中图分类号：TP393　　论文编号：1028716 15-S062
学科分类号：081203

硕士学位论文

基于 DDS 的飞机协同设计数据服务中间件的设计与实现

研究生姓名　　卞华星
学科、专业　　计算机科学与技术
研 究 方 向　　计算机网络与分布式计算
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

二〇一五年三月

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

Design and implementation of aircraft collaborative design data services middleware based on DDS

A Thesis in
Computer Science and Technology

by

Bian Huaxing

Advised by

Associate Prof.Chen Dan

Prof.Zhuang Yi

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Engineering

March, 2015

承诺书

本人声明所呈交的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：_____

日　　期：_____

摘 要

随着网络和分布计算技术的发展，协同设计技术已成为当前的研究热点之一。在协同设计平台上可方便的进行设计方案、多学科交流和信息共享。多个异地单位可以协同完成一项复杂的工程，然而如何保证各个异地单位之间数据的及时交互问题也日益突出。现有的数据交互平台虽然能够有效实现数据的交互，但是针对数据的实时性或数据服务质量控制方面却显得不足。国际对象管理组织 OMG 针对不同的分布式实时计算环境推出了数据分发服务 DDS 规范，提供了一种异构平台之间的互操作标准。因此，开展基于 DDS 的飞机协同设计数据服务中间件的研究具有重要的应用价值。

本文在研究 DDS 技术和中间件技术的基础上，结合飞机协同设计特点，设计并初步实现了基于 DDS 的飞机协同设计数据服务中间件(ACD_DSM)。论文主要工作有：1）对数据分发服务 DDS 相关标准、体系结构、模块划分以及关键技术进行了研究；2）分析了开源 OpenDDS 的全部代码，并将其改进后应用到飞机协同设计系统的研发中，设计并实现了 ACD_DSM 的消息代理组件；3）针对各子系统数据异构的问题，采用 XML 相关技术，设计并实现了应用适配器，为解决各专业软件系统数据异构的问题奠定了基础；4）重点研究了 DDS 的发布/订阅自动发现机制，并结合目前主流自动发现协议，提出了一种改进的自动发现协议 DCFSDP，在不影响系统其他方面性能的情况下，可降低系统内存资源消耗和网络负载。

目前论文研究工作已初步实现了基于 DDS 的飞机协同设计数据服务中间件，并尝试应用到飞机协同设计系统中，该系统正在试用。测试实验结果表明该数据服务中间件能够满足飞机协同设计各专业软件信息交互的要求；并通过实验验证了数据分发服务 DDS 应用到飞机协同设计中的有效性。

关键词：数据分发服务，发布/订阅，协同设计，自动发现机制，适配器

ABSTRACT

With the rapid development of distributed technology and information technology, the collaborative design technology makes a complex project that could be completed by several organizations from different places, which has become one of the current hot issue. Although the existing interactive data platforms can effectively implement the data interaction, in the aspect of real-time or service quality control they make an inadequate performance. Under this circumstance, the international Object Management Group launched Data Distribution Service specification for different distributed real-time computing environment, which provides an interoperable standard and enables heterogeneous platforms to realize real-time data interaction. Therefore, it is valuable to research the designing of data service middleware based on.

This paper finished designing and initial implementing data service middleware through the aspect of DDS aircraft collaborative, which was based on the study DDS and middleware technology and combined with the features of aircraft collaborative design. The main works are: 1) Research on the relevant DDS standards, architecture, module partition and key technologies; 2)analysis the whole codes of OpenDDS and apply the improved OpenDDS to the research of aircraft collaborative design system; 3) For the problems of each heterogeneous data, use of XML-related technologies to design the application adapters. Through this solve the problem of the heterogeneous data of professional software systems; 4)Focus on the automatic discovery mechanism of DDS publishing and subscribing. Combined with the current mainstream auto-discovery protocol, propose an improved auto-discovery protocol DCF_SDP, which can further reduce the consumption of system memory resource and the network load without affecting the other performance of the system.

Thesis work currently initial realize the goal of designing of data service based on DDS aircraft programming, and has successfully be applied to the aircraft collaborative design system. Test results show that the data service middleware could meet the requirements of information exchange from current professional software system; in addition, verified the effectiveness when data distribution service DDS was applied to aircraft collaborative design.

Keywords: Data Distribution Service, publish/subscribe, collaborative design, auto-discovery mechanism, adapter

目 录

第一章 绪 论 1
1.1 研究背景及意义 1
1.2 国内外研究现状及发展趋势 2
1.2.1 数据集成研究现状 2
1.2.2 数据分发服务研究现状 3
1.2.3 研究发展趋势 5
1.3 论文主要工作 5
1.4 论文组织结构 6
第二章 相关知识及技术 9
2.1 数据分发服务（DDS） 9
2.1.1 DDS 的相关概念 9
2.1.2 DCPS 的模块划分 12
2.1.3 DDS QoS 策略控制 12
2.2 传统信息共享技术 14
2.2.1 Web 服务技术 14
2.2.2 传统消息中间件技术 16
2.2.3 现有信息共享技术的缺点 17
2.3 本章小结 17
第三章 基于 DDS 的飞机协同设计数据服务框架设计 18
3.1 需求分析 18
3.1.1 飞机协同设计特点 18
3.1.2 飞机协同设计应用场景 19
3.1.3 飞机协同设计功能需求 20
3.2 ACD_DSM 设计 21
3.2.1 ACD_DSM 结构设计 21
3.2.2 ACD_DSM 总体流程设计 22
3.3 OpenDDS 分析以及优缺点 24
3.3.1 OpenDDS 分析 24
3.3.2 OpenDDS 的优缺点 26
3.4 ACD_DSM 的信息感知组件设计 26
3.5 本章小结 27
第四章 ACD_DSM 的消息代理组件的设计 28
4.1 发布订阅通信模式 28

4.2 主题文件的定义与设计.....29
4.3 发布订阅自动发现机制的分析与改进.....30
4.3.1 基于 SDP 的自动发现协议.....30
4.3.2 基于 Bloom Filter 的自动发现协议 SDPBloom.....31
4.3.3 基于 DCF 的自动发现协议 DCFSDP 设计.....33
4.3.4 性能分析以及仿真实验结果分析.....38
4.4 ACD_DSM 的 QoS 策略控制.....43
4.5 本章小结.....44
第五章 ACD_DSM 的适配器的研究与设计.....45
5.1 适配器的功能与分类.....45
5.2 ACD_DSM 应用适配器结构设计.....46
5.3 ACD_DSM 应用适配器核心技术.....47
5.3.1 统一消息格式生成.....47
5.3.2 消息解析.....50
5.3.3 消息翻译.....51
5.4 本章小结.....52
第六章 ACD_DSM 在飞机协同设计系统中的应用.....53
6.1 飞机协同设计系统概述.....53
6.2 信息感知组件实现.....54
6.3 消息代理组件的实现.....55
6.3.1 配置 ACE/TAO.....55
6.3.2 配置 OpenDDS.....57
6.3.3 发布订阅过程实现.....57
6.4 适配器消息翻译模块实现.....60
6.5 系统运行测试.....61
6.6 总结.....64
6.7 后期研究工作.....64
参考文献 65
致 谢 70
在学期间的研究成果及发表的学术论文.....71

