中是多部门的信息交互，部门间信息交互效率低下，部门分支错综复杂，需要相互统一协作。因此，实时高效地数据信息交互能够有效地缩短飞机协同设计的时间周期。

4）良好的应用设计：飞机协同设计本身是一种协同设计的思想研制和管理飞机的综合性工程技术，为了适应快速发展互联网时代带来的频繁的业务拓展，迫切需要设计良好的软

件架构来适应新/旧部门的加入与淘汰。

5）QoS 服务质量配置：飞机协同设计各部件制造厂商和部门单位数量庞大且各部门都存在自己的业务需求，需要提供丰富的服务质量策略的支持及人性化的配置方式。此外，服务质量与各飞机部件主题的良好衔接用以满足不同用户，不同部门的多样化需求。

6）自动发现算法的改进：在数据分发服务中，需要根据结点之间的主题信息是否相同以及 QoS 策略是否兼容来最终确定主题是否匹配。而基于布隆过滤器的自动发现算法在各结点之间传递数据时效率较低。而且，不支持 QoS 兼容性判定，导致最终的主题匹配率很低。当前研究采用服务力向量算法只是对 QoS 进行了粗判，主题匹配率仍然不高。所以急切需要改进的自动发现算法用以降低网络中传输的数据量，减缓了内存的消耗并且提高主题信息的匹配率。

2.2 基于 DDS 的飞机协同设计数据分发系统的总体架构设计

基于 DDS 的飞机协同设计数据分发系统的总体架构分为 3 层，自下而上依次是本地资源层，系统服务层和应用层。系统服务层作为整个数据分发系统的核心层，承载着应用层与本地资源层的衔接工作，将上层的应用程序指令传递给下层进行数据更新调配，再将下层的数据改变传递给上层的应用程序界面做呈现。具体结构如图 2.1 所示。

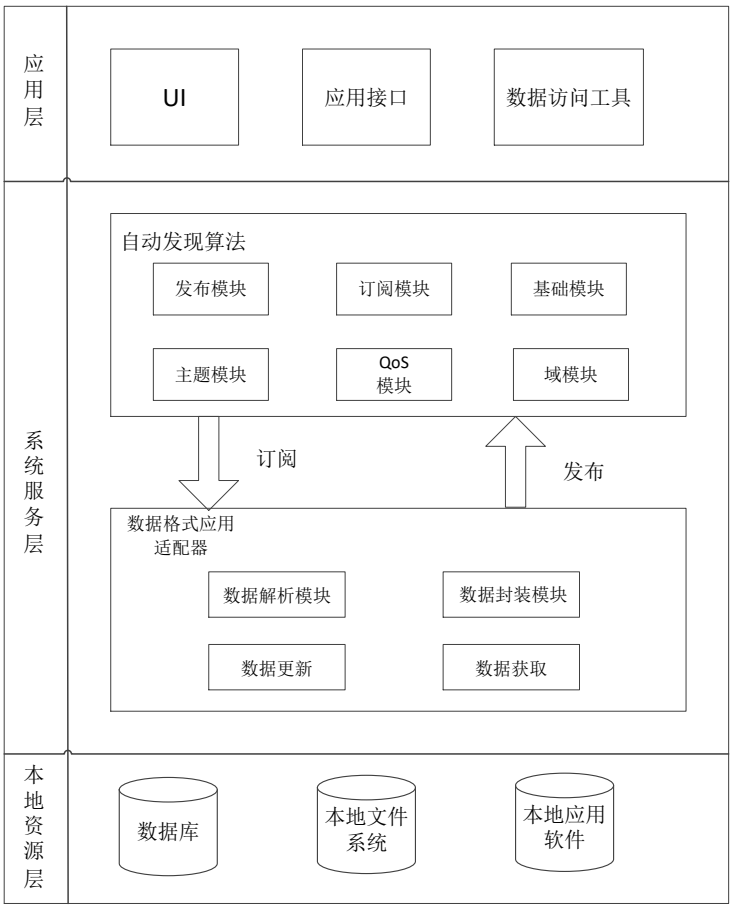

图 2. 1 基于 DDS 的飞机协同设计数据分发系统的总体架构图

本地资源层：作为整个基础架构最底层的部分，本地资源层提供了诸如数据库，本地文件系统，各种本地应用软件等多项设施。所有的基础设施最终都是服务于飞机协同设计中各项任务所产生的数据信息。

系统服务层：系统服务层最核心的功能组件包括自动发现算法部分和数据格式应用适配器部分。数据格式应用适配器部分用于数据的封装与解析，主要包含数据解析/封装模块、数据更新和数据获取。数据解析模块主要负责对接收的数据信息进行翻译解析成本地专业软件系统统一使用的数据格式。数据封装模块主要负责对获取到的数据信息进行封装，处理成统一的消息格式并传递给发布模块进行后续操作处理。数据更新主要负责依据数据解析模块解析的数据信息来更新本地数据库或者本地文件系统中的数据文件信息。数据获取主要负责获取数据库中改动的数据信息提取之后传递给数据封装模块进行封装。自动发现算法内容部分主要包含发布/订阅、基础、主题数据信息、QoS 兼容性策略和域模块。发布/订阅模块主要负责网络通信中各节点发布与订阅主题信息的相关功能。基础模块提供了组件的底层基础性功能。主题模块主要负责飞机协同设计中主题信息的定义。QoS 服务质量模块提供了丰富的策略定义，为定制多样的主题类型提供了多种的选择。域模块提供了域范围的划分和权限的控制。

应用层：应用层位于最上层，更加面向用户，提供最方便易用的功能及操作界面，是用户操作应用程序的接口。主要负责对底层数据变化做出对应的界面元素的改变。

2.3 基于 DDS 的飞机协同设计数据分发系统的流程设计

如图 2.2 基于 DDS 的飞机协同设计数据分发系统的流程设计图中展示了系统的整体操作流程，由以下几步构成：

1）系统各项信息的初始化配置，各部件厂商初始化自己的主题信息。

2）如果作为主题的发布者，有本地系统的信息感知组件监控数据中数据的变化并实时提取改动数据。

3）如果作为主题的订阅者，等待进入下一流程。

4）发布/订阅者首先创建各自的域参与者，接着分别创建发布/订阅者并注册对应数据类型。最后，创建发布/订阅主题，创建数据写入/读取者，使数据写入/读取者开始监听数据。

5）主题发布者发现有数据发布时激活写入者，通过数据格式应用适配器对数据进行封装并发布主题信息。没有数据发布时继续进行数据监听。

6）主题订阅者发现有数据到达激活数据读取者，获取数据进入下一流程。没有数据达到时继续进行数据监听。

7）主题订阅者运用数据格式应用适配器对数据进行解析，解析成本地专业软件系统统一的数据格式并存入数据库中，同时做数据改动的界面展现工作。

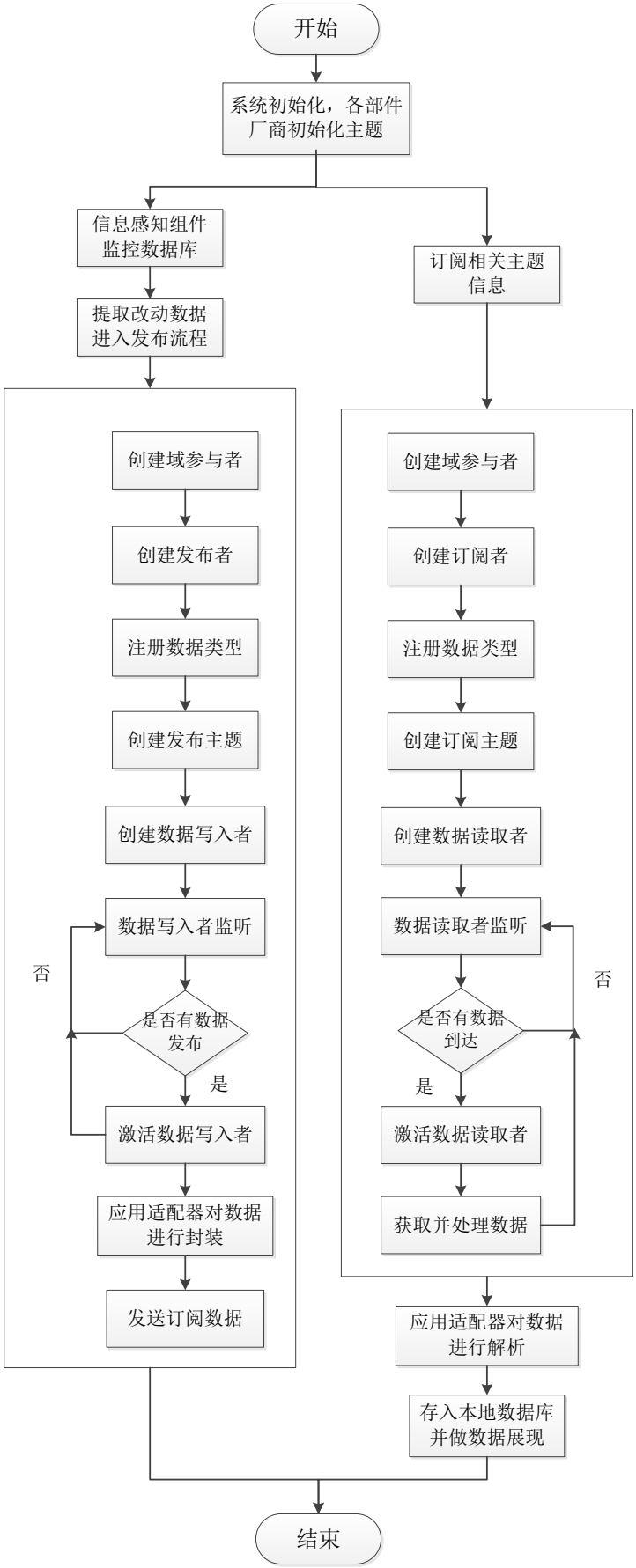

图 2. 2 基于 DDS 的飞机协同设计数据分发系统的流程设计图

2.4 数据分发服务（DDS）关键技术

本文提出的基于 DDS 的飞机协同设计数据分发系统，能够高效地协同各厂商异构系统的数据传递，在满足各厂商进行发布订阅功能需求的前提下，替换了原有的数据传输格式，从根源上提升了数据传输效率。同时，改进了自动发现算法，使得数据传递的过程更加高效，主题匹配率更加准确。在此过程中，有以下几项关键技术。

2.4.1 自动发现机制

自动发现作为数据分发服务的核心技术，是指参与通信网络中的各结点都能够主动地发现其他结点[26][27]，以此作为网络中各结点交互通信的前提条件。以数据驱动，基于发布/订阅的数据分发服务规范中，当结点加入或者离开系统时，自动发现机制都会自动地以某种方式通知各结点，系统中结点信息的变化情况。自动发现机制主要工作于各结点传递数据消息之前，提供高保障的数据通信服务。基础的自动发现机制模型结构如图 2.3 所示：

<1> 数据包解析：自动发现机制初次获取网络通信环境中对端传递的各通信节点的数据信息包，依据各数据信息包中提供的匹配信息来进行解析匹配并提取远程节点的相关信息。

<2> 数据信息交互匹配：自动发现算法会批量的匹配本地接收到的数据信息包，数据信息的匹配率是区分算法优劣的标准。设计并实现动态高效地自动发现算法一直以来都成为数据分发服务主要的研究工作。

<3> 端点匹配入库：整个匹配过程依附于 GDS 提供后台支持，当远程端点信息与本地端点信息相匹配时，将远程相关端点信息存入本地 GDS 中。

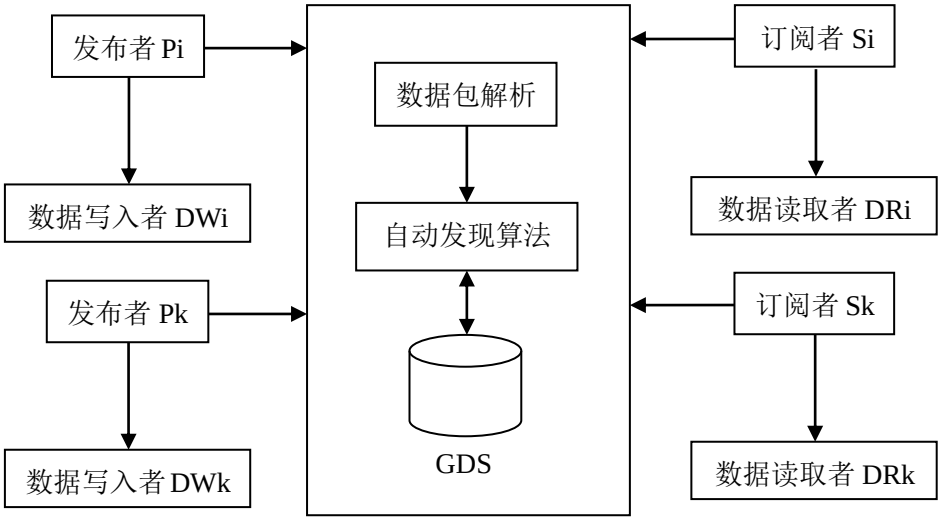

图 2. 3 自动发现机制模型结构

2.4.2 服务质量控制

DDS 标准中定义了大量丰富的 QoS 兼容性策略，通过其来控制本地和端点之间的 DDS 实体属性，对于前者 QoS 的控制会极大地影响资源的利用率，而后者的控制对数据分发系

统中数据传输效率与内存使用率起到至关重要的作用。应用程序可以通过设置内部 QoS 策略来对网络中传输的数据进行管理、控制和优化。如表 2.1 中给出了在数据分发服务中多种 QoS 的策略选项。其中主题(T：Topic)、数据写入者(DW：DataWriter)、数据读取者(DR：DataReader)、发布者(P：Publisher)、订阅者(S：Subscriber)、域参与者(DP：Domain Participant) 、域参与者工厂(DPF：Domain Participant Factory)。

表中 RxO 模式表示当前 QoS 策略是否满足于 Request vs Offered 模式[28]，也即发布和订阅方使用过程中是否需要进行 QoS 策略的兼容性检查和判别。一般分为三种情况，当此特性为 YES 时，表示该策略可以被设置用于发送方和订阅方，并且要求对该特性值进行兼容性检查和判别，此时的兼容特性值必须明确地被定义。当此特性为 NO 时，表示该策略可以被设置用于发送方和订阅方，并且要求对该特性值两者的设置时相互独立的，此时所有可以相互组合的特性值都可以被认为是相互兼容的。当此特性为 N/A 时，表示该策略可以被设置用于发送方或者订阅方，并且要求对该特性值两者不能同时设置，此时不需要协同双方的兼容性。

表 2. 1 DDS 中所支持的 QoS 策略

QoS 策略	适用范围	RxO 模式	可修改性
DURABILITY	T, DR, DW	Y	N
DURABILITY SERVICE	T, DW	N	N
LIFESPAN	T, DW	N/A	Y
HISTORY	T, DR, DW	N	N
PRESENTATION	P, S	Y	N
RELIABILITY	T, DR, DW	Y	N
PARTITION	P, S	N	Y
DESTINATION_ORDER	T, DR, DW	Y	N
OWNERSHIP	T, DR, DW	Y	Y
OWNERSHIP STRENGTH	DW	N/A	Y
DEADLINE	T, DR, DW	Y	Y
LATENCY BUDGET	T, DR, DW	Y	Y
TRANSPORT PRIORITY	T, DW	N/A	Y
TIME BASED FILTER	DR	N/A	Y
RESOURCE LIMITS	T, DR, DW	N	N
USER_DATA	DP, DR, DW	N	Y
TOPIC_DATA	T	N	Y
GROUP_DATA	P, S	N	Y
ENTITY_FACTORY	DPF,DP,P,S	N	Y

WRITER_DATA_LIFECYCLE	DW	N/A	Y
READER_DATA_LIFECYCLE	DR	N/A	Y

2.5 本章小结

本章对基于 DDS 的飞机协同设计数据分发系统的需求进行了详尽的分析，并在此基础上充分阐述了基于 DDS 的飞机协同设计数据分发系统的总体架构设计和流程设计；详细论述了数据分发系统中相应模块的功能，展示在网络通信中数据分发系统的运作流程，为后文研究和实现工作做了铺垫。最后，介绍了关于数据分发服务 DDS 的三个关键技术：发布/订阅过程及其核心模块自动发现机制，并说明了服务质量 QoS，为本文研究创新提供依据。

第三章　基于 JSON 数据交换格式的飞机协同设计应用适配器

在飞机协同设计中，各厂商间使用的专业软件系统不尽相同，又因各自需求和生产环境等因素的不同，造成使用的操作系统、编程语言、数据库等方面存在差异。为了解决异构系统间数据格式不统一等问题，原始基于 XML 的应用适配器因 XML 数据格式在封装和解析过程中的复杂性，使得其运行过程中带来许多不必要的性能损耗。因此，优化现有适配器，提升系统整体性能降低损耗显得尤为迫切。本章基于 JSON 数据交换格式，用 JSON 替代 XML；通过封装，解析，存储与压缩实验数据表明，JSON 相较 XML，在飞机协同设计环境下对系统整体存取时间性能有显著提升。同时，在压缩率上也有一定的提升。因此可以作为复杂背景下优化系统性能的一个可行的解决方案。

3.1 简介

伴随着互联网技术的快速发展，网络传输数据的复杂性不断增大，针对各种特殊环境数据交换格式的改良也在不断的探索与研究中。在早期飞机协同设计环境下的系统平台中，常见的网络通信节点数据交互格式是采用 XML，XML 格式有两种主流的解析方式：DOM 和 SAX。

基于 DOM 的解决方案是将整个 XML 文件存储在一个对象的结点树中，该对象的结点可以与 XML 文件的层次结构和数据展开匹配。因此，当发生某种查询需求时，可以通过查询结点树来获得期望的数据信息。DOM 的解析方式会读入整个文档的内容，之后在内存中构建 DOM 树，并自动生成树中的所有结点。开发者可以通过外部的 DOM 接口按需求对树进行实际的操作。此种解决方案优势在于整个 XML 文档都在内存中，操作更加开放，支持各种增删改查等功能。然而缺陷在于可能会将许多当前不需要使用的结点也加载入内存，消耗了大量的内存空间，致使文档的解析效率降低。

基于 SAX 的解析是一种以事件驱动的轻量级解决方案。即读取时不需要将整个文档都载入进内存，只需要依据需求设定的阈值来定制读取的数据量。之后，在具体响应事件的回调代码中增加对 XML 文档数据的处理逻辑。这种方案优势在于对 XML 文档的解析时分段的，可以在任意位置开始与结束，无需将整个文档都载入到内存中，具有更强灵活性。此外，SAX 具有一边读取一边解析的良好特性，使其更适合应用于大型 XML 文档结构的数据解析。事实上，这正满足了应用程序在某些特定条件下终止解析的特殊需求。相较于 DOM 的解析方式，在灵活性与易用性上的优势都很明显。但其缺陷表现为存储性比较差，也即不会对数据做存储处理，因此导致无法对 XML 文档数据进行随机访问，在数据修改方面会出现当前待修改的目标对象为读入内存的问题。

现阶段，JSON(Javascript Object Notation) 数据格式在多年的发展中逐渐成为主流的网络数据传输交换格式[29]。JSON 因其简易的语法与高速的传输和封装解析效率引起了近年来

多方的研究与关注[30-36]。本章基于飞机协同设计的背景，对原有平台中采用 XML 数据交换格式的适配器进行改进。设计基于 JSON 数据交换格式的异构系统适配器，从而降低封装和解析数据传输格式的时间与网络带宽的消耗。

3.2 基于 XML 的飞机协同设计适配器模型

在飞机协同设计中，飞机部件制造厂商的各专业软件系统的编码语言可能不同，存储使用的数据库不同，操作系统不尽相同，因而存在各专业软件系统的数据格式不统一的问题。为了让各异构专业软件系统能够信息共享，需要基于飞机协同设计的应用适配器，将异构系统信息转化为飞机协同设计系统统一的数据格式，从而帮助各飞机部件厂商进行协同设计与开发。原始基于 XML 专业软件系统与总线的应用适配模型图如图 3.1。

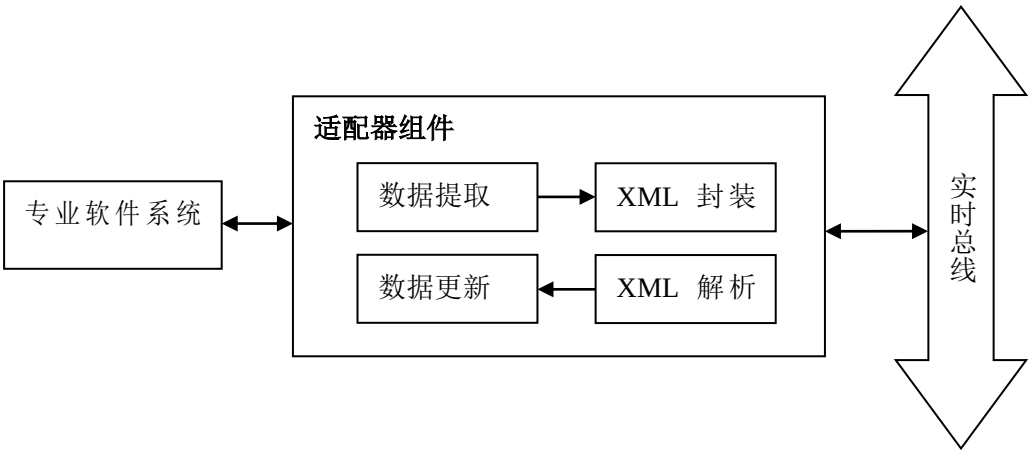

图 3. 1 基于 XML 的应用适配器模型图

数据提取模块：通过动态监控数据库中数据发生的变化，提取相关有效信息传递给封装器。

XML 封装器：接收数据提取模块传递的本地特征的数据，进而将其序列化成能够在总线上传输的统一数据交换格式，即 XML 格式。然后，传递到实时总线中。

XML 解析器：本地专业软件系统从总线中获得对自身有效的数据信息，XML 解析器对相关信息进行反序列化，转换成本地能够理解并存储的数据。

数据更新模块：将接收到的反序列化数据更新到本地专业软件系统的数据库中。

模型中应用了 XML 的封装与解析器。其中，可拓展标记语言(XML：eXtensible Markup Language)是用来传递数据的一种交互方式。XML 是由通用标准化标记语言衍生而来，因此没有所谓标签集合的概念和对应的语法规范，但遵循严谨的句法规则，也因此对于所有的 XML 数据文档采用正确的解析方式都规定文档本身是规范统一的结构，即句法规定文档开始与结束的标签必须一一对应，否则会解析错误。此外，XML 文档标签有严格的顺序要求且在语句结构上要符合相应的技术规范。

```
<?xml version="1.0" encoding="gb2312"?>
<DATAINFOLIST>
  <INFO>
    <FIELD>FSMR</FIELD>
    <NAME>桨毂中心纵向位置</NAME>
    <TYPE>4</TYPE>
    <PLATFORM>1</PLATFORM>
    <GROUP>旋翼</GROUP>
  </INFO>

  <INFO>
    <FIELD>BLMR</FIELD>
    <NAME>桨毂中心垂向位置</NAME>
    <TYPE>4</TYPE>
    <PLATFORM>1</PLATFORM>
    <GROUP>旋翼</GROUP>
  </INFO>
</DATAINFOLIST>
```

图 3.2 飞机协同设计下 XML 格式传输数据示意图

图 3.2 中是基于飞机协同设计的 XML 传输数据文档示例。首先语法上需要对 XML 的版本及编码进行定义。开始标签与结束标签必须一一对应。DATAINFOLIST 中包含有两个 INFO 子项，INFO 子项中又分别含有关于飞机部件参数的数据信息。

基于 XML 的应用适配器中，首先从专业软件系统的数据库中对相关变化数据进行数据提取，然后传递给 XML 封装器封装成 XML 格式数据，即图 3.2 中格式。封装完毕后传输到实时总线共享给其他专业软件系统。在总线中存在本地专业软件系统需要的数据时，从实时总线中获取 XML 数据，由 XML 解析器解析成本地对象并存入数据库中。

因 XML 在封装和解析的过程中复杂性较高，相应的在系统存取上的时间消耗较大。而 JSON 数据格式因其简易的语法与高速的封装解析和良好的存取效率受到了广泛关注。所以，本章综合飞机协同设计的背景，采用了基于 JSON 数据交换格式来解决异构系统数据格式不统一的问题。

3.3 基于 JSON 的飞机协同设计适配器模型

Javascript 对象符号(JSON：Javascript Object Notation) 是一种极为轻量的数据信息交互格式，其本身具有易读易写的特性。JSON 作为 Javascript 编程语言标准的一个子集，使用了完全独立于其他编程语言的原始格式，但在编程结构上也遵循类 C 的风格。

3.3.1 JSON 与 XML

JSON 作为飞机协同设计环境下一种新型的数据交换格式与原始 XML 相比有着多方面的优势，主要表现在易读性、数据体量与占用空间、是否利于在服务端创建数据、是否易于在客户端解析数据、扩展性、轻量级和重要级、编码与解析的量级等方面。

- 易读性

在可读性方面 JSON 与 XML 都具备良好的可读性，但是在实际的开发操作过程中 XML

文档结构因为有严格的标识符使其具有更优秀的可读性。而 JSON 为了发挥极致的简洁性，被设计的更加轻量，也因此结构过于简单，只是通过层次结构来区分模块。现阶段许多集成开发环境能够将 XML 文件的格式转化为层次更清晰的结构，感官上也更加易读。

- 数据体量和占用空间

很明显存储相同数据量的数据，JSON 要比 XML 使用更少的硬盘或者内存空间。JSON 官方也给出了几个有力的样例来证明这一特点。

- 是否利于在服务端创建数据

XML 格式文件一直都作为主流的数据传递交换格式且当前大部分编程语言都对其有良好的支持，并提供了设计良好的应用接口供第三方开发者调用。例如 JAVA 编程语言中有 JAXB、XMLBeans、Dom4j 和 JDOM 等多种方式来将数据写入 XML 中。而伴随中原始和移动互联网的快速发展，各种网络应用对网络实时性的要求愈发增高，JSON 数据交换格式也逐渐成为主流数据交换格式。JSON 官方就提供了多种编程语言的 API 接口，JAVA 语言上早已有 JSON-lib、GSON、fastJSON 等第三方解析包。

- 是否易于在客户端处理数据

在网络数据通信环境中，当数据信息传递到客户端时，内部返回的 JSON 格式数据相当简单，如果是浏览器，只需要调用 Javascript 中的 eval 接口函数就可以将 JSON 转换为相应对象，最后通过内部对象来访问相应的数据域值。另一方面，XML 格式的数据解析首先要生成 DOM 树，过程相对繁琐。

- 可拓展性

可扩展性有助于减少生产者与消费者之间的数据耦合。在 AJAX 应用里，客户端脚本应该合理地兼容不可知的数据扩展。XML 具有可拓展性，但存在一定的缺陷，当实施扩展时，客户端代码必须改动。即解析 XML 数据代码的改动。相对于 JSON，只是增加相关的属性值，在客户端中也只是通过对象访问属性值，而不会导致语法上的紊乱，致使程序崩溃。

- 轻量级和重量级

JSON 只提供了整体数据信息的解析方案，然而这种方案只适合数据量较小的情况会有极佳的效果。XML 是更加重量级提供大规模数据信息渐进解析的解决方案，一般适用于海量数据信息的解析处理。

- 编码与解析的量级

在编码上，两种数据交换格式都有自己的编码方式和成熟的编码工具。但无论是从编码速度还是响应事件上 JSON 的编码过程都要快于 XML。而由于 JSON 格式的简易性，其不用借助外在的工具也能完成编码工作。相比于 XML，有严格的标签限制，在实施的过程中要复杂的多。JSON 官方也给出了极为简洁的语法规则，使其具有高清晰低冗余的特性。所以，XML 一般适用于标记大型文档，JSON 一般适用于网络中数据信息的交互与处理。

在解析上，即便是简单的网络应用程序，在开发的过程中 XML 的解析也需要耗费大量

的资。无论是客户端利用 JS 解析，还是服务器端生成和处理都伴随着繁杂的代码逻辑，导致开发效率低下。而 JSON 则相反。

3.3.2 应用适配器模型

在保持原有应用适配器结构完整性的前提下，只针对 XML 封装器以及解析器进行替换。通过简单的替换，降低系统开销以及网络带宽的消耗。

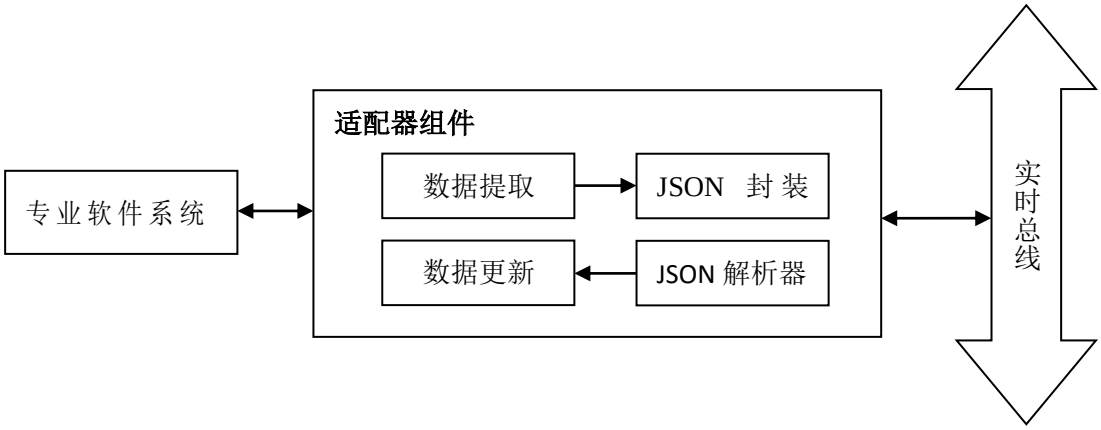

图 3.3 基于 JSON 的应用适配器模型图

JSON 封装器：尽管划分成 JSON 封装器，但 JSON 的封装可以只是简单的数据排列与拼接，相较 XML 复杂的语法格式要简便的多并且达到的功能效果相同。

JSON 解析器：通过本地专业软件系统采用的编程语言 API 接口将 JSON 数据转换为相应集合对象，然后逐层封装成本地数据模型对象。

```
{
  "DATAINFOLIST": {
    "INFO": [
      {
        "FIELD": "FSMR",
        "NAME": "桨毂中心纵向位置",
        "TYPE": "4",
        "PLATFORM": "1",
        "GROUP": "旋翼"
      },
      {
        "FIELD": "BLMR",
        "NAME": "桨毂中心垂向位置",
        "TYPE": "4",
        "PLATFORM": "1",
        "GROUP": "旋翼"
      }
    ]
  }
}
```