图表清单

图 1.1 论文各章节的组织结构图.....8
图 2.1 DDS 层次结构图 9
图 2.2 DDS 的体系结构.....11
图 2.3 DCPS 模块组成图.....12
图 2.4 Web 服务协议栈 15
图 2.5 SOAP 消息格式 15
图 2.6 消息中间件应用模式图.....16
图 3.1 飞机协同设计仿真应用场景图 20
图 3.2 ACD_DSM 框架图 21
图 3.3 总体流程图.....23
图 3.4 DDS 发布/订阅流程图 24
图 3.5 OpenDDS DCPS 实体关系图.....25
图 4.1 发布订阅原理图 28
图 4.2 主题类型文件定义.....29
图 4.3 基于 SDP 自动发现协议两阶段.....30
图 4.5 BFV 插入和查询样例.....32
图 4.6 SDPBloom 算法描述 33
图 4.7 Bloom Filter 原理图.....34
图 4.8 DCF 示意图.....34
图 4.9 DCFSPD 算法 36
图 4.10 DCFSDP 协议阶段描述.....37
图 4.11 DCFSDP 流程图.....38
图 4.12 IDL 定义描述.....40
图 4.13 实验 1 结果柱状图.....42
图 4.14 实验 2 结果.....43
图 4.15 OpenDDS QoS 配置示例.....43
图 4.16 QoS XML 配置文件.....44
图 5.1 传统应用适配器示意图.....46
图 5.2 应用适配器结构图.....46

图 5. 3 飞机协同设计消息流向.....47
图 5. 4 统一消息格式算法.....50
图 5. 5 XSL 示例.....52
图 6. 1 嵌入 ACD_DSM 的飞机协同设计系统示意图.....53
图 6. 2 Insert 触发器.....55
图 6. 3 config.h 描述.....56
图 6. 4 ACE/TAO 编译 bat 文件.....56
图 6. 5 XSL 文件.....60
图 6. 6 基础配置界面.....61
图 6. 7 主题配置界面.....62
图 6. 8 手动发布界面.....62
图 6. 9 监控界面（发布）.....63
图 6. 10 监控界面（订阅）.....63
表 4. 1 DDS 支持的 QoS.....35
表 4. 2 实验 1 场景描述及实验结果.....41
表 4. 3 实验 2 场景描述及实验结果.....42
表 6. 1 数据表 log 表结构.....54

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
RTPS	Real-time Publish-Subscribe	实时发布订阅
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	Counting Bloom Filter	可计算布隆过滤器
DCF	Dynamic Count Filters	动态可计数过滤器
DCFV	Dynamic Count Filters Vector	动态可计数过滤器向量
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

计算机网络技术尤其是互联网技术在最近几年得到飞速的进步，计算的资源分布范围逐渐宽广，互连水平也愈发提高。在地域位置方面，数量庞大的参与数据交换的实体分布在世界各个角落，其言行也将跟从时间的变化呈现出非静态的演变。更因为系统实体相互间缺乏一定的相关性，最终导致无数个信息孤岛出现。在飞机设计领域同样也出现了类似问题，该领域中的各单项技术取得了突飞猛进的发展，如动力学分析与设计、飞行力学分析与设计等，以上学科现在都已经拥有自己专用的设计与分析工具软件。然而，因为各个系统的信息交互能力相当薄弱，每一学科的专业软件系统就逐渐演变为相互排斥的信息孤岛。

文献[4]指出飞机设计是一项复杂的系统工程，其总体设计阶段需要进行大量的关键性决策，并决定了一架飞机全寿命周期大约 80%的成本。传统的飞机总体布置设计采用一种串行设计模式，即在前一个飞机研制单位或部门完成某项设计任务期间，其他单位或部门可能将无法从事任何工作；只有当前一个部门将设计结果移交给后面单位或部门时，他们才能进行下一步的设计工作。并且由于已有的各专业软件系统缺乏统一的数据标准与统一的管理方法，各学科的应用系统通常建立在不同的平台之上，不同的运行环境之中，而且一般采用不同的编程语言，各子系统信息交互困难，基本上是通过人工的方式（如发邮件）进行信息的交互。由此可以看出，这种串行开发模式的最大缺陷在于：不同设计开发部门之间缺乏交流，每个部门仅考虑其局部的功能性要求，没有从整体和全局视角进行综合权衡。由于交流不及时、协调不充分、步调不一致的原因，常常造成大跨度的设计修改或重新设计，导致设计周期长、成本高。而且设计人员工作空间分布、工作时间分散，由于网络环境下的群体协同工作支持力度不足，导致群体协作效率低、设计冲突多。

针对以上问题，需要充分发挥各单位或部门的优势，共享开发的资源和经验，多单位或部门能够联合协同设计。飞机协同设计[5]是一种跨地域、跨时间、跨学科的多功能小组协同工作方式,能够有效地支持分布式产品设计和开发。通过协同工作、交互协作、分工合作等方式使产品生命周期全过程人员都能够为产品的开发做出贡献。在文献[6]中指出协同设计的主要思想就是要求能够实现数据共享，从而确保在产品设计过程中各阶段能够同时进行。

飞机设计相关学科发展迅速，各个部门均有自己专业使用的专业软件系统，为了实现各部门之间可以协同设计，重新开发实现难度较大，因为飞机设计人员对计算机相关技术不熟悉，而计算机对飞机设计中的专业知识也不能理解。针对现状，最好的方案是在保持原有的个专业软件系统自治性的情况下，实现各软件系统之间的信息交互。本文设计并实现了基于 DDS 的飞

机协同设计中间件，实现了对现有的飞机设计各专业软件系统的信息共享和交互。本文提出的中间件使用数据分发服务[1-3]（Data Distribution Service for Real-time System，DDS）作为消息代理组件，并在研究分析开源项目 OpenDDS 的基础上，修改相关代码实现消息代理组件，实现各专业系统之间的通信，并选用选用基于 XML 的通用数据表示技术设计并实现了中间件的适配器组件，解决因现有的专业软件系统使用的平台、编码方式、数据库不同而导致的数据异构问题。最后，通过将飞机协同设计中间件应用到飞机协同设计系统中，进一步提高设计的研发进度，缩短研发周期，促进航空航天产业网络化、信息化和全球化，对实现航空工业跨越式发展具有重大意义。

1.2 国内外研究现状及发展趋势

本文需要解决的问题是如何使各专业软件能够实现信息共享，而各系统需要交互的是系统数据，因而可以理解为数据集成问题。本小节首先介绍了数据集成的发展，而基于分布式中间件的数据集成是近几年研究的热点，而本文采用基于 DDS 的数据服务中间件实现各专业软件系统之间信息共享，其后分析了数据分发服务的发展现状。

1.2.1 数据集成研究现状

20 世纪 80 年代，已经有人谈及多数据库系统[7] 这一全新的概念，对于这一概念的研究大体致力于运用传统的数据库技术来桥接异构数据源，其研究的侧重点在于联邦数据库系统[8]。所谓联邦数据库系统，即将全部数据源分布到一个单一的集成系统之中。该方式相对容易，因为集成系统具有统一的数据模式，因此无需思量关于分布数据的转化以及统一的问题。不可否认的是该方法仍然留有许多缺陷，具体表现为：其一，若要建构一个集中式的系统要求投入漫长的开发时间，并且它对主机的性能持有不低的要求，其实现代价相对过高；其二，系统的扩展和维护将波及到系统全局，不同集成系统之间难以实现各自模块间的相互分享。

数据仓库模式[9]所使用的数据集成方式与前文中提到的各种方法的迥异之处在于：数据仓库模式并非是逻辑形式的集成方式，它是经由 ETL[10] (Extract、transform and load，提取、传输和加载)工具来实现数据在物理上的变迁。所以，它通常被视作为物理数据集成方法。在集成系统当中，数据仓库存在于客户端与数据源间，其作用是存储源于各数据源中的待集成数据，系统负责供给数据仓库的查询机制。该体系结构的亮点与优势在于具有较快的查询响应速度，而它的问题也显而易见，如果数据源中的数据变化频度较高，想要在数据仓库中查阅最新的数据则必须定点及时地更新数据仓库，这将会带来极其昂贵的维护成本。