图 3.4 飞机协同设计下 JSON 格式传输数据示意图

图 3.4 中显示一条 JSON 数据，可以看做“键/值”（key/value）对来处理。DATAINFOLIST

中包含有一个 INFO 子项，INFO 子项中可以看做有两个数组，数组中分别含有关于飞机部件参数的数据信息。

基于 JSON 的应用适配器中，首先从专业软件系统的数据库中对相关变化数据进行数据提取，然后传递给 JSON 封装器封装成 JSON 格式数据，即图 3.4 中格式。封装完毕后传输到实时总线共享给其他专业软件系统。在总线中存在本地专业软件系统需要的数据时，从实时总线中获取 JSON 数据，由 JSON 解析器解析成本地对象并存入数据库中。尽管整体流程相似，但封装解析与存储的过程中 JSON 数据格式显著降低了系统存取性能的消耗。

3.4 实验结果与分析

3.4.1 封装解析与存储

本章以飞机协同设计过程中传递的数据作为实验的基础，分别对 JSON 和 XML 两种数据传输格式进行了封装解析并存入本地数据库的操作，通过实验结果的数据对比表现出两者的差异。在 500，5000，10000 条数据库数据上消耗时间对比（单位：ms）,考虑到封装解析和存储的过程中可能出现延迟的情况，对每个测试区间范围进行 10 次测试，测试结果再取平均值，得到封装解析与存储的时间消耗如表 3.1 所示。

表 3. 1 JSON 与 XML 封装解析与存储时间消耗对比

数据库类型	数据量（条）	JSON（ms）	XML（ms）
MYSQL	500	27294	31620
	5000	285088	329659
	10000	573470	652524
ORACLE	500	27866	31982
	5000	283335	328515
	10000	573517	668419
SQLSERVER	500	27196	31936
	5000	276932	320515
	10000	570736	659085

表 3.1 中在 MYSQL 环境下，500，5000, 10000 条数据量时，JSON 与 XML 差值分别为 4326ms，44571ms，79054ms。JSON 相较 XML 在时耗上分别约有 15.84%，15.63%，13.78%的提升。在 ORACLE 环境下，500，5000, 10000 条数据量时，JSON 与 XML 差值分别为 4116ms，45180ms，94902ms。JSON 相较 XML 在时耗上分别约有 14.77%，15.94%，16.54%的提升。在 SQLSERVER 环境下，500，5000, 10000 条数据量时，JSON 与 XML 差值分别为 4740ms，43583ms，88349ms。JSON 相较 XML 在时耗上分别约有 17.42%，15.73%，15.47%的提升。

为了表现出随数据量的增加而带来的时间上的损耗差异，本章通过在样本范围内缩短选

取节点的区间，并通过图 3.5，3.6，3.7 表现。图中实线与虚线分别表示封装解析与存储 XML 与 JSON 在 500 到 10000 条数据上的时间消耗。可以看出随着数据量的增大，JSON 相较 XML 时耗优势明显。

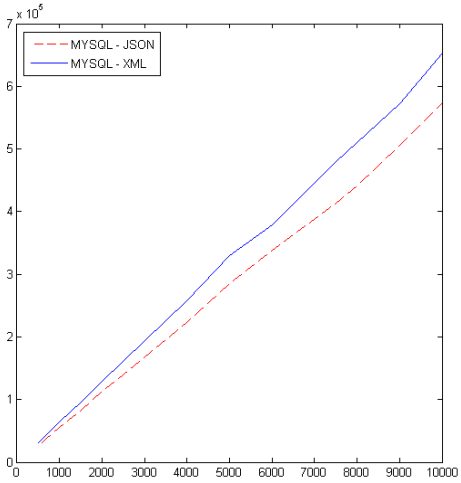

图 3. 5 MYSQL 环境下 XML 与 JSON 封装解析与存储时间损耗

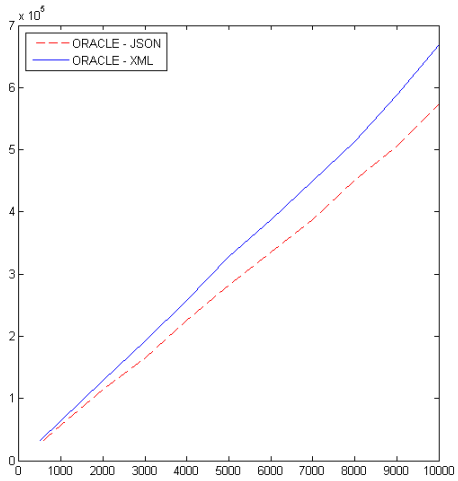

图 3. 6 ORACLE 环境下 XML 与 JSON 封装解析与存储时间损耗

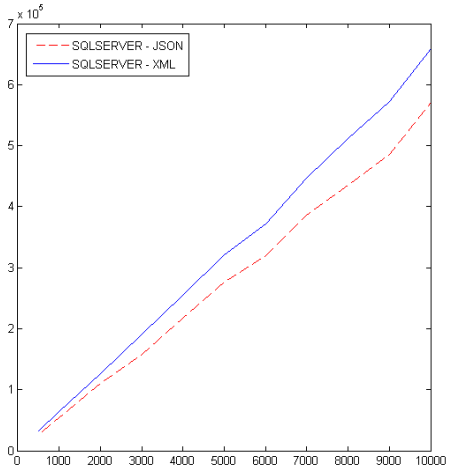

图 3. 7 SQLSERVER 环境下 XML 与 JSON 封装解析与存储时间损耗

3.4.2 数据压缩实验

对 JSON 与 XML 在空间上的对比，如表 3.2。

表 3. 2 JSON 与 XML 数据压缩对比

数据量（条）	JSON（bits）	占用空间（bits）	XML（bits）	压缩率（%）
500	4,041	4,096/8,192	4,144	97.52%
5000	6,394	8,192	6,573	97.28%
10000	8,629	12,288	8,813	97.92%

表 3.2 中在 500，5000,10000 条数据库数据下，JSON 文件大小分别为 4041bits，6394bits，8629bits，XML 文件大小分别为 4144bits,，6573bits，8813bits。两者差值分别为 103bits，179bits，184bits。通过压缩率可以看出 JSON 相较 XML 有一定的压缩提升。在占用空间方面，对于 500 条数据，JSON 比 XML 所需字符数上优势明显，在其余数据量上占用字符数相同。

3.5 本章小结

本章提出了基于 JSON 数据交换格式的异构系统应用适配器，因 JSON 格式较 XML 有许多先天的优势，所以对应用适配器进行改进和优化。实验表明 JSON 与 XML 在封装解析、存储、压缩方面都有相应的优势。因为飞机协同设计自身的复杂性，需要通过各方面的努力降低系统能耗与网络带宽的消耗，本章在相关方面做的研究是成为众多解决方案之一，对后续优化工作的开展提供了帮助。

第四章　基于 CBF 和 QoS 兼容的自动发现算法

在 DDS 中，需要根据结点之间的主题信息是否相同以及 QoS 策略是否兼容来最终确定主题是否匹配。而基于布隆过滤器的自动发现算法在各结点之间传递数据时效率较低。而且，不支持 QoS 兼容性判定，导致最终的主题匹配率很低。当前采用服务力向量的算法只是对 QoS 进行了粗判，主题匹配率仍然不高。本章进行了两个方面的改进。第一，采用压缩的布隆过滤器(Compressed Bloom Filter)替代原有过滤器，有效地降低了网络中传输的数据量，减缓了内存的消耗；运用了更少的 hash 函数，降低了计算的复杂度；同时，可以获得更小的虚警率(false positives)，提高了主题信息的匹配率。第二，添加 QoS 策略兼容性判定的设计与实现，从而极大地提高了主题的匹配率。

4.1 自动发现算法

自动发现算法作为 DDS 的核心技术。DDS 主题匹配的主要判别依据是结点之间的主题信息是否相同及 QoS 策略是否相互兼容。当前国内外众多研究学者仍然在对 DDS 规范与 DDS 适用的多领域进行广泛的研究探索，其中尤其面向 DDS 的自动发现算法开展了深入的探讨。

文献[21]中指出 DDS 标准中简单自动发现协议要求使用者预先静态配置网络中的各节点信息，不能够很好的支持大规模动态网络。所以，为了适用于大规模动态网络，文章提出了一种自适应动态网络服务发现机制。但是存在不能预先判定主题信息是否相同等缺陷。进而文献[22]通过实验证明 SDP 不利于小规模环境且不易于系统拓展维护，文章基于 BF 的匹配方式提出了一种自动发现算法，简称 SDPBloom，旨在更加高效的解决标准 DDS 自动发现算法存在的高内存消耗以及高网络负载等诸多缺陷。但该方案主要不足在于不能预先判别 QoS 策略是否相互兼容，所以文献[25]中提出的基于服务力向量的自动发现算法，在一定程度上处理了因 QoS 不兼容带来的问题，初步提升了主题的匹配率。本章基于上述方案在传输网络数据时存在过高的带宽消耗与 QoS 策略粗判的不足，构建了 Compressed Bloom Filter, 旨在进一步减少网络通信数据传输量并减少 hash 函数个数降低计算的复杂度。另一方面，设计并实现了基于位运算的 QoS 判别方法，提升主题的匹配率。

4.2 基于 Bloom Filter 的自动发现算法

2009 年间 RTPS 标准 2.1 的发布版本[37]中指出数据分发服务的自动发现算法在实现的过程中至少要支持 SDP(Simple Discovery Protocol)协议，而 Standard 版本的数据分发服务的自动发现算法就是遵循了此种协议实现的算法。在标准中详细描述了发现的过程主要分为两个阶段：简单参与者与简单端点发现阶段。DDS 中的自动发现算法内部使用了数据读取者和数据写入者。只有当数据读取者与写入者的主题相关数据信息和 QoS 策略相互兼容时，网络通信中的对端节点才进行实际主题数据信息的传输与交互。

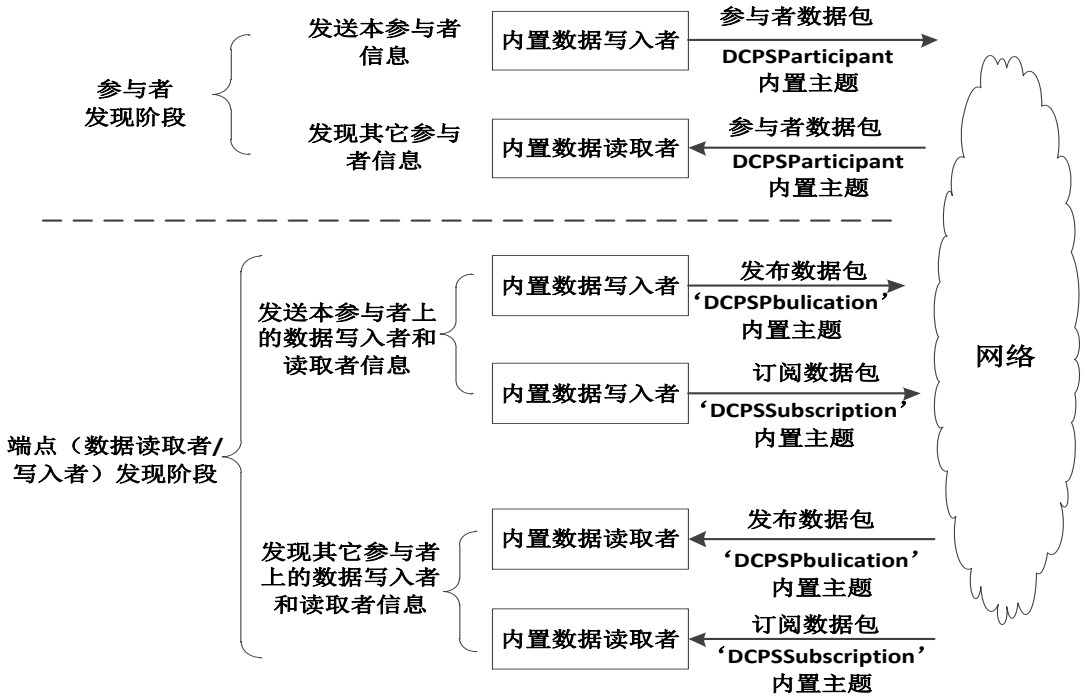

图 4.1 基于 SDP 自动发现算法

（1）参与者发现阶段。第一阶段开始，使用一个内置的数据写入者发送本地 Domain 中所有相关参与者信息的数据包给远程 Domain 中的对应参与者。使用一个内置读取者读取相关远程 Domain 参与者信息的数据包。包中主要包含参与者的全局唯一标识符 GUID(Globally Unique Identifier)、对端使用的 QoS、地址和端口号等。这些数据信息通过高效的组播方式周期性的发送。参与者通过定期发送对应信息的数据包来延长 Domain 参与者的生命周期。当参与者的 QoS 发生变动时，这些状态信息也同样需要发送。

（2）端点发现阶段。应用程序中数据读取者和写入者的相关数据信息，其中主要包含全局唯一标识符 GUID、对端使用的 QoS 等。整个体系中的消息发送方和订阅方之间进行数据交互。这个阶段会使用可靠的传输方式。当系统中检测到数据写入者或者数据读取者时，由传递消息内置的判别机制决定此消息是否与本地应用程序匹配，当且仅当双方的主题相关信息和 QoS 各选项相互兼容时，才判别为匹配成功，双方才开始实际主题数据信息的交互与使用。

上述 SDP 协议没有预先匹配主题信息的匹配机制，因此传输了大量不必要的网络数据，消耗了额外的网络带宽。针对此种缺陷引入的 SDPBloom 用于预先的主题信息匹配，待通信双方就主题达成共识后再进行实际主题相关的数据传递。从而有效降低网络带宽消耗。BF 的具体匹配过程如图 4.2 所示。

添加：
H1(a) | H2(a) | H3(a) = 00100101
H1(b) | H2(b) | H3(b) = 10010100
Bloom Filter: 10110101
查询：
H1(m) | H2(m) | H3(m) = 00100110
H1(n) | H2(n) | H3(n) = 10010001

图 4. 2 Bloom filter 匹配过程

图 4.2 中 Bloom Filter 是一个长度为 8 的位字符串。元素 a，b 分别进行 hash 之后得到相应位字符串。当添加到过滤器中时，第三位发生冲突。查询时，元素 m 因为第 2 位与 BF 对应位置不匹配，所以不在 BF 集合中。而元素 n 每一位都与 BF 相互匹配，因此判定元素 n 在集合 BF 中。但在上述操作过程中元素 n 并没有被添加进 BF 集合中，由此产生虚警率问题，记为 f。由文献[38]得到式 1（其中令 $p = e^{-kn/m}$ ），k 表示 hash 函数的个数，m 表示数组大小，即 Bloom Filter 大小。n 表示元素的个数。当满足式 2 时，对于 f 取最优解。

$$f = (1 - e^{-kn/m})^k = (1 - p)^k \quad (1)$$

$$k = (\ln 2)(m/n) \quad (2)$$

SDPBloom 没有考虑 QoS 策略是否兼容，当主题数量增大，主题数据量增加时，仍然会消耗大量的网络带宽。文献[25]在参与者阶段将端点 QoS 策略兼容性等问题考虑在内，使得各结点和网络减少大量 QoS 不兼容的端点信息传递，从而进一步降低了内存消耗和网络负载。作者主要以服务力 SA 在参与者发现阶段对端点 QoS 的兼容性进行粗判。