基于分布式中间件展开数据集成逐渐成为近几年学术研究的敏感点。众多国外的中间件厂商已经将数据集成产品作为竞争的焦点，他们通常以应用集成中间件为视角来研究数据集成，技术核心聚焦于 XML 消息总线机制、应用层路由以及应用适配器等方面。Iona，Tibco，

WebMethod，PcopleSoft，CA，SAP，GE 和 IBM，BEA 等主要的中间件厂商开始把数据集成和应用集成中间件视为技术以及市场发展的关键所在。

近几年，国内外中间件技术发展很快，许多公司均有自己的中间件。国外比较常见的中间件技术有 OMG 组织的 CORBA[11](公共对象请求代理架构)、JavaRMI[12]、Java 消息服务(JMS)[13]、IBM 公司的 Web Sphere MQ[14]、Web services[15]。国内常见的中间件有北京中科国际软件的 A2E-DataIntegrator[16]、东方通科技的 TongIntegrator[17]和中创软件公司的 Inforbroker[18]等。

1.2.2 数据分发服务研究现状

数据分发服务[1-3]作为数据分发体系的标准，与 CORBA、JMS 技术相比较，针对以数据为中心的数据分发体系的研究起步较晚。在 2004 年之前，数据分发的产品在实时性、快捷性和灵活性等方面表现不是很好。在文献[12]指出，目前阶段，作为 OMG 组织所拟定的标准之一的 CORBA 技术，它是一种将对象或服务作为中心的主要针对对象应用程序的体系规则，采用了客户机/服务器通信模式，通信机制相对复杂且存在服务器瓶颈等问题，能较好地适用于对实时性要求较低的分布式处理。JMS 技术采用了发布/订阅通信模式，但是缺乏应用级 QoS 策略以及仅能够支持 java 语言，同样不适合紧急任务或实时性要求高的系统。

国际对象管理组织正是意识到当前一些数据分发技术存在很多缺陷或不足，根据美国国防部信息技术标准注册中心委托，召集并组织了大量具有丰富经验的底层技术人员开展研究工作，于 2004 年 12 月发布了面向分布式实时系统的数据发布服务标准 1.0 版本[1]。此后，OMG 组织还先后在 2005 年、2007 年颁布了 DDS 标准 1.1[2]和 1.2 版本[3]，每个版本的颁布都为设计出更好的 DDS 中间件产品提供指南。我国在 2004 年也颁布了一个数据分发服务指南与规范[19]，为基于网络的分布式数据分发服务系统的建立提供了一种规范，其内容主要包括科学共享数据分发的过程及各过程的内容和需要遵循的标准。

目前，国外已经有比较成熟的 DDS 产品，主要有以下几种：

- RTI DDS[20][21][21]：这是由 RTI 公司推出的，也是第一个支持 DDS 规范的商业产品（前身叫做 NDDS[22]）。RTI 公司是 DDS 市场的领导公司，它实现了完整的 DCPS 和部分的 DLRL。
- OpenSplice DDS[23]：该产品最初是由 Thales 开发的，在 2006 年 PrismTech 公司获得，它实现了完整的 DDS，即包括 DCPS 和 DLRL。
- OpenDDS[24]：由 Object Computing，Inc 公司就 DDS 规范 1.0 版本，通过使用 C++语言来完成的对象管理组织数据分发服务，并且源码开放。OpenDDS 将自适应通信环境 ACE/TAO 作为实现跨平台的环境，用户在使用 OpenDDS 前，只需要对 ACE/TAO 进行简单的环境设置和编译即可，它主要实现的是 DCPS 部分。

- Java-based DDS[25]：这是 Universite Grenoble(France)教授 Didier Donsez 根据教学的目的需求，用 Java 语言开发实现的一个 DDS。它主要基于 ORB 所提供的事件服务来实现数据分发，并在 Jacorb 上测试通过。
- MilSOFT DDS[26]：由土耳其 MilSOFT 公司根据对象管理组织提出的 DDS 标准 1.2 版本开发实现的，可支持多种语言，但目前占用的市场较小。
- CoreDX[27][28]：由 Twin Oaks Computing, Inc 公司用 C 开发实现的，它是作为一种专门为现代嵌入式系统而设计的数据分发服务标准兼容实现体，CoreDX 更加专注于通信性能的提高、先进网络协议的利用、以及吞吐量的提高和服务延迟的降低。

目前国外仍有很多学者正在对 DDS 规范以及这些成熟的 DDS 产品进行多方面的研究，其中主要有针对 DDS 自动发现机制进行了大量的研究，如文献[29]中指出标准 DDS 自动发现机制需要用户事先静态预配置网络中的各节点信息，不利于应用于规模宏大的网络之中；为此，针对大规模动态网络，他们提出了一种自适应动态发现框架思想，主要是从实现以下三个目标出发：1）将 DDS 实体和共享数据有效地进行联系；2）支持明确通信提示；3）与标准无缝集成。文献[30]对 DDS 的自动发现机制进行了研究与分析并同样指出其只能适用于中小规模的网络，不利于系统的扩展，因此从系统可扩展性的角度提出一种基于 Bloom Filter[31]自动发现算法 SDPBloom，该算法能够更加有效的解决标准 DDS 自动发现机制存在的高内存消耗以及高网络通信量等问题，增强了 DDS 的可拓展性等方面。文献[32]指出 SDPBloom 算法虽然能减少内存消耗，但是 Bloom Filter 本身易于出现误报率的弊端；因此，Handityo Aulia Putra 等人在此基础上提出基于改进 CBF（Counting Bloom Filters）[33]的自动发现算法，将 Bloom 与 HashTable 结合，有效的消除了使用过滤器带来的误报率，但另一方面，由于 HashTable 携带了所有端点的部分信息，网络传输量也很大，并且仍没有处理 SDPBloom 算法中遗留的 QoS 的问题。南京航空航天大学后来提出了基于服务力向量的发布订阅自动发现算法[34]，一定程度上处理了因 QoS 不兼容带来的问题。

此外，DDS 作为一个数据分发标准已被市场快速的接受采纳，尤其在实时分布式系统中被广泛应用。在国外，DDS 被应用于美国海军国防部的开放结构计算环境 OACE（Open Architecture Computing Environment）以及舰自卫系统 SSDP（Ship Self-Defense System）中，这两个系统应用都是美军未来战斗系统的一部分[35]。商用市场方面，以美国 RTI 和法国 THALES 为首的公司已经开发了以 DDS 标准为研究出发点的数据分发中间件平台产品，并被普遍地运用于通信、军事、航空航天、工业自动化控制、智能交通、以及金融等多种领域。

而在国内，目前我国对分布式实时通信环境下数据分发服务的研究仍处于起步阶段，尤其在以设计相关中间件平台为目标的研究并未深入开展[36]。文献[37]指出了国内对 DDS 的研究主要是针对现有的消息中间件进行二次开发，目前为止国内也很少有成熟的 DDS 产品。但在军事

领域方面，文献[38]指出了我军在战术信息分发服务技术邻域已展开了多年的研究，尤其是在信息检索和信息发布与订阅、以及信息智能推送等技术方面做了深入的研究，也取得了一些科研成果。在应用研究领域方面，近年来我国学者也对 DDS 规范展开了大量的研究，如：文献[39]对 DDS 规范进行了研究并设计了一种基于 ACE 的以数据为中心的分布式实时数据分发服务中间件；南京航空航天大学谷青范教授在文献[40]中针对现有 DDS 中间件的动态服务发现和容错机制做了一些分析和改进，将目前发布/订阅模型中信息的集中存储结构设计成 P2P 分布式结构，并利用 chord 协议[41]动态实现发布与订阅主题数据的匹配，从而实现中间件的自适应能力。

1.2.3 研究发展趋势

基于分布式中间件是数据集成这几年的发展趋势，而现有的中间件技术大部分为商用，并且在 QoS 策略支持和开发语言、平台商有比较苛刻的要求。而 DDS 以数据为中心，为数据分发提供了一种互操作标准，其具有丰富的 QoS 策略和自动发现机制等优点，保障了系统的非功能型需求。

另一方面，相较于以往的通信机制，发布/订阅通信模型具备异步、多点通信的特征，这些特征确保通信参与者在时空和控制流上彻底解耦，较好地适应了大型分布式系统松耦合通信的要求。发布/订阅系统现在获得了各界的关注，而且拥有很好的发展前景。DDS 能够支持系统中用户的动态添加与删除，从而增强系统的灵活性和可扩展性。

综上所述，DDS 提供了一种如何在订阅者与发布者之间建立可靠的、实时的、自适应的高性能数据交互的实现方式，DDS 实体通信采用的是发布/订阅通信模式，该模式在时间、空间和同步关系方面可完全解耦，并且具有丰富的 QoS 策略支持。因而在数据集成和航空航天领域中有很好的发展前景。

1.3 论文主要工作

本文针对飞机设计过程中，各学科协同设计时信息交互与共享困难的问题，设计了基于 DDS 的飞机协同设计数据服务中间件。论文工作首先认真分析了对象管理组织 OMG 制定的 DDS 标准，并深入研究了 OCI 开源项目 OpenDDS，结合飞机协同设计需求，改进了自动发现机制，并将其应用到作为数据服务中间件的消息代理组件。其次，由于各学科的专业软件系统使用的平台不同、编码语言不同、数据库等不同，运用 XML 相关技术，设计了中间件的适配器，从而解决了数据异构的问题。最后将研究的中间件应用到飞机协同设计系统中，初步实现了各专业软件系统的信息交互和订阅发布。本文具体工作如下：

1) 数据分发服务技术的研究。主要研究了 DDS 相关标准、体系结构以及模块划分；并深入分析了 DDS 的开源实现 OpenDDS 的全部代码，并将其改进后应用到数据服务架构中。

2) 飞机协同设计数据服务中间件的设计。本文结合飞机协同设计的需求，设计出具有可扩

展性、松耦合的数据集成框架。秉持灵活以及可扩展的原则，发挥 DDS 的特性，构建了基于 DDS 的飞机协同设计数据服务中间件框架。

3) 发布/订阅自动发现机制的研究以及改进。论文工作研究了 DDS 规范中基于 SDP 协议的发现协议、基于 Bloom Filter 改进的自动发现协议 SDPBloom，提出了一种基于 DCF 的自动发现协议,设计了相关算法，并对算法进行了性能分析和实验验证，实验结果表明，设计的局域 DCF 的数据分发自动发现协议有效地减少了网络负载和内存消耗。

4) 采用 XML 相关技术设计并实现了中间件的适配器组件部分，解决了现有的各专业软件系统的数据格式不统一的问题。

5) 基于 DDS 的飞机协同设计数据服务中间件的应用。本文设计的基于 DDS 的飞机协同设计数据服务中间件已在飞机协同设计系统中得到初步应用。测试实验结果表明该数据服务中间件能够满足飞机设计各专业软件信息交互的要求;还通过实验验证了数据分发服务 DDS 应用到飞机协同设计中的有效性。

1.4 论文组织结构

本文在数据分发服务模型分析研究的基础上，将飞机协同设计视如研究对象，构建并达成了基于 DDS 的飞机协同设计数据服务中间件，并将其运用到飞机协同设计系统当中。全文共分为七章，论文组织结构如图 1.1 所示：

第一章：阐明文章的研究背景以及意义，介绍有关与主题相关的国内外研究现状以及未来发展趋势，且着重说明本文所做的研究工作，最后列出本文的组织结构框架。

第二章：介绍了数据分发服务的相关标准和体系结构特征，包括一些主要的基本概念、模块划分以及针对 DDS 的特点进行了分析；然后分析了当今主流的两个信息共享的技术方式，并给出两种方式的不足之处，为后面章节做铺垫。

第三章：本章第一小节首先进行了需求分析，给出了飞机协同设计的特点、应用场景和功能需求；在第二小节中给出了基于 DDS 的飞机协同设计中间件的总体设计；第三小节分析了开源项目 OpenDDS，并给出了不足之处；最后一小节给出了信息感知组件部分的设计思路。

第四章：本章第一小节首先介绍了发布订阅通信模式，其后三小节在 OpenDDS 的基础上，完成 ACD_DSM 的主题文件的设计、发布订阅自动发现协议的改进和 QoS 控制方法改进。本章重点是对自动发现机制的研究，针对数据分发服务中现有的自动发现机制的不足，对 DDS 的自动发现机制进行改进，给出了基于 DCF 的自动发现协议，有效的降低了网络负载。

第五章：本章首先对适配器的功能与分类做了简要介绍，其中对传统应用适配器做了分析，然后结合本文的需求，设计了数据服务中间件的应用适配器。最后介绍应用适配器中的三个核心技术即统一的消息格式生成，消息解析、消息翻译。

第六章：本章首先对飞机协同设计系统进行概述，数据服务中间件嵌入到各专业软件系统中，从而实现各专业软件系统的信息交互；然后对信息感知组件，消息代理组件，适配器组件详细设计部分作简要介绍，最后给出飞机协同设计系统的运行实例，并给出了本文总结以及展望。