在 RxO 模式中，网络中通信对端节点需要首先判定 QoS 兼容性策略，当且仅当 offered 域值大于等于 requested 域值时，对端节点才能建立交互通道。文献基于 requested 域值与 offered 域值的区间长度设计了一种服务力，随着区间长度的增大，服务力也越大。依据 QoS 表中满足 RxO 模式的 8 个 QoS 兼容性策略的服务力集合来构建服务力向量的规则。改进后，当本地系统接收到网络中对端节点的数据信息包时，提取包末尾的字段与本地数据信息包对应部分进行逻辑与操作，结果为真表示对端节点相互兼容，反之不兼容。综上所述，当通信双方兼容不成功时，就没有必要建立通信传递数据信息，因此，降低了双方内存资源与网络带宽的消耗。因为文章只是对 QoS 进行粗略的过滤与筛选，所以仍然存在 QoS 策略匹配不精确，消耗额外网络带宽等问题。

4.3 基于 QoS 兼容策略的 Compressed Bloom Filter 自动发现算法

针对 BF 传输网络数据流量仍然会有过高的内存和网络带宽的消耗，此外，基于 SAV 的自动发现算法只是对 QoS 策略进行粗判，无法达到最优的主题匹配率。所以，本章采用 CBF 进一步减少网络传输数据流量降低内存消耗并且使用更少的 hash 函数降低虚警率，在 QoS 方面，构建通信双方遵循的传输协议表，表中按位唯一确定每一个 QoS 的子选项，当

CBF 携带其传递到请求方时逐一进行匹配，从而有效提升主题匹配率，降低主题相关的网络数据流量。

4.3.1 基于 Compressed Bloom Filter 的自动发现算法

在文献[38]中，CBF 有 4 个基础性能指标：k 表示 hash 函数的个数，f 表示虚警率，m 表示未压缩的过滤器大小，z 表示传输大小。建立关于 CBF 的最优问题求解如下：令 z 作为期望的压缩后数据大小。使得在位数组中的所有位是 0 的概率为 p；每一位相互独立。同时，为了方便数学上的估算，假设有最优的压缩算法。同时，假设 m bit 过滤器能够被压缩到 mH(p) bits，引入式 3，这个二进制熵函数用于建立 Bloom Filter 大小 m 和传输大小 z 的映射关系(即式 4)。这里，我们使用接近最优的算术编码[39-40]压缩算法。

$$H(p) = -p\log_2 p - (1-p)\log_2(1-p) \quad (3)$$

$$m = z / H(p) \quad (4)$$

上述优化问题分析如下：给定 n 和 z，选择 m 和 k 使得 f 最小化，其中 z 服从 $mH(p) \le z$。原始最优 BF 的方案是选择 m = z 同时 $k = (\ln 2)(m/n)$ 以便 p = 1/2；这种方式能够保证 $f \le (0.6185)^{z/n}$。

但是，对于压缩后的 BF 有更好的解决方案。理论上，一旦允许压缩，上述 k 的选择可能是最差的。所以，根据式 4 和式 5 重写 f 关于 p 的函数，得到式 6，其服从式 4。

$$p = e^{-kn/m} \quad (5)$$

$$(1-p)^{(-\ln p)(m/n)} \quad (6)$$

由式 4 与式 6 得到式 7。对式 7 熵展开并求最值后，得出结论：对于未压缩 BF 选取最优 hash 函数个数时，所得到的最小虚警率与压缩 BF 的最大虚警率相同。上述结论为 CBF 在有效降低虚警率上提供了有力的数学推导证明。

$$f = (1-p)^{-(z\ln p/nH(p))} \quad (7)$$

4.3.2 基于 QoS 兼容策略的设计与实现

针对文章[25]对 QoS 兼容策略粗判的不足，本章提出了更加彻底的 QoS 改进方案，从根本上提高了发现阶段 QoS 的匹配，从而极大地提升了主题的匹配率，降低网络带宽传输的数据量。

因为在 23 种 QoS 中，只有 8 种需要满足 Request vs Offered 模式，即双方可以分别对 QoS 进行设定，因为在通信时需要进行兼容性判定，所以兼容值需要明确定义不能为空。本章只构建基于这 8 种 QoS 的通信协议，规则如表 4.1：

表 4. 1 QoS 协议表

QoS 策略	RxO 模式	适用规则	通信双方规则
RELIABILITY	Y	RELIABLE > BEST_EFFORT	提供方 >= 请求方
DEADLINE	Y	INFINITY > NOT_INFINITY	提供方 <= 请求方
DURABILITY	Y	PERSISTENT > TRANSIENT > TRANSIENT_LOCAL > VOLATILE	提供方 >= 请求方
OWNERSHIP	Y	SHARED || EXCLUSIVE	提供方 = 请求方
PRESENTATION	Y	GROUP > TOPIC > INSTANCE	提供方 < 请求方
LIVELINESS	Y	MANUAL_BY_TOPIC > MANUAL_BY_PARTICIPANT > AUTOMATIC	提供方 >= 请求方
DESTINATION_ORDER	Y	BY_SOURCE_TIMESTAMP > BY_RECEPTION_TIMESTAMP	提供方 >= 请求方
LATENCY_BUDGET	Y	ZERO > NOT_ZERO	提供方 <= 请求方

表 4.1 中 8 个 QoS 策略都需要满足 RxO 模式。此外，对于其子选项需要严格满足“适用规则”一列的描述。例如，当请求方要求 RELIABILITY 的 RELIABLE 规则时，提供方必须提供 RELIABLE。否则匹配不成功。相反，当请求方为 BEST_EFFORT 时，提供方可以是 RELIABLE 和 BEST_EFFORT 任意一项。

当确定通信双方运行规则时，依照 QoS 的通信协议表进行编码，范例如图 4.3 所示。

```
// RELIABILITY
DDSQOS_RELIABILITY_BEST_EFFORT {
    public int getSequence() {
        return 1 << 0;
    }
},
DDSQOS_RELIABILITY_RELIABLE {
    public int getSequence() {
        return 1 << 1;
    }
},
```

图 4. 3 QoS 的通信协议表编码范例

针对 8 个 QoS 构建通信对端协议表。每个子选项占一位 1^0、1^1…1^20 以此类推，共 20 位，需要 3 字节。通信前，通信各方都维护一份协议规则以便之后的兼容性检查和判断。开始通信，CBF 携带 QoS 传递到请求方，首先通过 CBF 判断当前接收消息是否有匹配的主题名，匹配成功后进行 QoS 甄别。在经过表 4.1 和图 4.3 严格的 QoS 判别之后，确认当前主题完全匹配，开始主题数据信息的传递与交互。

4.4 实验

4.4.1 Compressed Bloom Filter 改进

首先实验数据采用基于飞机协同设计平台各厂商之间的主题数据，其分别按每个元素占 8 bits 和每个元素占 16bits 进行数据压缩实验（因为 SAV 自动发现算法是基于 BF 实现的，所以本实验目标对象为 BF 和 CBF），实验结果如表 4.2 和表 4.3 所示

表 4. 2 BF 中每个元素占 8 bits 的实际大小与传输大小对比

元素个数 n(个)	100	500	1000
BF 大小 m(bits)	800	4000	8000
实际大小(bits)	832	4032	8000
传输大小 z(bits)	1105	1304	1392

表 4. 3 BF 中每个元素占 16 bits 的实际大小与传输大小对比

元素个数 n(个)	100	500	1000
BF 大小 m(bits)	1600	8000	16000
实际大小(bits)	1600	8000	16000
传输大小 z(bits)	1199	1562	1720

压缩部分采用上文提及的算术编码[39-40]。表 4.2 中每个元素占 8bits，主题个数分别采用 100，500 和 1000，对应的 BF 大小为 800bits，4000bits 和 8000bits。存储主题过程中对于较少的主题数有时会需要额外的空间，如对于 100 个主题，实际需要 832bits。当采用算术编码后得到传输大小 z，对应大小分别为 1105bits，1304bits 和 1392bits。数据表明，随着主题个数的增加，采用 CBF 能够极大地缩小存储空间，减少网络数据流量。表 4.3 提供了当每个元素占 16 bits 时相似的数据表现。

依据 2.1 节的证明，CBF 可以减少网络传输数据量，同时应用更少的 hash 函数来获得更小的虚警率。 本章应用飞机协同设计平台下的主题数据，旨在降低 hash 计算的复杂度并获得优化的虚警率。实验结果如表 4.4，4.5，4.6 所示。

表 4. 4 压缩前后 BF 中每个元素占 16 bits 的 hash 函数个数与虚警率对比

每个元素占用位(bits)	m / n	16	50
每个元素传输位(bits)	z / n	16	16.3
hash 函数个数(个)	k	11	2
虚警率	f	0.000459	1.436E-4

表 4. 5 压缩前后 BF 中每个元素占 15bits 的 hash 函数个数与虚警率对比

每个元素占用位(bits)	m / n	15	40
每个元素传输位(bits)	z / n	15	15.06
hash 函数个数(个)	k	10	2
虚警率	f	0.000743	0.000306

表 4. 6 压缩前后 BF 中每个元素占 20 bits 的 hash 函数个数与虚警率对比

每个元素占用位(bits)	m / n	20	80
每个元素传输位(bits)	z / n	20	20.02
hash 函数个数(个)	k	14	2
虚警率	f	6.713E-5	1.523E-5

表 4.4 中对于每个元素占 50bits，压缩后每个元素传输位只需要与原始 16bits 相同数据量。此外，相比原始 BF 需要用 11 个 hash 函数才能达到最优的虚警率，CBF 只需要 2 个 hash 函数来获得更低的虚警率。表 4.5 和表 4.6 在不同的数据量上做了相似的类比实验以获得类似结论。

4.4.2 基于 QoS 兼容策略的主题匹配

随机 100 个主题并随机分配其 8 个 QoS，请求方建立对应 100 个厂商，分别订阅 100 个主题，每种过滤器分别进行 10 次试验取平均值。当厂商分别接收到广播的 Compressed Bloom Filter 时，首先进行主题名匹配，匹配通过，则进入 QoS 匹配，在完成一系列 QoS 匹配后表示匹配成功，否则匹配失败。这里，依据规则可以省略一部分判断。例如，DURABILITY 中，当请求方为 VOLATILE 时，因为提供方>=请求方，所以请求方可以是 VOLATILE、TRANSIENT_LOCAL、TRANSIENT、PERSISTENT 任意选项之一，所以此时不用进行额外的判断，默认为通过此项筛选。

表 4. 7 QoS 兼容的主题个数

主题个数	QoS 兼容的主题个数		
	Bloom Filter	基于 SAV 的 BF	基于 QoS 的 CBF
100	1.53	82.72	100
250	3.34	198.07	250
500	2.66	413.80	500

表 4.7 中因为原始的 BF 不具有判别 QoS 的能力，所以其只是判断主题名是否匹配，如果匹配则通过。之后开始传递主题实际相关的数据，但传递给请求方之后还是要进行所有 QoS 的匹配筛选。由于 QoS 相对较多且限制复杂，所以造成原始 BF 匹配率低下。同时，这也证明了预先匹配 QoS 的重要性。尽管基于 SAV 的 BF 在 QoS 的改进上取得了显著的效果，但仍然存在一定的匹配错误率。而本章提出的基于 QoS 的 CBF 则能够完全匹配各种主题。

4.5 本章小结

针对 Bloom Filter 的数据冗余和 SAV 自动发现算法 QoS 判别不精确等问题，本章基于 Compressed Bloom Filter 进行改进。不仅有效的降低了通信双方需要传递的数据量，而且可以使用更少量的 hash 函数获取更优的虚警率。同时，提出了基于 QoS 兼容策略的设计与实现方案，使得总体的主题匹配率显著提高。实验也进一步表明，无论是针对基于 BF 还是基于 SAV 的自动发现算法，本章提出的基于 QoS 的 CBF 都表现出显著的优势。

第五章　基于 DDS 的飞机协同设计数据分发系统的实现

本章对基于 DDS 的飞机协同设计数据分发系统的实现进行了介绍，首先介绍系统实现的开发环境，其中包括软硬件配置。其次，对网络通信中异构系统数据格式适配器和数据分发服务 DDS 的发布/订阅的核心模块自动发现两个主要功能模块进行了详细的介绍。最后，对基于 DDS 的飞机协同设计数据分发系统进行整合测试，主要包括功能测试和性能测试两个方面。

5.1 开发环境简介

基于 DDS 的飞机协同设计数据分发系统在开发中选用开源项目 OpenDDS 的相关组件来实现系统的核心功能。开发环境的基础配置如下：

1）开发环境硬件配置

操作系统：Windows 7 专业版（64bit）

处理器:　Interl(R)　Core(TM) i3-4160 CPU @ 3.60GHz

内存：　4.00GB

硬盘：　500GB HDD

2）开发环境软件配置

集成开发环境：Visual Studio 2015

ACE 版本：　ACE+TAO+CIAO-6.2.7

DDS 版本：　OpenDDS-3.5.1

因为 OpenDDS 是在 ACE/TAO 开发环境下实现的，所以必须要预先配置好 ACE/TAO，过程如下：

- 配置 ACE/TAO

TAO 是一种跨平台多终端的应用程序，使得开发人员从内部复杂的实现细节中解放出来，更加专注于业务功能的本身。例如本文只需要集中发布/订阅的功能扩展。对 TAO 的操作分为以下几个部分：

首先，需要配置基于 perl 语言的运行环境。同时，需要进入 ACE 的下载页面，获取 TAO 的下载链接并下载源码。DOC 提供 standard 和压缩两种下载方式。本文使用 ACE+TAO+CIAO-6.2.7 的压缩版本，下载完毕后解压到本地。

之后，需要设置 TAO 的环境变量，设置过程如图 5.1：

```
ACE_ROOT = C:\Users\cash\Desktop\ACE_wrappers\
ACE_ROOT = C:\Users\cash\Desktop\ACE_wrappers\
PATH = %ACE_ROOT%/bin;%ACE_ROOT%/lib
```

图 5. 1 设置环境变量

在编译选项的配置上，需要自定义 config.h 文件替换 ACE 根目录文件夹下 ace 中的原同名文件，其中配置内容与注释如图 5.2：

```
#include "ace/config-win32.h"

#ifndef ACE_CONFIG_H                          // 定义宏
#define ACE_CONFIG_H
#define ACE_DISABLE_WIN32_ERROR_WINDOWS       // 禁用错误窗口
#define ACE_DISABLE_WIN32_INCREASE_PRIORITY   // 禁用增加优先级
#define ACE_HAS_MFC 1                         // 应用`MFC`库
#include "ace/config-win32.h"
#endif
```

图 5. 2 config.h 文件配置内容及注释

最后，分别编译 ACE 和 TAO 工程，因为 ACE 本身有 800 多个项目，TAO_IDL 和 TAO_VC 分别有 11 和 2200 个项目组成，所以本文使用命令行编译来提升编译效率。打开 Visual Studio 2008 的 cmd 命令控制框，输入如图 5.3 命令，相关注释也如图所示。

```
vcbuild %ACE_ROOT%/ace/ace_vc9.sln "Debug | Win32"
        // 参数可以选择"Debug|Win32"或者"Release|Win32"
          // 根据编译项目时用`Debug`或者`Release`模式，会生成不同的dll或lib文件
```

图 5. 3 编译 ACE 命令及注释

编译 TAO 时，分别依次编译 TAO_IDL_vc9.sln 和 tao_vc9.sln。在编译生成好 ACE/TAO 的基础上，只需要配置 OpenDDS 的环境变量 DDS_ROOT= D:\openDDS\DDS、PATH=%DDS_ROOT %\lib 和 % DDS_ROOT %\bin 。编译命令为 msbuild DDS_vc10.sln/p:Configuration=Debug > build_dds_debug.log。

当编译完成时，OpenDDS 的基础开发环境就已经建立完成了，此时可以基于 OpenDDS 进行二次开发。本文主要实现其发布/订阅的过程，并基于其核心组件自动发现做改进。相应发布/订阅实现过程如下。

5.2 发布订阅过程简介

数据分发服务是一种以发布/订阅作为主要通信方式的网络服务模型结构，在网络通信交互环境下，系统的各个节点都可以作为发布或订阅方且可以同时兼具两者。当开始网络传输时，主要分为发布过程和订阅过程。

- 发布过程

当结点 *i* 需要发布主题相关消息时，在简单参与者发现阶段，域参与者会首先创建对应主题数据信息的发布者 P_i，再由 P_i 来创建数据写入者 DW_i。DW_i 将会发布主题对应消息 T_i 并写入主题对应兼容性的 QoS，最后通过广播的方式传递给系统中其他结点，发布成功后，写入者进入阻塞状态。系统中结点检测到有满足本结点相关订阅主题信息时，会唤醒数据写入者 DW_i，将实际数据发送到远程订阅者一方，同时，根据主题信息的服务质量将主题相关数据写入到主题发送缓冲区中。DDS 标准规范中，制定了发布过程的具体操作步骤，如图 5.4 所示。

<1> 调用 create_participant 接口函数，用于创建一个域的参与者。

<2> 域参与者调用 create_publisher 接口函数，用于创建一个主题信息发布者。

<3> 域参与者调用 register_type 接口函数，用于注册一个将要发布的数据类型。

<4> 域参与者根据步骤 3 创建的主题数据类型，调用 create_topic 接口函数用于创建主题。

<5> 主题发布者调用 create_datawriter 接口函数，用于为相应主题创建一个数据写入者。

<6> 为了内部编码的数据类型的安全，函数内部使用指定接口转换数据类型。如使用类型安全的 narrow 接口函数，用于强制转换数据写入者为指定数据类型。

<7> 调用相应类型的 write 接口函数，用于写入主题数据信息。

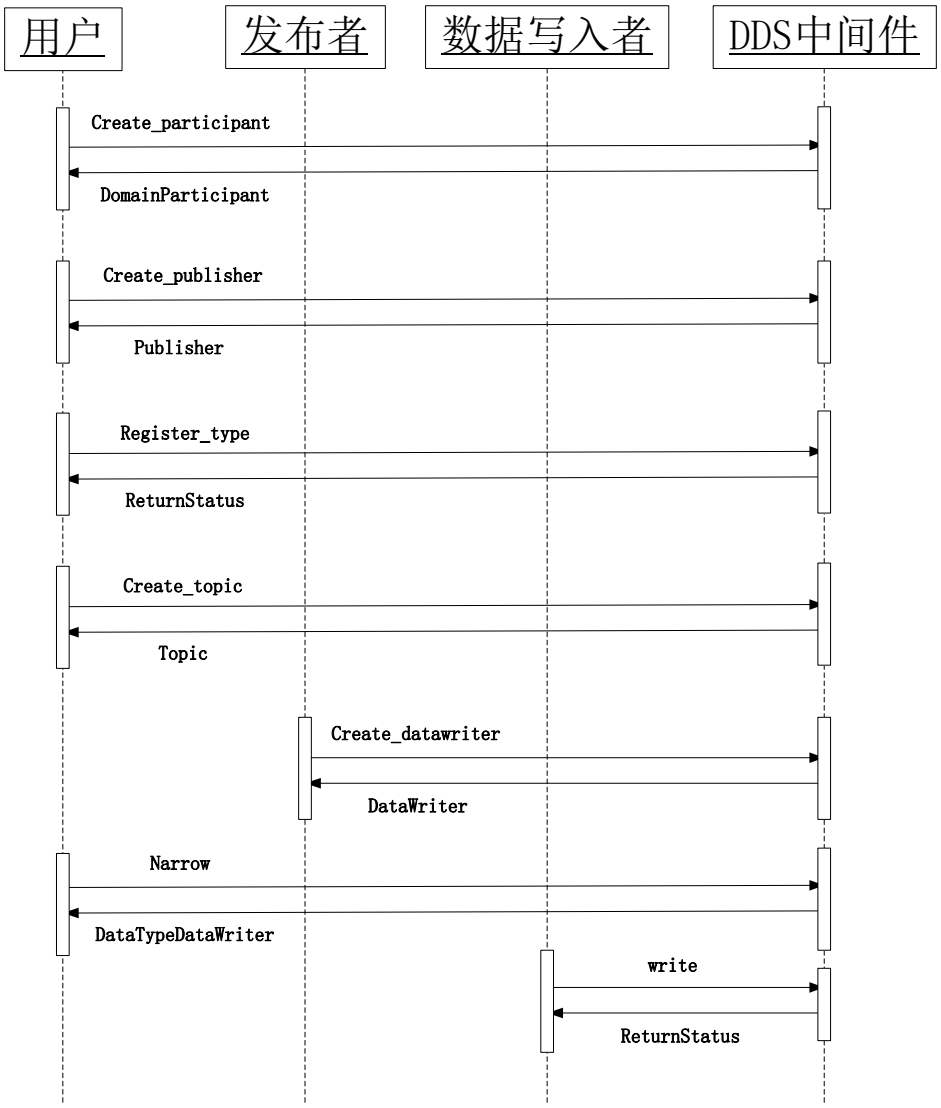

图 5.4 发布过程时序图

- 订阅过程

当结点 j 需要订阅主题 T_j 的相关数据信息时，域参与者会创建对应主题信息的订阅者 S_j，其创建数据读取者 DR_j。之后，DR_j 会在整个系统上发现当前活动结点中的发布者 P_i，然后系统根据内部相关筛选策略选取数据通信发送方，发送方将主题数据相关的描述信息及 QoS 兼容性策略加入到发送数据包中，发送数据包后 DR_j 进入阻塞状态等待数据包到达被唤醒。DDS 标准规范中，规定了订阅过程的相关步骤，如图 5.5 所示：

<1> 调用 create_participant 接口函数，用于创建一个域的参与者。

<2> 域参与者调用 create_subscriber 接口函数，用于创建一个主题信息订阅者。

<3> 域参与者调用 register_type 接口函数，用于注册一个将要订阅的数据类型。

<4> 域参与者根据步骤 3 创建的主题数据类型，调用 create_topic 接口函数用于创建主题。

<5> 主题订阅者调用 create_datareader 接口函数，用于为对应主题创建一个数据读取者。

<6> 为了内部编码的数据类型的安全，函数内部使用指定接口转换数据类型。如使用类型安全的 narrow 接口函数，用于强制转换数据读取者为指定数据类型。

<7> 调用相应类型的 reader 接口函数，用于读取主题数据信息。

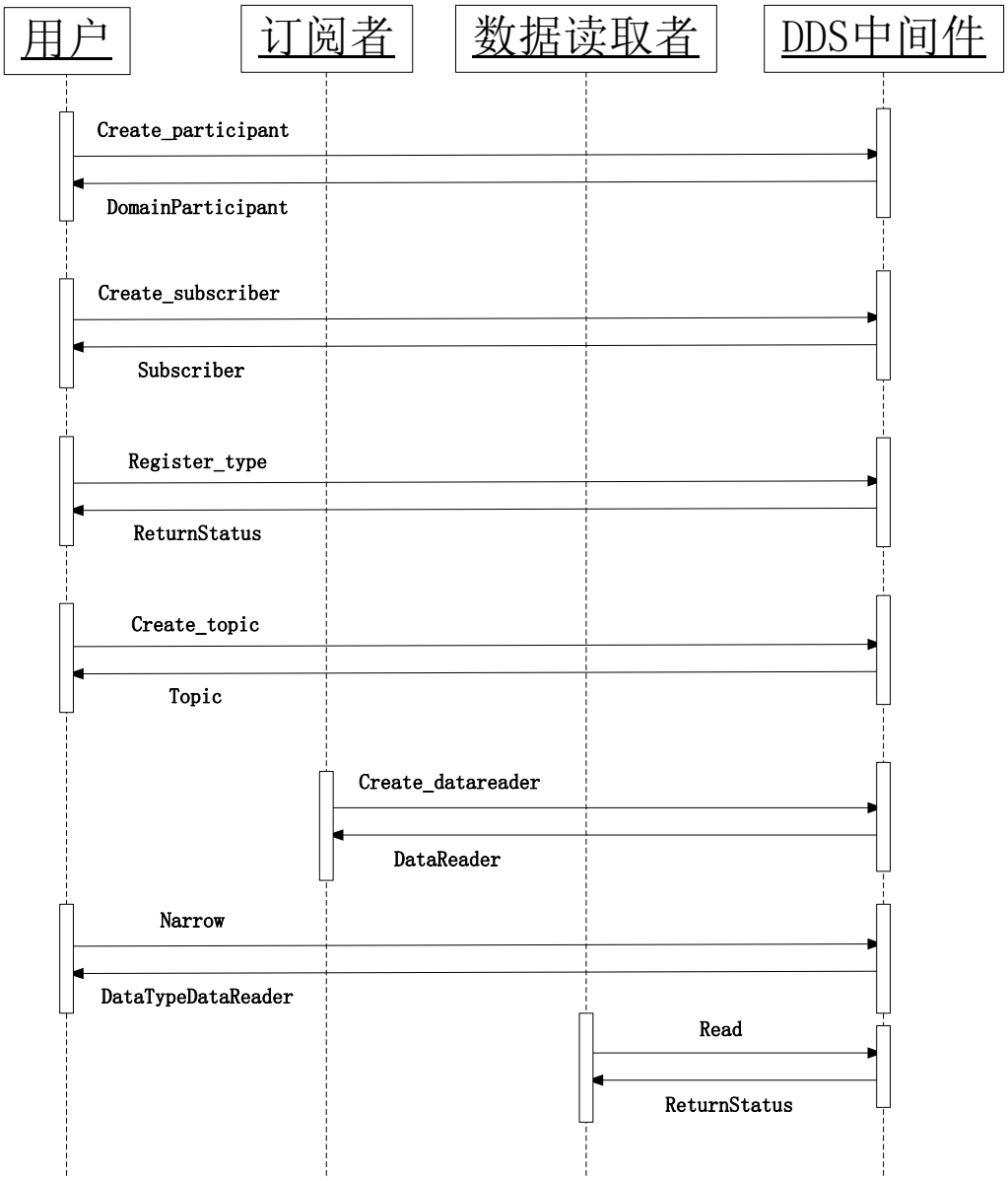

图 5.5 订阅过程时序图

5.3 发布过程实现

根据 OpenDDS 的开发者文档一个简单的消息发布过程需要分为发布过程和订阅过程，其中发布过程又包含初始化参与者对象、注册发布数据类型和创建自定义主题数据信息、创建对应主题发布者、创建相关数据写入者和等待对应主题订阅者以及发布相关数据五个阶段，订阅过程又包含初始化参与者对象、注册发布数据类型和创建对应主题、创建对应主题订阅者、创建相关数据读取者和监听者、数据读取者监听实现五个阶段，如图 5.6 所示。

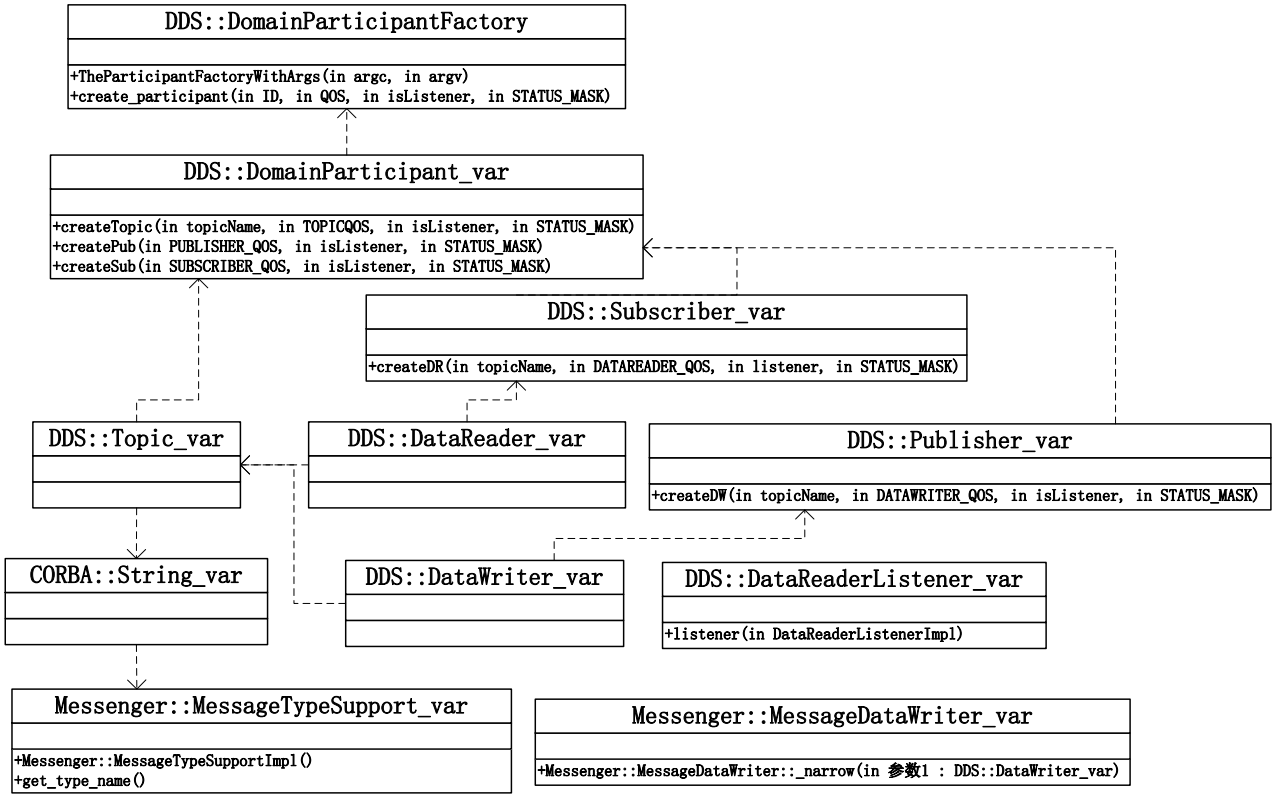

图 5.6 发布/订阅初始化模块类图

<1> 初始化参与者

首先在主函数中初始化一个 OpenDDS 的参与者。图 5.6 中宏 TheParticipantFactoryWithArgs 已经被预先定义，并且通过命令行参数来初始化一个域参与者工厂。这些命令行参数用于初始化 ORB，这种 ORB 是 OpenDDS 服务使用的，同时也是这种服务本身使用的 ORB。以上允许我们在命令行上运用 ORB_init()选项和 OpenDDS -DCPS*形式的配置选项。

create_participant()函数操作运用域参与者工厂来注册这个过程，通过指定 ID 来在域中创建一个参与者。这个参与者使用默认的 QoS 策略和无监听者。在中间件和应用程序通信中，运用 OpenDDS 默认的状态 mask 确保所有相关通信的改变（例如，可用数据和生命周期的缺失）。这里的域参与者对象引用会在之后用于注册消息数据类型。