zkq 20151222

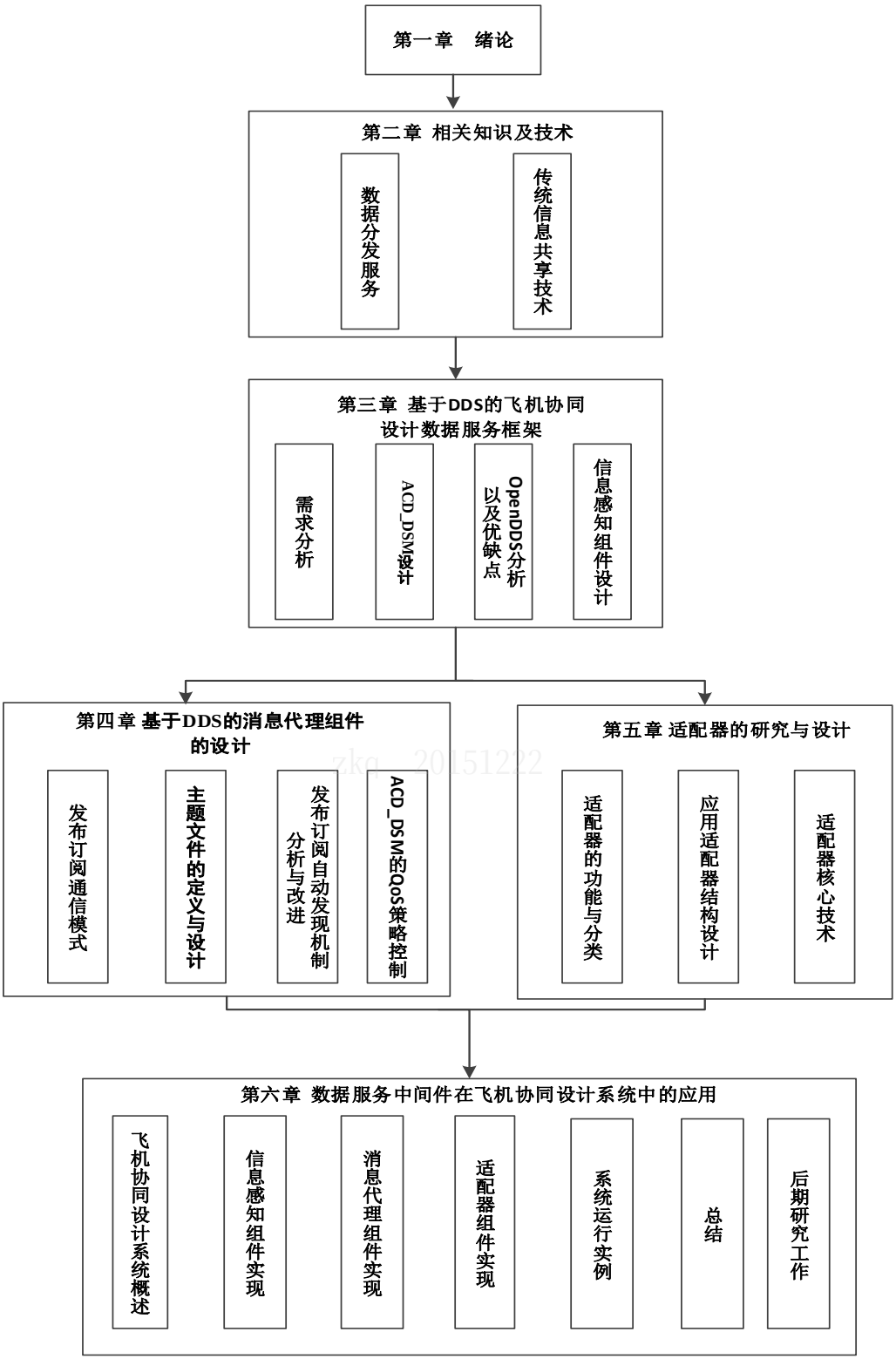

图 1.1 论文各章节的组织结构图

第二章 相关知识及技术

本文的目标是实现能让现有的飞机设计软件系统之间信息共享，采用了基于 DDS 的飞机协同设计数据服务中间件完成。本章首先对数据分发服务 DDS 做了相关介绍；然后分析了当今主流的两个信息共享的技术方式，并给出两种方式的不足之处。

2.1 数据分发服务（DDS）

数据分发服务（DDS）以向分布式系统提供可靠、高效、实时的数据分发服务为基本目标。在 DDS 标准发布以前，由于缺乏一种互操作性标准，使得随着信息交互规模的变大以及交互类型的复杂化，不同地域、不同平台之间的数据交互变得尤为复杂，无法满足系统的发展需求。此外，对于实时性要求比较高的系统，更是难以满足。对象管理组织（OMG）从 2000 年开始构建一种新的互操作标准，并在 2004 年 12 月制定出第一个数据分发服务标准，该规范明确定义了一种实现框架和用户访问接口，以及多达 22 种 QoS 策略来满足不同用户的需求和提高系统灵活性。该规范将 DDS 标准划分成数据本地重构层 DLRL (Data Local Reconstruction Layer) 和以数据为中心的发布-订阅层 DCPS（Data-Centric Publish-Subscribe)[42]。DDS 的基础和核心即为 DCPS 层，它主要实现的功能是将合适的信息有效发送到合适的接收者。而创建在 DCPS 之上的 DLRL 层，它主要实现的功能是抽象 DCPS 层所提供的服务，同时与底层服务保持一种互为映射的关系，职责在于把简单的服务集成到应用层。DDS 层次结构如图 2. 1[1]所示。

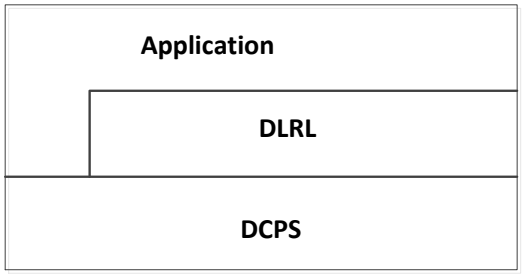

图 2. 1 DDS 层次结构图

2.1.1 DDS 的相关概念

在数据分发服务规范里引入了一些全新相关实体的概念，主要是指域参与者、发布者、订阅者、数据写入者、数据读取者、主题等等，下面文章将对这些概念进行阐释。

- **域 (Domian)：**所谓域实质上指的是一种基本结构，它能够满足许多应用者之间互相通信的需求，的数据写入者和读取者只要满足运用相同的数据类型这一前提，便能够在相同的域中实现互相通信。而多域指的是这样一种情形，即假使信息分发中间件能够同时支持

大于单一个域的特性，多域最大优势为它能够谨遵系统提出的需求实现扩展灵活进而满足各种要求，而数据类型相异那么所创建的域也不尽相同。假如我们将某个指定的数据例程创建于某一域中，那么这个数据例程将无法被该特定域以外的应用其他任何域的使用者所获得。多域的数据隔离功用效果突出，举例来说，在某一域中实现交换控制和指挥数据效用的同时，多域还能够在除此以外的其他域中实现有关用户状态信息的互换工作。另外，多域的吸纳性优越，便于在现有的系统当中纳入全新的功能，例如将某一项新的功能添加到测试显示正常的分布式环境中，如果想要降低新功能对原来环境中现存的功能的制约与影响，多域则为一种理想的选择方式。

- **发布者 (Pulisher)：**也就是指主题提供方。一个发布者能够发布许多不同的主题，其主要职责在于创建以及管理数据写入者。
- **数据写入者 (Data Writter)：**主要负责将已经发布的主题信息登记到发布者-主题表中，同时将主题中的数据放置到数据存储区中。一旦发布方与订阅方的需求相吻合，那么数据通信就会被创建，数据写入者将被激活。
- **订阅者 (Subscriber)：**指的是主题请求方。当某节点需要订阅主题信息时，订阅者就会自动被创建。订阅者能够订阅诸多不同的主题，负责创建以及管理数据读取者。
- **数据读取者 (Data Reader)：**主要负责读取域中符合自身订阅需求的主题数据信息。数据读取者的激活依赖于订阅方和发布方成功建立起数据通信。
- **服务质量 (Quality of Service)：**服务质量为 DDS 极其紧要的一部分，也是 DDS 中最大的亮点。每一个主题都有一个相应的服务质量参数，如持久性、优先性、可靠性等等。质量服务 QoS 策略控制能够通过 XML 配置文件方式实现，也可以直接借助编码的方式达成。DDS 中，QoS 能够确保在正确的时间内将正确的数据传递到正确的数据请求者处（3R 原则）。
- **全局数据空间（Global data space）：**这是整个 DDS 服务模型的核心部分。此空间是由几张数据表构成，主要为发布方和订阅方的数据通信供应信息依据。应用 GDS 管理策略的作用在于，能够及时更新数据表进而确保数据一致性。
- **主题 (Topic)：**主题的使用对象主要是信息发布者，其基本作用在于一旦确定或者界定了主题设定数据的具体发布对象，那么介于为主题提供数据的发布者以及相应的订阅者的连接便被建立起来，它主要由主题的类别、主题的名称（通常是仅有不可复制的标识符）以及数据的具体服务质量参数这三个最基本的要素所构成。主题类型主要是限定主题当中所运用的数据的具体类别，而主题名即域当中独一无二地可以用来标记主题的系列字符串，某一域中的主题只能被赋予唯一的定义并且这种规则适用于任何域。数据订阅者一般而言是依照主题名来建立数据订阅实体的，接着再将向数据订阅者进行数据发送的具体方式告

知于数据分发中间件的发布者。一旦应用要求发布某一特定的数据类型具体信息，那它务必事先定好数据发布者以及数据写入者，通常这两者需要应用根据信息需求的类型来进行创建。类似地，如果应用层提出接收数据的请求，那么创建数据订阅者以及数据读取者就是及时之需。某通信节点所指定发布者的主题必须可以适应此外的节点相应订阅者所创建主题，也就是说这两者只有在满足相互匹配的前提之下，通信才可以被建立，否则，不同节点想进行互相通信的目标就难以实现。