<2> 注册相关数据类型和创建对应主题对象

首先，创建一个 MessageTypeSupportImpl 对象，之后调用 register_type()函数用类型名注册类型。其中注册类型为空字符串类型名，因为 MessageTypeSupport 接口库被标识作为类型名。同时，也可以运用一个特定的类型名，如“Message”。下一步，从类型支持对象获得注册类型名并通过类型名用参与者调用 create_topic()函数创建主题。

<3> 创建发布者

接下来用默认的发布者 QoS 创建发布者。如发布/订阅初始化模块类图 5.6 中通过 DDS::DomainParticipant_var 对象创建发布者。

<4> 创建数据写入者和等待订阅者

创建好发布者之后，通过发布者创建数据写入者。依据主题对象的引用、默认 QoS 兼容性策略和一个空的监听者对象引用来创建数据写入者。接着调用 narrow 函数将数据写入者引用转换为一个 MessageDataWriter 对象引用，因此我们能够运用特定类型进行发布操作。

图 5.6 中运用状态和等待设置，因此发布者需要等待订阅者来建立连接并完全初始化。如果代码中出现失败的等待订阅者，可能会导致在与订阅者建立通信连接之前，发布者就已经发布了主题。所以在等待与订阅者建立通信之前，涉及到以下几个基本的步骤：

a. 从我们创建的数据写入者获得当前状态

b. 确保发布与当前状态相互匹配

c. 创建一个等待集合

d. 使得当前状态依附于等待集合

e. 获得与发布相互匹配的状态

f. 如果当前匹配数量是 1 或者更多，从等待集合中分离出状态并在处理后发布

g. 等待这个等待集合（在指定期间内是有界的）

h. 迭代循环至步骤五

对应流程如图 5.7 所示。

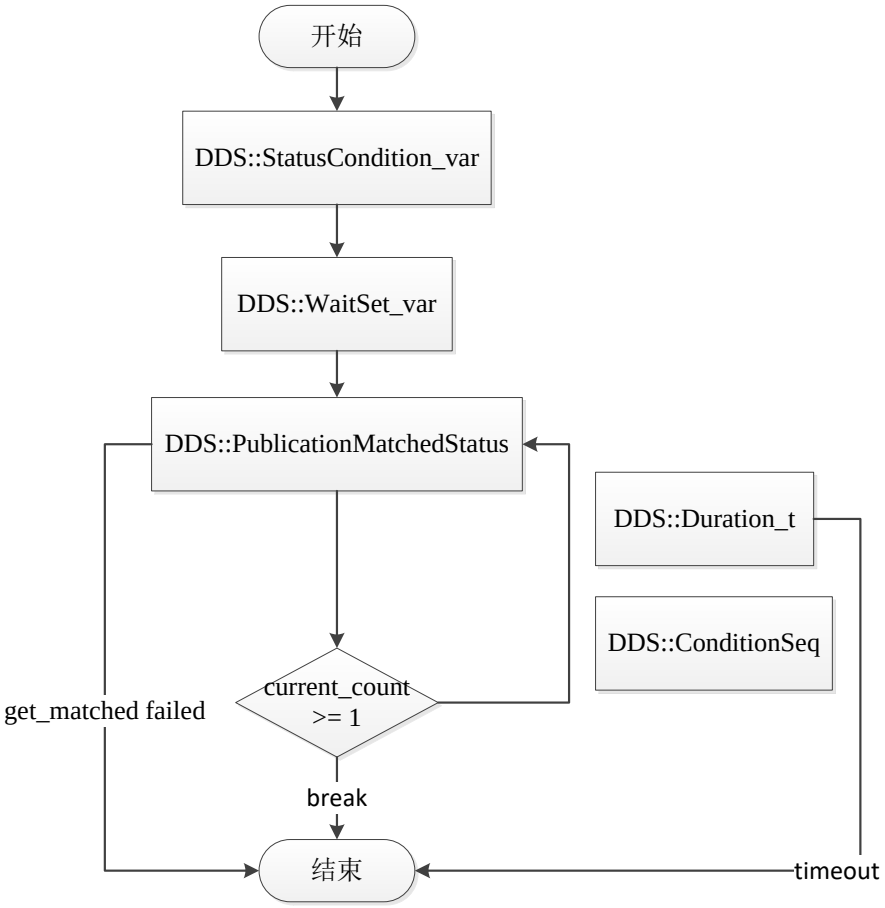

图 5. 7 等待与订阅者建立通信流程图

<5> 发布

消息发布的实现相当直接。对于所有的循环迭代，调用 write()使得一个消息被分发至所有已经建立连接通信的订阅者，这些订阅者为我们的主题被注册。因为 subject_id 是消息的主键，所有时间的 subject_id 被增加并且 write()被调用，一个新的实例被创建。调用 write()指定实例用于发布样例。无论是 register_instance()返回的一个句柄还是 DDS::HANDLE_NIL 都应该能够通过。通过一个 DDS::HANDLE_NIL 值指示数据写入者应该决定实例，通过检查样例的主键。核心代码如图 5.8 所示。

```
Messenger::Message message;
message.subject_id = 99;
message.from = "Comic Book Guy";
message.subject = "Review";
message.text = "Worst. Movie. Ever.";
message.count = 0;
for (int i = 0; i < 10; ++i) {
        DDS::ReturnCode_t error = message_writer->write(message,
                DDS::HANDLE_NIL);
        ++message.count;
        ++message.subject_id;
        if (error != DDS::RETCODE_OK) {
                // 日志或者其他方式来处理错误状态
                return 1;
        }
                                }
```

图 5.8 发布过程实现核心代码

5.4 订阅过程实现

大部分订阅者代码与发布者的是相近或者完全相同的。如果代码相似本文会省略掉一部分代码，从而专注于实现细节的差异。

<1> 初始化参与者

订阅者开始的实现部分与发布者是完全相同的，我们会初始化服务并加入域中。

<2> 注册数据类型和创建对应主题信息对象

下一步，初始化消息类型和主题。当主题在域中以相同的数据类型和兼容的 QoS 策略已经被初始化时，create_topic()函数的调用会返回一个引用来响应现存的主题。如果这个类型和 QoS 策略在 create_topic()调用中被明确指定，与现存的主题不相互匹配，那么之后会调用失败。此时，同样有 find_topic()这样的函数操作我们的订阅者使得其能够简单的获取现存的主题。

<3> 创建订阅者

下一步，以默认的 QoS 策略创建一个订阅者。如发布/订阅初始化模块类图 5.6 中通过 DDS::DomainParticipant_var 对象创建订阅者。

<4> 创建数据读取者和监听者

我们需要用创建的数据读取者来关联监听者对象，当数据可用时，可以用监听者对象来发现。如发布/订阅初始化模块类图中的构造了一个监听者对象。DataReaderListenerImpl 类将应用于下一部分。

监听者被分配在堆上并且被分配给一个 DataReaderListener_var 对象。这种类型提供引用计数行为，因此当最后的引用被移除时，监听者自动清理。在 OpenDDS 应用程序代码中这种堆分配是典型的应用方式并且使得应用程序开发人员从纷繁复杂的分配对象的生命周期管理中解脱出来。

接下来，创建数据读取者并且管理到主题上，其中使用默认的 QoS 策略属性和刚刚创建的监听者对象。当前线程空闲，用于执行其他应用程序工作。当样例可用时，在一个 OpenDDS 线程上，我们的监听者对象将会被调用。

<5> 数据读取者监听实现

我们的监听者类实现 DDS::DataReaderListener 接口，被通过 DDS 说明标准定义。DataReaderListener 被封装在 DCPS::LocalObject 之中，解决了隐约继承对象，如_narrow 和_ptr_type。这个接口定义了一系列我们必须实现的操作，所有的操作都是用于通知我们不同的事件。OpenDDS::DCPS::DataReaderListener 为 OpenDDS 的特殊需求定义操作，如失连和重连事件的更新。接口定义源码如图 5.9 所示。

```
module DDS{
	local interface DataReaderListener : Listener {
		void on_requested_deadline_missed(in DataReader reader,
			in RequestedDeadlineMissedStatus status);
		void on_requested_incompatible_qos(in DataReader reader,
			in RequestedIncompatibleQosStatus status);
		void on_sample_rejected(in DataReader reader,
			in SampleRejectedStatus status);
		void on_liveliness_changed(in DataReader reader,
			in LivelinessChangedStatus status);
		void on_data_available(in DataReader reader);
		void on_subscription_matched(in DataReader reader,
			in SubscriptionMatchedStatus status);
		void on_sample_lost(in DataReader reader, in SampleLostStatus status);
	};
                                        };
```

图 5.9 接口定义实现源码

代码中监听者类引出许多监听者简单打印状态的操作。唯一真正需要的是on_data_available()并且它是我们唯一需要研究的这个类的成员函数。函数中缩小了通用数据读取者，进入对类型明确的 MessageDataReader 接口的监听者。从消息读取者获得下一个样例。如果获取是成功的并且返回有效的数据，一般会打印所有消息的字段。

当读取数据时，数据内部可能存在无效数据。当存在有效数据时，有效的数据标志会指示。为了回调通知的目的，有两种有效数据的样例被传递给监听者。其一是处理通知，当 DataWriter 明确地调用 dispose()时，它将被获得。其二是未注册通知，当 DataWriter 明确地

调用 unregister()时，它将被获得。处理通知携带实例状态集合被传递给 NOT_ALIVE_DISPOSED_INSTANCE_STATE 并且未注册通知携带实例状态集合传递给 NOT_ALIVE_NO_WRITERS_INSTANCE_STATE。

此外，如果样例数据可获得，服务将再次调用这个函数。但是，一次读取一个简单的值不是一种最高效地方式用于处理获得的数据。数据读取者接口提供一系列不同的选项用于在更多高效的方式中处理数据。

5.5 网络通信中异构系统数据格式适配器的设计与实现

5.5.1 数据格式适配器的设计

在飞机协同设计中，各厂商间使用的专业软件系统存在差异，又因为各自的业务需求和生产环境等因素的不同，造成使用的操作系统、编程语言、数据库等方面有很大区别。为了解决异构系统间数据格式不统一等问题，本文基于 JSON 数据交换格式，设计并实现了 JSON 格式的数据适配器。希望通过对原有数据适配器的改进来使得在飞机协同设计环境下系统整体存取时间性能的提升。数据发送与接收时序图如 5.10 所示。

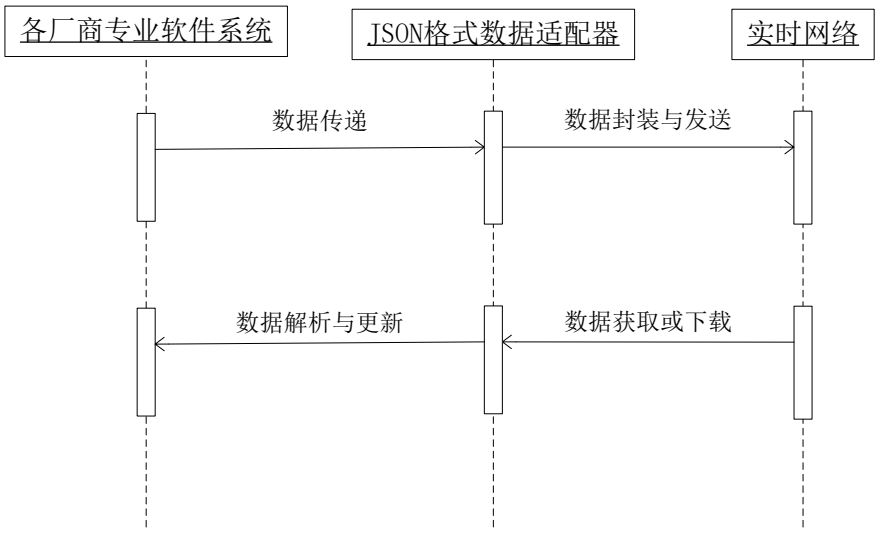

图 5. 10 数据发送与接收时序图

首先，从各厂商专业软件系统中提取相关主题数据和 QoS 策略信息传递给 JSON 格式数据适配器，在其内部会对数据进行封装，其中会有 JSON 封装器组件，用于数据信息的排列与拼接从而封装成规范的 JSON 格式信息并传递到实时网络中以供其他厂商的专业软件系统获取。当实时网络中有本地专业软件系统感兴趣的内容信息时，进行数据的获取或者下载，再将获取的 JSON 文件传递给 JSON 格式数据适配器，内部通过开源的接口函数实现来解析 JSON 文件信息，转换为相应的集合对象，然后逐层封装成本地数据模型对象以供本地系统处理使用。JSON 格式数据适配器因为随着数据量的增大，封装和解析的时间会过长，消耗资源也会过多，所以可以做成一个单例，在本地专业软件系统运行时可以始终维持其生

命周期的存在而不必每次使用都创建，以节省每次创建和释放的系统开销。

5.5.2 数据格式适配器的实现

本文采用的 Json-lib 解析方式来实现对 JSON 文件的解析。核心代码如图 5.11 所示

```
File file = new File("C:\\Users\\...\\modelDataInfo_100_QoS.json");
Scanner scanner = null;
StringBuilder buffer = new StringBuilder();

scanner = new Scanner(file, "gb2312");
while (scanner.hasNextLine()) {
    buffer.append(scanner.nextLine());
}
scanner.close();
String str = buffer.toString();

JSONObject jb = JSONObject.fromObject(str);
jb = (JSONObject) jb.get("DATAINFOLIST");
JSONArray ja = (JSONArray) jb.get("INFO");
```

图 5.11 数据格式适配器实现源码

图 5.11 中首先获取一个 file 对象，即是文件在磁盘上的句柄。创建一个 Scanner 对象和 StringBuilder 对象，前者设置对应文件句柄和编码后循环迭代逐行扫描文件中的内容，之后将内容添加到 StringBuilder 对象中，即保存在内存中。再将 StringBuilder 对象用转换为字符串的接口函数转换为字符串，用字符串创建 JSONObject 对象，最后逐层获取文件中的内容，直到获取为一个 JSONArray 对象为止。至此 JSON 文件的解析完成，可以以数组的方式来进行接下来的操作。此外，这些方法和内容还可以被封装在一个函数当中，对外提供对参数的方法以便实现不同的业务需求。

5.6 DDS 发布/订阅中自动发现算法的设计与实现

5.6.1 自动发现算法设计

在数据分发服中，需要根据结点之间的主题信息是否相同以及 QoS 策略是否兼容来最终确定主题是否匹配。而基于布隆过滤器的自动发现算法在各结点之间传递数据时效率较低。而且，不支持 QoS 兼容性判定，导致最终的主题匹配率很低。当前研究采用服务力向量算法只是对 QoS 进行了粗判，主题匹配率仍然不高。本文进行了两个方面的改进。第一，采用压缩的布隆过滤器 CBF 替代原有过滤器，有效地降低了网络中传输的数据量，减缓了内存的消耗；运用了更少的 hash 函数，降低了计算的复杂度；同时，可以获得更小的虚警率，提高了主题信息的匹配率。第二，添加 QoS 策略兼容性判定的设计与实现，从而极大地提高了主题的匹配率。自动发现算法的流程设计图 5.12 所示。

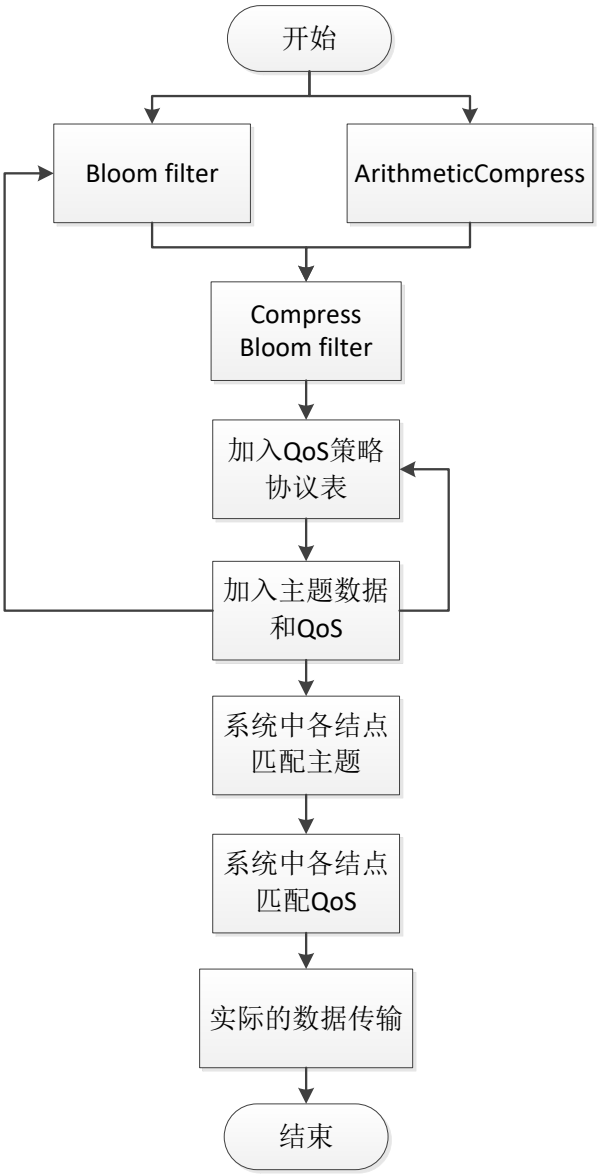

图 5. 12 自动发现算法流程设计图

图 5.12 中，Bloomfilter 是采用文献[41]中 Github 上的源码实现，同样的 ArithmeticCompress 是采用文献[40]中 Github 上的源码实现。首先创建 BF，使用少量数据初始化，并使用算术编码提供的压缩接口对 BF 进行压缩，从而生成 Compress Bloom filter。下一步，将第四章设计的 QoS 兼容策略协议表加入到解析数据过程中，构建好基于 CBF 和 QoS 兼容的自动发现算法后，开始加入网络通信中实际的数据，通过主题数据构建 BF，通过 QoS 策略协议表解析后得到每个主题对应 QoS 策略并缓存。接着，利用网络解析数据构建容器中的数据与本地期望订阅的主题和 QoS 策略进行匹配判断。最后，进行主题和 QoS 策略都匹配成功的实际数据传输。

5.6.2 自动发现算法的实现

<1> 应用 CBF

实现 Compress Bloom filter 的核心代码如图 5.13 所示。

```
String data2 = (String)jsonObj.get("NAME");
tmp.add(data2);
BloomFilter bf = new BloomFilter(8000, 100);
bf.addAll(tmp);

byte[] b = bitSet2ByteArray(bf.getBitSet());
byte[] compressed = compress(b);

public static byte[] compress(byte[] b) throws IOException {
        InputStream in = new ByteArrayInputStream(b);
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        BitOutputStream bitOut = new BitOutputStream(out);

        FrequencyTable freq = getFrequencies(b);
        ArithmeticCompress.writeFrequencies(bitOut, freq);
        ArithmeticCompress.compress(freq, in, bitOut);
        bitOut.close();
        return out.toByteArray();
}
public static byte[] bitSet2ByteArray(BitSet bitSet) {
        byte[] bytes = new byte[bitSet.size() / 8];
        for (int i = 0; i < bitSet.size(); i++) {
                int index = i / 8;
                int offset = 7 - i % 8;
                bytes[index] |= (bitSet.get(i) ? 1 : 0) << offset;
        }
        return bytes;
}
```

图 5. 13 Compress Bloom filter 的核心代码

首先使用之前的开源组件 JSONObj 获取解析 JSON 数据交换格式后的对象，通过接口方法获取主题名（对应 JSON 格式中“NAME”键）。这里 tmp 对象是一个语言内容器对象，用于存储所以主题信息。接着创建 Bloom filter，容器大小为 8000kb，元素个数限制为 100 个。接着将 tmp 添加到 Bloom filter 中。因为算术编码的开源实现对外提供是一个 byte 数组的参数限制，所以需要在传递前将 Bloom filter 中的 BitSet 集合先转换为 byte 数组，这里调用 bitSet2ByteArray 函数方法。转换后调用算术编码对外提供的 compress 接口方法，在其中使用算术编码内部实现的流对象进行压缩处理。

<2> 构建及判别 QoS

实现 QoS 分为两个部分，首先是构建 QoS 兼容策略结构体，如表 5.1 所示。

表 5. 1 构建 QoS 兼容策略结构体

QoS 策略	子选项	占位
RELIABILITY	RELIABLE	1^0
	BEST_EFFORT	1^1
DEADLINE	INFINITY	1^2

	NOT_INFINITY	1^3
DURABILITY	PERSISTENT	1^4
	TRANSIENT	1^5
	TRANSIENT_LOCAL	1^6
	VOLATILE	1^7
OWNERSHIP	SHARED	1^8
	EXCLUSIVE	1^9
PRESENTATION	GROUP	1^10
	TOPIC	1^11
	INSTANCE	1^12
LIVELINESS	MANUAL_BY_TOPIC	1^13
	MANUAL_BY_PARTICIPANT	1^14
	AUTOMATIC	1^15
DESTINATION_ORDER	BY_SOURCE_TIMESTAMP	1^16
	BY_RECEPTION_TIMESTAMP	1^17
DESTINATION_ORDER	ZERO	1^18
	NOT_ZERO	1^19

表 5.1 中 8 个 QoS 兼容策略是必须满足 RxO 模式，按照第四章中设计的 QoS 策略兼容性协议表对 8 个大项 20 个小项进行编码，编码成一个多枚举选项的结构体。子选项依次占一位从 1 的 0 次方到 1 的 20 次方，每个主题的 QoS 存储需要额外的 3 个字节。

之后，是嵌入 JSON 数据格式解析过程中的 QoS 策略判别并缓存。核心实现如图 5.14 所示。

```
for (int i = 0; i < size; ++i) {

        JSONObject jsonObj = (JSONObject)ja.get(i);
        String dura = (String)jsonObj.get("_DURABILITY_");
        if (dura.equals("_PERSISTENT_")) {
               qosOptions[i] |= DDS_OPTIONS.DDSQOS_DURABILITY_PERSISTENT.getSequence();
        } else if (dura.equals("_VOLATILE_")) {
               qosOptions[i] |= DDS_OPTIONS.DDSQOS_DURABILITY_VOLATILE.getSequence();
        } else if (dura.equals("TRANSIENT_LOCAL")) {
               qosOptions[i] |= DDS_OPTIONS.DDSQOS_DURABILITY_TRANSIENT_LOCAL.getSequence();
        } else if (dura.equals("_TRANSIENT_")) {
               qosOptions[i] |= DDS_OPTIONS.DDSQOS_DURABILITY_TRANSIENT.getSequence();
        }

        ...

        String data2 = (String)jsonObj.get("NAME");
        recvqosName[i] = data2;
        tmp.add(data2);
}
```

图 5. 14 QoS 策略判别并缓存实现

图中 size 表示需要解析的数据量，循环迭代中获取当前的 JSON 对象，根据 8 个 QoS 名分别获取并判断子选项后缓存到 qosOptions 容器当中用于之后的主题匹配。而且对各个子选项都按照第四章中设计的 QoS 策略兼容性协议表的适用规则的描述进行匹配。描述中各子选项存在一定的优先级，如图中 DURABILITY 选项，当请求方要求 PERSISTENT 的子选项时，提供方必须提供 PERSISTENT。否则匹配不成功。相反，当请求方为 VOLATILE 时，提供方可以是 DURABILITY 的任意一个子选项。

5.7 测试实验与系统分析

5.7.1 功能测试

全面的功能测试是指针对基于 DDS 的飞机协同设计数据分发系统的各项功能模块进行的整合测试，主要工作是在现实网络通信环境下，协同各飞机部件制造厂商之间的数据传输，整合各厂商异构系统数据格式和数据分发服务 DDS 的发布/订阅的核心模块自动发现两项核心功能及相关附属功能进行的综合性测试。图 5.15，展现了基于 DDS 的飞机协同设计数据分发系统主题管理界面的展示及系统中集成的功能。

图 5. 15 基于 DDS 的飞机协同设计数据分发系统的功能配置及集成功能

基于 DDS 的飞机协同设计数据分发系统主要分为主题管理、数据服务、监控管理和工具继承等功能。主题管理界面主要是创建、删除和查找相关主题信息。单个主题有主题名、操作类型、QoS 配置和 XSL 文件等信息，新增主题信息如图 5.16 所示，图中除配置基本信

息之外还具有可选的 QoS 详细配置，根据本地专业软件系统的业务需求定制私有的 QoS 兼容策略，并且对于订阅主题信息需要上传对应的 JSON 文件用于数据文件的解析。

图 5. 16 基于 DDS 的飞机协同设计数据分发系统新增主题

在真实的网络环境下，各飞机部件制造厂商通过自动发现算法实时的获取系统中各厂商的状态信息，自由的发布订阅主题相关信息，并通过 Bloom filter 广播对应信息，当通信双方协定好主题信息和服务质量兼容策略时，真实的消息会被发送到接收方，并应用本地系统数据格式适配器解析为高效地数据传输格式用于提取有效信息。其中，本文在通信前，通信各方都维护一份协议规则以便之后的兼容性检查和判断。开始通信，CBF 携带 QoS 传递到请求方，首先通过 CBF 判断当前接收消息是否有匹配的主题名，匹配成功后进行 QoS 甄别。在经过 QoS 兼容策略协议表及其编码实现严格的 QoS 判别之后，确认当前主题完全匹配，开始主题数据信息的传递与交互。

5.7.2 性能测试

性能测试主要是针对基于 DDS 的飞机协同设计数据分发系统的整体性能进行测试，主要测试真实网络环境下，各飞机部件制造厂商交换的数据信息量，其中包括使用了 CBF 压缩与 QoS 兼容策略完整匹配的数据量与改进前的对比，以及使用改进的数据交换格式后带来的数据量上的优势。

图 5. 17 基于 DDS 的飞机协同设计数据分发系统监控操作

图 5.17 中展示了设置监控的部分界面，其中可以根据业务需求选择过滤器的类型，选择自定义的 QoS 兼容策略，以及通信双方匹配 QoS 时所采用的匹配方式和使用的基础数据交换格式，也即使用的对应数据格式适配器。

表 5. 2 真实网络环境下两种数据交互方式匹配主题数时间损耗表

主题数 / 配置选项	500	1000	1500	2000	2500	3000
基于 SAV 的 BF + XML 数据格式	32026ms	68863ms	97553ms	130998ms	167555ms	201988ms
基于 QoS 的 CBF + JSON 数据格式	28492ms	60801ms	84564ms	114349ms	146676ms	177932ms

表 5.2 是在真实网络环境下，500 主题数到 3000 主题数场景中选择基于 SAV 的 BF 与 XML 数据格式配置和基于 QoS 的 CBF 与 JSON 数据格式配置的时间消耗测试数据。由测试数据可以看出，在整体基于 DDS 的飞机协同设计数据分发系统中，本文提出的基于 JSON 数据交换格式的飞机协同设计应用适配器与基于 CBF 和 QoS 兼容的自动发现算法相较于原始的设计方案使时间性能提升了 13%到 15%，从而为未来设计更加复杂的业务系统提供了更优的设计方案与解决思路。

5.8 本章小结

本章主要介绍了基于 DDS 的飞机协同设计数据分发系统的实现，由前文中总体架构设计和流程设计提到的网络通信中异构系统数据格式适配器和数据分发服务 DDS 的发布/订阅的核心模块自动发现两个核心部分的设计，并结合第三四章基于 JSON 数据交换格式的飞机

协同设计应用适配器和基于 CBF 和 QoS 兼容的自动发现算法实现了对应的功能模块，模拟真实网络环境下，各飞机部件制造厂商实时的数据交换，并对其功能和性能进行了测试。

第六章　总结与展望

6.1 论文工作总结

协同设计技术长久以来都在不断发展精进中，尤其运用于大型项目，复杂工作和生产环境下多任务协同开发。早期系统功能只是简单地将生产流程秩序化，尽量通过严格的生产规范来降低人事协调带来的种种阻碍。但随着生产的目标对象逐渐复杂，需求也不断的迭代更新，分发系统的交互响应能力又有了更高的要求，高响应性，高扩展性，高负载可用性的，无论是当前还是未来都始终是研究工作的重心。为了后期研究工作的顺利进行，需要从系统底层机制出发，控制好网络通信传输的基础元素，尽可能的优化核心实现机制，从根源上对系统进行了调整和改进。

本文在飞机协同设计的生产与工作环境下，设计与实现基于 DDS 的数据分发系统，研究工作从数据传输格式、发布订阅中的自动发现机制等方面开展了深入的探索与实践论证。关于相关问题的展开，分以下几个方面：

（1）针对数据分发服务基础设施、开发者文档与相关源码进行了深入的分析与研究。权衡比较国内外在不同适用场景不同维度下，数据分发服务的适用特性与应用性调整。结合飞机协同设计开发环境，设计与规划系统优化性能改进方案。

（2）在综合探究原始系统各部分组成结构与功能的前提下，考虑到飞机各部件生产单位因部件个体差异性，导致各自使用的资源方式有所不同。为了使系统各方高效地共享生产数据，也即网络通信对端获得统一的数据传输方式，设计并实现了基于 JSON 数据交换格式的飞机协同设计应用适配器，同时也比较了其与原始 XML 方式的各种不同。在确定前者简化了封装，解析和存储等优势后，通过基于飞机协同设计的主题数据进行了实验的论证。实验表明，本文提出的适配器应用方案获得了更优的编解码和存取性能。

（3）在理解了原始基于 Bloom filter 的发布订阅自动发现机制后，深入探索了近些年来针对此问题的相关改进工作，采用了更加适用于飞机协同设计环境下的 Compressed Bloom Filter。CBF 的应用相较原始过滤器在网络中的数据传输效率更高，传递数据量更少。也因此可以用更少的 hash 函数来获得比以往更优的错误率。同时，在 CBF 的基础上，提出了基于 QoS 判断机制，设计了网络中通信对端支持的 QoS 判别规则表，依据规则实现了预先甄别方法，并针对飞机协同设计的主题数据进行了实验论证。实验表明，基于 QoS 的方式极大地提高了主题的匹配效率。

（4）根据当前可应用的飞机各部件主题发布订阅的交互数据，模拟真实生产和作业环境，本文设计并初步实现了基于 DDS 的飞机协同设计数据分发系统。完成了基于 JSON 的应用适配器，基于 QoS 的自动发现等功能。

6.2 未来研究工作展望

即便本文基本完成了基于 DDS 的飞机协同设计数据分发系统的设计与实现，并且克服了原始基于 XML 的应用适配器封装解析复杂的过程，运用基于 JSON 的应用适配器作为改进方案，使得系统整体存取时间性能平均提升了 15.68%。同时，为了更加高效的数据传递和服务质量的支持，实现了基于 QoS 的 CBF 自动发现机制，降低了网络传输数据量，简化了 hash 函数的计算量且极大地提高了主题的匹配率。但是，随着项目规模的不断扩大，业务需求的不断变化，本文可以从以下几个方面进行改进：

（1）基于信息感知组件内部的日志系统。在发布订阅的实现过程中，需要数据库系统中数据实时响应，对变化数据的捕获方法显得尤为重要。基于触发器，日志，API 等是当前相对良好的解决方案。但为了适应未来高实时性，高响应性等需求，应该实现一套完善的日志管理系统。

（2）全局数据空间（GDS）的改进与优化。作为数据分发服务（DDS）的核心部分，全局数据空间主要包含发布主题表数据信息，订阅登记表数据信息，发布数据缓冲区，订阅失败队列四个部分。同时，也包括协调主题的匹配和管理、发布订阅者的自动发现。尽管现有的方案可行，但是仍然存在很大的改进空间。

（3）在真实的飞机协同设计与生产环境下，各厂商之间的交互数据可能更加复杂多变，主题与主题，选项与选项间的关系也更加繁琐，相应的数据处理逻辑需要反复测试和修改才能使得系统更加完善。所以，应用于实际场景下系统测试是未来工作的重心。

参考文献

[1] Kvan T. Collaborative design: what is it?[J]. Automation in Construction, 2000, 9(4):409-415.