数据分发服务体系的结构如图 2.2[42]所示。仅当满足处于同一域中的前提条件，这样域中的实体能够实现相互间通信，每个域中可以有一个或多个数据读取者和数据写入者，同一个数据写入者或读取者也可以同时加入一个或多个不同域。数据读取者和数据写入者之间是通过主题互相关联的，只有具有相同类型主题的数据读取者和数据写入者才会建立通信连接。

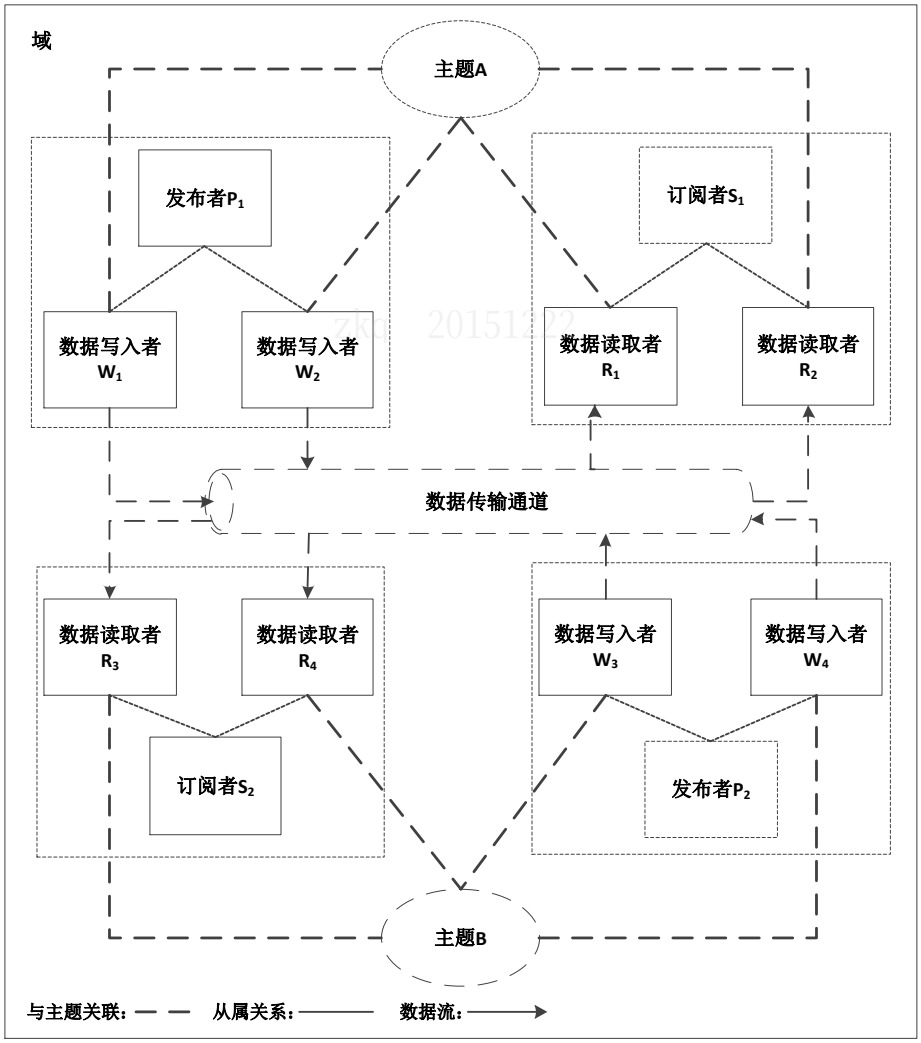

图 2.2 DDS 的体系结构

2.1.2 DCPS 的模块划分

数据分发服务规范将核心层 DCPS 划分为五大模块，即域模块、发布模块、订阅模块、主题定义模块以及基础架构模块，如图 2. 3[1-3]所示：

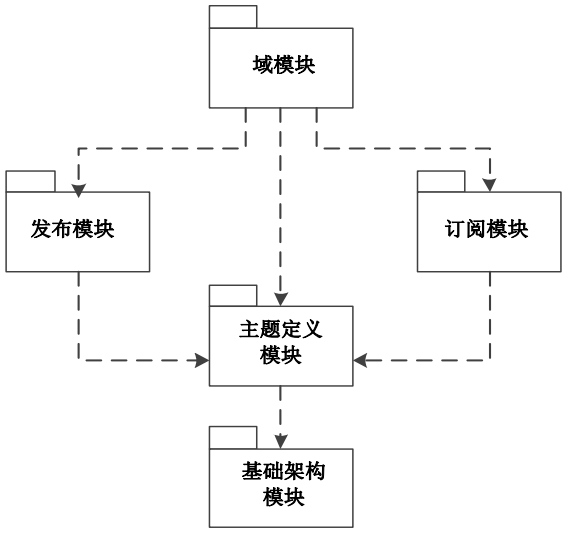

图 2. 3 DCPS 模块组成图

1) 基础设施模块（Infrastructure Module）：负责定义抽象的类和接口，这些类和接口将由通过其它的模块继承实现。该模块为中间件提供了不同的交互方式类型：基于通知（notification-based）和基于等待（wait-based）。
2) 域模块（Domain Module）：域模块包括域参与者（DomainParticipant）类。域参与者是服务的入口点和产生其他实体类的类工厂，它同样也是组成服务对象的容器。
3) 主题定义模块（Topic-Definition Module）：主题定义模块包括了主题（Topic）、内容过滤主题（ContentFilteredTopic）和 多主题（MultiTopic）类以及主题监听器（TopicListener） 接口。应用系统为了能够对主题对象进行定义、发布或接收以及 QoS 的检查必须实现这些类和接口。
4) 发布模块（Publication Module）：发布模块包括了发布者和数据写入者类以及发布者监听器（PublisherListener）和数据写入者监听器（DataWriterListener）接口。这些类和接口是发布方所必须的。
5) 订阅模块（Subscription Module）：订阅模块包括订阅者、数据读取者、读条件、查询条件类以及订阅者监听器（SubscriberListener）和 数据读取者监听（DataReaderListener）接口。这些类和接口是订阅方所必须的。

2.1.3 DDS QoS 策略控制

所谓的服务质量策略其实指的就是一些特性的组合，这些特性的作用体现在对数据分发服

务的控制上，数据分发服务的实现无法离开应用相应的服务质量策略。处于系统里的全部实体均具备相应的服务质量策略，主要涵盖了主题、数据写入者、数据读取者、发布者、订阅者以及域参与者等等。

DDS 标准实现了对系列 QoS 策略的界定，而这些 QoS 策略的基本功能为当相关的应用程序在应用过程当中能够自行界定可靠性、容错资源使用、等基于服务所提出的需求。在本论文当中介绍的是部分 DDS 标准的较为有份量的 QoS 策略。

1. 截止日期（DEADLINE）

截止日期与主题、数据记录者以及数据读取者这三个要素相互关系紧密，截止日期确定的是数据记录者在发送数据时所具备的最小速率。此外，截止日期还界定了数据读取者在接收新的数据时所预期的最大的等候时长。倘若主题希望可以按期实现对所有例程的革新，那么久需要设定具体翔实的截止日期参数。这就是说，于发布方而言必须设定一个数值，而这个值的设置并非随性而定，必须将当下应用的实际能力作为权衡的标准；而于订阅方来说必须设定一个对应的需求数值，而这个数值只能由需求的发布方来供给。一旦数据记录者和数据读取者的服务之间配对成功，那么服务将进一步检查相应的设置是否相互适应，例如检测其是否符合:提供截止时间<=请求截止时间，一种结果是无法符合上述条件，那么就将通过运用监听机制或者是运用条件机制等方式将此次匹配的“通信双方的 QoS 设置无法实现匹配，双方互相通信无法创建”这一失败结果传达到双方对象实体处；另外一种结果是符合上述条件即提供截止时间<=请求截止时间，此时就能够实现双方互相通信的创建。

其主要的功能简述如下：确保所有数据记录者能够参与制定发布数据的具体速率以及结束数据的具体速率、指示何时指定数据源不可用、将是否到达截止日期的消息及时反馈到应用层。

2. 时长（DURABILITY）

它所判定的是 DDS 是否实现了对属于历史数据的取样值的保留，它能够为后来加入的读取者供应历史取样值，满足他们使用历史数据的需求。时长包含了以下几方面的设置：

- 瞬间保存：主要指系统并不保存任何的历史数据的取样值，此状态无需人为设定，它属于默认的设置。
- 临时状态：即系统仅仅保存少数的历史取样值，所保存取样值的量的大小则由对应的系统资源总量的大小正相关。
- 永久状态：就是说系统将历史取样值的保存位置设定为硬盘，处于该状态之下用户能够根据自己的需求或是时间规制加入网络，同时能够获取系统在用户加入该网络之前卞发布的历史数据。

3. 可靠性（RELIABILITY）

可靠性策略主要界定了由数据读取者所请求的以及由数据记录者所提供的两个可靠性的

具体级别。这个级别应当是有序的，通常而言主要包括下面的类型也即尽力而为(BEST_EFFORT)类型以及可靠(RELIABLE) 类型。

如果可靠性类型属于尽力而为型，那么就不会产生数据重传的现象，因为该类型通常适用这样的情景即存在的是对最新数据十分关心与介意的用户。相反，如果属于可靠型，那么数据记录者就会将特定量的发送数据保存到已发送的数据当中，这样做的目的在于便于后期展开数据的重传工作。在完成出对某数据的发送任务之后，数据记录者将会从已发送数据的列表当中将此数据删除，删除的前提是他接受到来自数据读取者所反馈的确认接收的信息。

4. 历史记录（HISTORY）

即数目，它指的是指定由中间件保存的数据采样的具体数量。我们能够将其设定成“保存前 N 个”，这里“N”就是所要保存的数据采样的详细数目是 N，通常而言，数值“1”是 N 的系统默认设置值。

假设某应用的数据存在于某一持续变化的过程里，但凡外界有新应用请求加入该域，那么域中相应的特定主题的使用者就将会得到一定量的历史采样数据。某些情况下，这一 Qos 策略相对管用，例如当数据用户对某应用提出获取其历史采样数据的需求时，历史记录策略将是最为有效的策略。另外，它还具有自定义的属性，用户能够根据自己的需要界定历史记录的具体数值，或许一个数据记录者可能只要求在历史记录中保存最新近的 5 个数据值，而另外一个数据记录者则可能需要保存 10 个甚至更多的数据值，运用该策略就极好地能够实现不同记录者的不同需求。

2.2 传统信息共享技术

目前已有的异构系统一般是通过中间件和 web 服务的方式来实现信息的共享。本小节对这两项技术进行相应的介绍，然后给出现有的技术存在的不足点。

2.2.1 Web 服务技术

应用 Web 服务[52]技术，应用程序能够通过无关于编程语言和平台实现相互通信。Web 服务主要是运用 URI 来完成对自身标识的一套软件系统，它的接口以及绑定都需要借助 XML 来完成定义以及描述，此外，它还支持直接交互的实现。

Web 服务是建立在一系列相对开放的 Internet 标准之上， 其基本标准主要包括 XML[45]、SOAP [49](Simple Object Access Protocol)、WSDL[50](Web Service Description Language)和 UDDI[51](Universal Description ，Discovery and Integration)。实质上，它是由协议栈按照一套某种规则来构建的所构成的层次化体系结构，如图 2. 4 所示。

Orchestration Language	BPEL	QoS	Management	Security
Service Discovery,Integration	UDDI			
Service Description	WSDL			
Messaging	SOAP			
Application	HTTP,SMTP			
Network&Transport	TCP/IP			

图 2.4 Web 服务协议栈

SOAP 被应用于实现 Web 服务，它实质上是一种被简单界定了的基于 XML 的非重量级的协议，能够在相对松散的分布环境当中完成结构化、类型化的信息的互换。它仅界定了一种能够模块化的封装机制以及应用定义的数据来开展编码的机制。SOAP 所用的是如今被世人所普遍应用的 HTTP 协议以及 XML。SOAP 信封是 SOAP 消息的根元素，另外在 SOAP 信封中还包括两个子元素：分别是 SOAP 头部和 SOAP 主体。具体的 SOAP 的消息格式如下图 2.5 所示。

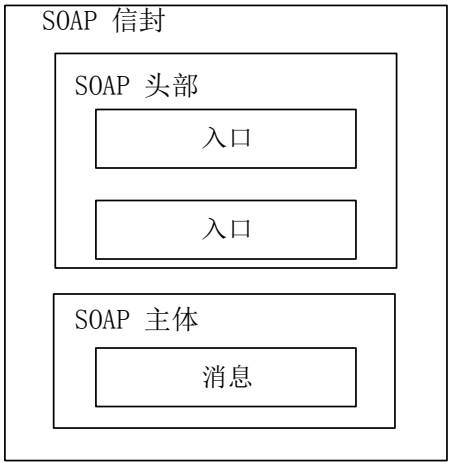

图 2.5 SOAP 消息格式

WSDL 是一种基于 XML 格式描述的有关 Web 服务的语言。基本目的就是将 Web 服务提供者的全部 Web 服务的内容形成一个系统的文档，发布给具体的服务请求者。而服务请求者则根据这个 WSDL 文档来建立相对应的 SOAP 请求消息，并且通过 HTTP 协议传送给 Web 服务的实际提供者；当服务提供者接到即时的请求消息时，便开始操作对应的 Web 服务。当执行服务实现以后，便把 SOAP 应答消息反馈给服务的请求方，随后服务请求方根据 WSDL 文件将其自行拆解成能为自己理解的文档信息。

2.2.2 传统消息中间件技术

中间件属于基础软件，它的定义一直模糊不清，众多定义中，被人们广泛接受的是国际数据公司 IDC 的定义，具体描述为：它为一种相对独立的系统软件或者服务程序，通过这种软件能实现分布式应用软件在各种的技术之间的资源共享 [53]。中间件主要应用在客户机/服务器的框架之中，主要管理计算资源和网络通讯。

中间件是处于分布式应用程序和操作系统(包含底层的通信协议)之间的一个软件层。其主要功能是创建不同的分布式软件模块之间互相操作的机制，从而在分布式环境中避免底层出现的异构性和复杂性等问题。另外，它为上层提供具体的运行与开发环境，以满足用户开展灵活、高效地开发和集成复杂的应用软件的需求。基于应用程序接口使其具有较为强大的通信能力和较好的扩展性[54]。

消息中间件(Message-Oriented Middleware，MOM)主要通过高效可靠地消息传达机制来开展与平台不相干的信息交互，并且以数据通信为基准来开展分布式系统集成[55]，是一种比较特殊的中间件。它实现了对操作系统、硬件、网络协议以及数据库在内的异构性消弭和隐蔽，提供给上层应用系统可靠的数据交换、并且具有优良的移植性和高度可复用性等优点。工作的具体流程为：客户端产生消息，并将该消息传送给消息中间件，经由路由、过滤等相关的技术处置，由消息中间件同步或者异步地把处理后消息将会转发到相应的接收者处。消息中间件的应用模式图如图 2.6 所示。