[2] OMG, Data distribution service for real-time systems specification [S].Version1.0. http://www.omg.org/spec/DDS/ Dec.2004.

[3] OMG, Data distribution service for real-time systems specification [S].Version 1.1. http://www.omg.org/spec/DDS/ Nov.2005.

[4] OMG, Data distribution service for real-time systems specification [S].Version 1.2. http://www.omg.org/spec/DDS/ Jan.2007.

[5] 兰晓伟，于长云. JAVA RMI 与 CORBA 的集成研究[J]. 天津理工大学学报，2005, 21(3):26-29.

[6] 王东. 基于发布/订阅的数据集成技术及其管理系统的研究与实现[硕士学位论文]; 国防科学技术大学, 2005.

[7] 秦耀坤，曹元大. 用 TongIntegrator 实现电子商务中的系统集成[J]. 现代电子技术, 2004, 27(8):28-29.

[8] 王东，李若梅，谭庆平,等. InforBroker 在国税应用整合中的应用[J]. 计算机工程与科学, 2006, 28(2):101-103.

[9] Meeting Real-Time Requirements in Integrated Defense Systems, http://www.rti.com.2007

[10] 卢传富，蔡志明，夏学知. 数据分发服务体系结构的研究[J]. 计算机与数字工程，2008, 36(5):67-69.

[11] 梁林波. 基于 RTI-DDS 的无人机地面站通信系统研究与设计[硕士学位论文]. 电子科技大学, 2011.

[12] 杨震，阳洋. 基于 DDS 规范的战场信息分发中间件平台研究[J]. 通信技术，2009, 42(12):185-187.

[13] 谷青范，康介祥，冯国良，等. 动态自适应 DDS 实时中间件的研究与实现[J]. 计算机科学, 2012, 39(7): 36-38.

[14] Joung Y J, Wang J C. Chord: A two-layer Chord for reducing maintenance overhead via heterogeneity[J]. Computer Networks, 2007, 51(3): 712-731.

[15] Schmidt D C, van't Hag H. Addressing the challenges of mission-critical information management in next-generation net-centric pub/sub systems with opensplice dds[C]//Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on. IEEE, 2008: 1-8.

[16] 陈春甫. 基于 DDS 的数据分发系统的设计与实现 [硕士学位论文]; 复旦大学, 2008.

[17] 分布式数据分发服务中间件 CoreDX DDS .http://www.cnstech.com.cn/show.asp. 2009.

[18] CoreDX DDS Quick Start Guide for C Version 3.4 Nov 2011.

[19] Object Computing, Inc. OpenDDS, 2010. http://www.opendds.org.2014.

[20] The Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol(DDSI) Specification Version 2.1[S], 2009.

[21] Wang N, Schmidt D C, van't Hag H, et al. Toward an adaptive data distribution service for dynamic large-scale network-centric operation and warfare (NCOW) systems[C]//Military Communications Conference, 2008. MILCOM 2008. IEEE. IEEE, 2008: 1-7.

[22] Sanchez-Monedero J, Povedano-Molina J, Lopez-Vega J M, et al. Bloom filter-based discovery protocol for DDS middleware [J]. Journal of Parallel and Distributed Computing, 2011, 71(10): 1305-1317.

[23] Putra H A, Nugroho D A, Kim D S, et al. Discovery protocol for data distribution service in naval warships using extended counting bloom filters[C]//Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on. IEEE, 2013: 1-8.

[24] Ahmadi M, Wong S. A cache architecture for counting bloom filters[C]//Networks, 2007. ICON 2007. 15th IEEE International Conference on. IEEE, 2007: 218-223.

[25] 翟海波, 庄毅, 霍瑛. 基于服务力向量的发布/订阅自动发现算法[J]. 计算机工程, 2014, 40(9):51-54.

[26] Alazzawi A N, Abdelmoty A I, Jones C B. What can I do there? Towards the automatic discovery of place-related services and activities[J]. International Journal of Geographical Information Science, 2012, 26(2): 345-364.

[27] van Berkel S, Turi D, Pruteanu A, et al. Automatic discovery of algorithms for multi-agent systems[C]//Proceedings of he fourteenth international conference on Genetic and evolutionary computation conference companion. ACM, 2012: 337-344.

[28] Corsaro A, Schmidt D C. The Data Distribution Service–The Communication Middleware Fabric for Scalable and Extensible Systems-of-Systems[J]. System of Systems, 2012

[29] JSON: The Fat-Free Alternative to XML ［EB/OL］. 2011. http://www.json.org/xml

[30] Maeda K.Performance evaluation of object serialization libraries in XML，JSON and binary formats［C］Digital Information and Communication Technology and Its Applications，Second International Conference on. IEEE， 2012: 177－182.

[31] Cavalieri F，Guerrini G，Mesiti M. XSPath: Navigation on XML Schema-s Made Easy ［J］. IEEE Transactions on Knowledge and Data Engineering，2014，26(2) : 485－499.

[32] Maarouf,M.Y.,Chung,S.M..XML Integrated Environment for Service-Oriented Data Management[C]; 2008, 20th IEEE International Conference on Tools with Artificial

Intelligence;2008, 361-368.

[33] Phan B V，Pardede E．Active XML(AXML) research: Survey on the representation，system architecture，data exchange mechanism and query evaluation［J］Journal of Network and Computer Applications， 2014， 37: 348 － 364．

[34] Cavalieri F，Guerrini G，Mesiti M． XSPath: Navigation on XML Schema-s Made Easy[J]． IEEE Transactions on Knowledge and Data Engineering， 2014， 26(2) : 485 － 499．

[35] Ramon Lawrence. The space efficiency of XML[J]. Information and Software Technology, 2004, 46(4):753-759.

[36] Object Management Group. The Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol(DDSI) Specification[S]. Version 2.1, 2009.

[37] Mitzenmacher M. Compressed Bloom filters[J]. Networking IEEE/ACM Transactions on, 2002, 10(5):604-612.

[38] Langdon G G. An Introduction to Arithmetic Coding[J]. Ibm Journal of Research & Development, 1984, 28(2):135-149.

[39] Arithmetic Coding https://github.com/nayuki/Reference-arithmetic-coding

[40] Bloom Filter https://github.com/MagnusS/Java-BloomFilter

[41] Michi Henning, Steve Vinoski Advanced CORBA Programming with C++Addison Wesley Longman.1999.10:101-103

[42] Microsoft Component Services--A Technology Overview.Wiley Computer Publishing. 1998.01:31-34

[43] David Reilly,Java 与 CORBA --一种光滑的融合.http:// www.corba.com.cn/download/ 200122603/ index.html

[44] Dirk Slama, et al. Enterprise CORBA. Preritice Hall. 1999:23-31

[45] Dean J, Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters.[J]. Communications of the ACM, 2008, 51(1): 107-13.

[46] Palankar M R, Iamnitchi A, Ripeanu M, et al. Amazon S3 for science grids: a viable solution?[C]// International Workshop on Data-Aware Distributed Computing. ACM, 2008.

[47] Isard M, Budiu M, Yu Y, et al. Dryad: distributed data-parallel programs from sequential building blocks[J]. Acm Sigops Operating Systems Review, 2007, 41(3):59-72.

[48] Tayal S. Task scheduling optimization for the cloud computing systems[J].International Journal of Advanced Engineering Sciences And Technologies(IJAEST), 2011, 5(2): 111-5

[49] Buyya R, Broberg J, Goscinski A M. Cloud Computing: Principles and Paradigms[C]// Wiley Publishing, 2011:1-41.

[50] Younge A J, Von Laszewski G, Wang L, et al. Efficient resource management for Cloud computing environments[C]// International Conference on Green Computing. IEEE Computer Society, 2010:357-364.

[51] Barham P, Dragovic B, Fraser K, et al. Xen and the art of virtualization[J]. Acm Sigops Operating Systems Review, 2003, 37(5):164-177.

[52] Azeez A, Perera S, Gamage D, et al. Multi-tenant SOA Middleware for Cloud Computing[C]// IEEE International Conference on Cloud Computing, Cloud 2010, Miami, Fl, Usa, 5-10 July. 2010:458-465.

[53] Maximilien E M, Ranabahu A, Engehausen R, et al.IBM altocumulus: a cross-cloud middleware and platform[J]. IBM altocumulus: a cross-cloud middleware and platform – ResearchGate, 2009, 805-6.

[54] 郑晨. 基于 OpenStack 的移动设备消息中间件研究与实现[D]. 东北大学, 2013.

[55] 翁世南. 面向流程定制的 RFID 云服务中间件的研究和设计[D]. 浙江大学, 2012.

[56] Barry D K. Web Services, Service-Oriented Architectures, and Cloud Computing, Second Edition: The Savvy Manager's Guide[M]// Web Services, Service-oriented Architectures, and Cloud Computing (Second Edition). 2013.

[57] Canuto M, Guitart J. Integrated policy management framework for IaaS Cloud middleware[J]. Computing, 2016, 98(5):471-494.

[58] Loewen G, Galloway M, Vrbsky S. Designing a Middleware API for Building Private IaaS Cloud Architectures[C]// IEEE, International Conference on Distributed Computing Systems Workshops. IEEE Computer Society, 2013:103-107.

[59] 褚轶群. 基于预测与预约机制的网格任务调度中间件研究[D]. 上海交通大学, 2008.

[60] 张再东. 私有云平台中共享虚拟资源访问管理和任务调度的设计与实现[D]. 北京邮电大学, 2015.

[61] 谷方舟，沈波. JSON 数据交换格式在异构系统集成中的应用研究[J]. 铁路计算机应用，2012，2：1~4

[62] 高静，段会川． JSON 数据传输效率研究[J]. 计算机工程与设计，2011，32：2267~2270

[63] 韩敏，冯浩． 基于 JSON 的地理信息数据交换方法研究［J］.测绘科学，2010， 35(1).

[64] 潘翔，徐小良. 基于 JSON 的网络传输优化策略研究[N].杭州电子科技大学学报（自然科学版），2015.

[65] 刘文，甘志春，李文，王更辉. 基于 XML 和 JSON 的格式化网络参数文件研究[J]. 计算机与网络，2013.

[66] 陈玮，贾宗璞. 利用 JSON 降低 XML 数据冗余的研究[J]. 计算机应用与软件，2012

[67] 张涛，黄强，毛磊雅，等． 一个基于 JSON 对象的对象序列化算法[J]． 计算机工程

与应用，2007，43(15).

[68] 崔璨，倪宏．使用 JSON 对 AJAX 技术中的 XML 性能的优化仿真[J].通信技术,2009,42(8):108-114.

[69] 方跃坚，余枝强，翟磊等. 一种混合并行 XML 解析方法[J]．软件学报，2013，24(6) : 1196 － 1206.

[70] 韩义波，宋莉，宋俊杰．Ajax 技术结合 XML 或 JSON 的使用比较[J].电脑知识与技术,2009,5(1):101- 103.

[71] 翟海波. 面向飞机协同设计的 DDS 技术研究[D]. 南京航空航天大学, 2014.

[72] 卞华星. 基于 DDS 的飞机协同设计数据服务中间件的设计与实现[D]. 南京航空航天大学, 2015.

[73] Tarkoma S, Rothenberg C E, Lagerspetz E. Theory and practice of bloom filters for distributed systems[J]. Communications Surveys & Tutorials, IEEE, 2012, 14(1): 131-155.

[74] Fan L, Cao P, Almeida J, et al. Summary cache: a scalable wide-area Web cache sharing protocol[C]// IEEE/ACM Transactions on Networking. 2000:281-293.

[75] Ahmadi M, Wong S. A memory-optimized bloom filter using an additional hashing function[C]//Global Telecommunications Conference. IEEE GLOBECOM . 2008: 2479–2483.

[76] Limam N, Ziembicki J, Ahmed R, et al. OSDA: Open service discovery architecture for efficient cross-domain service provisioning[J]. Computer Communications, 2007, 30(3): 546-563.

[77] Pal S K, Sardana P. Bloom Filters & Their Applications[J]. International Journal of Computer Applications Technology and Research, 2012, 1(1): 25-29.

致 谢

两年多的研究生学习与生活即将画上圆满的句号，从充斥着青涩感的我到现在略有感悟，这一切都发生在短暂的时光里，一路走来满满的都是心酸与汗水。研究生生涯终将结束，希望能够借此次论文对十多年的学生生涯做完美的总结。即将走入社会的我们，带着激情洋溢的心情踏上新的征程。

一直以来，我都十分感谢我的导师陈丹副教授和庄毅教授。没有她们就不会有如今的我，无论是在生活与学习上，还是在人生体会和感悟上，我都谨遵您们深深的教诲。您们以严谨规范的科研态度，认真负责的工作态度，和蔼亲切的生活态度指导着我们一步步走到现在，在您们的见证下，我们不断经历挫折，不断的克服又不断变得坚强，在这里，再一次对您们表示深深的敬意。最后，特别感谢两位导师在百忙之中抽出时间帮助我修改论文。

同时，需要感谢实验室的各师兄师姐给予我生活与学习上极大的帮助与鼓励，感谢你们在一路的学术研究生涯中给我提出的各种建议，各种方向，以及分享你们宝贵的研究心得与经验。在这样一个“大家庭”当中我深深的体会到了团队合作的重要性，也与大家共同创造了一段美好的过去。

最后，感谢我的父母很久以来对我的支持，您们的鼓励是我前进的动力，为了让您们生活的更好同样也是我前进的动力。

在学期间的研究成果及发表的学术论文

攻读硕士学位期间发表（录用）论文情况

1. 钱哨，陈丹. 基于 JSON 数据格式的飞机协同设计应用适配器[J]. 计算机与现代化, 2016(8):123-126.
2. 钱哨，陈丹. 基于 CBF 和 QoS 兼容的自动发现算法[J]. 计算机工程与科学.第一作者，审稿中。

攻读硕士学位期间参加科研项目情况

1. 2014 年至 2016 年，参与“飞机协同设计系统的研发”项目，主要负责研究探索 DDS 及开源项目 OpenDDS 源码的阅读与改进。