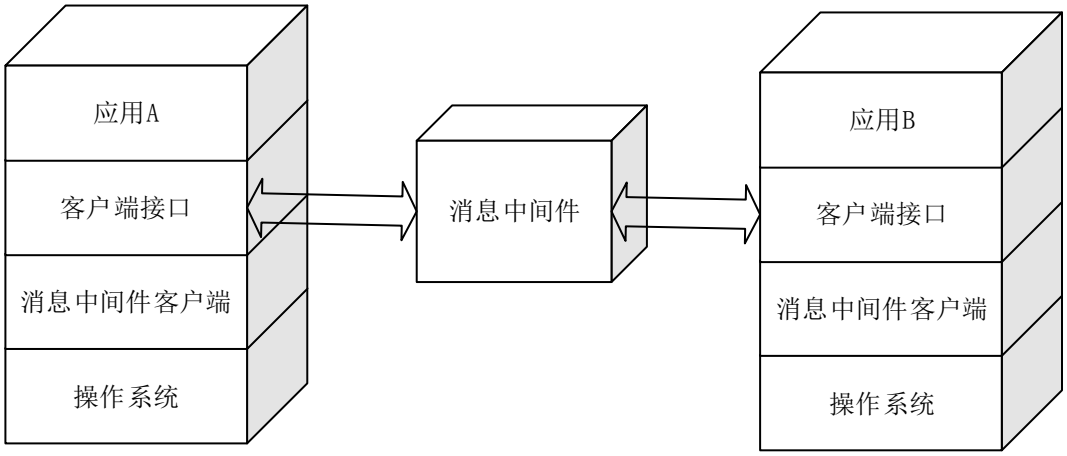

图 2.6 消息中间件应用模式图

MOM 的基础是消息通信机制，适合应用在事件驱动的应用[56]。MOM 编程所用的数据库为 MOM 的接口函数库，这个库能够较好地延伸到各类硬件平台以及操作系统上。其核心被装备在需要实施消息传递的系统之上，主要借助它们之间建立的特定的逻辑通道，最终完成消息的发送。同时还可以满足对企业数据进行异步传输的要求，一些原本相互独立的业务组件通过 MOM 就能够组成一个较为灵活、可靠的系统。

由以上的描述可知，消息中间件具有数据传输的方式多样化、有同步和异步两种传输方式、支持持久化和非持久化消息、支持内部消息处理、屏蔽开发细节，降低系统开发难度等优点。

2.2.3 现有信息共享技术的缺点

传统的消息中间件提供给消息通信一个比较好的底层通信框架，为企业级别的开发提供了强大的消息通信平台[85]。但还存在以下缺点：

1) 传统的消息中间件主要使用在局域网环境中，但在广域网中性能就不及 Web 服务。

2) 传统消息中间件的体系结构是基于远程过程调用，耦合度比较高，不一扩展。Web 服务是面向服务的体系结构，易扩展。

3) 传统消息中间件的调用方式较为固定，用户可支配性差。SOA 的体系结构总的来说是松散耦合的，可以很好的完成服务定制。

而对于 Web 服务技术，它的最大优点就是各组件之间具有比较好的互操作性，但也存在一些不足之处。

1) HTTP 协议的同步方式是基于请求/应答的，通信双方是紧耦合的；

2) HTTP 协议是无状态的，如果发生现错误，无法回到原本的状态，缺少事务特性。

3) HTTP 是以 BESTEFFORT 方式传输数据的,这是不可靠的通信方式。

4) 信息传输能力不及传统消息中间件。

2.3 本章小结

本章介绍了数据分发服务的相关标准和体系结构特征，包括一些主要的基本概念、模块划分以及针对 DDS 的特点进行了分析；第二小节对当前主流的信息交互方式，然后给出两种方式存在的不足，为后面选择 DDS 作为消息代理组件做铺垫。

第三章 基于 DDS 的飞机协同设计数据服务框架设计

面向飞机协同设计的数据服务中间件的主要目标是能够使现有的各专业软件系统之间可以信息共享，各专业软件系统能协同完成飞机设计任务。本章将从飞机协同设计的需求入手，本着在网络环境下通用、灵活、易扩展的原则，利用发布订阅通信模型的灵活性、松耦合性和易扩展性，设计了基于 DDS 的飞机协同设计数据服务中间件（ACD_DSM）框架。本文选用 OpenDDS 作为消息代理组件的开发基础，因而随后分析了 OpenDDS，并给出了不足之处，为下文的改进作铺垫；本章最后给出消息感知组件的设计。

3.1 需求分析

3.1.1 飞机协同设计特点

协同设计的思想是要求数据共享，即在设计过程中，各阶段可以同时进行。每个阶段生成需要的数据，虽然在没有完成设计之前数据是不完整的，但是通过数据模型和数据管理达到数据共享协同设计的目的。尽管协同设计在不同应用领域(或为了不同的协作目的)所构建的系统有所差异，但通常具有以下共同的特点[5]：

1) 分布性。参加协同设计的人员可能同地，也可能异地，通过网络进行通讯联系。因而协同设计是在网络的支持下分布进行的。

2) 交互性。协同设计产品开发过程的实施有多学科工作组负责组织和完成。在整个过程中，不仅涉及同一领域内的信息交互，而且还存在不同领域内的信息交互，这种信息交互方式不仅是同步的。

3) 动态性。设计过程的建立是一个动态的过程。在整个协同产品开发过程中，产品开发的进度、工作人员的任务安排、设备的状况等均在发生动态变化。为了使设计开发过程能够顺利进行，开发人员需要方便地获取各方面的动态信息。

4) 并发性和一致性。系统允许用户同时操作，需要维护数据的一致性。

5) 面向任务具有时效性。一个任务，多个用户，而且任务完成群组就解体，对于新的任务重新组建新的协同设计群组。

6) 协作和冲突性。产品开发过程存在着相互支持和约束及资源冲突。每个设计人员在开发过程中具有不同的设计任务，而为了更快、更好地设计出产品，设计人员要不断地进行密切的合作。

飞机协同设计是利用协同设计的思想研制和管理飞机的综合性工程技术。飞机设计是一个反复迭代、逐次逼近的过程,包括飞机总体设计和飞机分系统设计。飞机协同设计的主要特点如

下所示：

1) 团队合作。飞机设计是以多学科知识!新的科研成果、先进的制造工艺和试验手段作为设计的基础，依靠各种专业技术和系统来完成。飞机机体本身专业技术方面主要有空气动力学、结构力学、材料学、制造工艺等；要使飞机能飞行并完成规定任务，还应有动力装置、飞行控制系统、液压系统、电源系统、空调系统、燃油系统、救生系统、航空电子系统以及武器和火力控制系统等；同时为了保证飞机正常使用，还要有一套地面保障设备，包括随机设备、定检设备和场站设备等。如此广泛的多学科知识是个人无法掌握的，这就需要发扬团队精神，定期组织讨论，畅所欲言，对设计进行研究，帮助设计人员得出最佳化设计，各团队要经常向各方面的专家咨询，专家要及时对设计结果进行审核。

2) 设计过程的协同化。飞机设计要用系统论和系统工程的方法进行综合设计，其中总体设计尤为重要，飞机总体设计的任务就是始终坚持飞机系统整体功能和整体优化的原则，把不同的专业技术和系统创造性地综合在一起，这需要各分系统设计人员在设计过程中，要经常交流，把各自设计结果提供给总体设计人员和其他分系统的设计人员，保证设计结果的及时有效，其他人员也要及时反馈自己的意见,保证总体设计与各分系统设计要相互协调,综合权衡。

3) 缩短研制周期，降低成本。飞机的研制和开发的费用是非常昂贵的，要特别注意降低成本和缩短研制周期，飞机协同设计是针对传统设计的弊端,尽可能使设计一次成功，减少设计环节开发费用。

3.1.2 飞机协同设计应用场景

图 3.1 给出了飞机协同设计的各部门协作的仿真应用场景图（仅给出 3 个部门之间信息交互）。假定部门 A、部门 B、部门 C 需要及时的交互信息从而完成某一机型的设计，且需要满足以下需求：

1） 部门 A 负责飞机总体设计研制过程；

2） 部门 B 负责重量重心分析与设计的研制过程；

3） 部门 C 负责飞行力学分析与设计的研制过程；

4） 当部门 A 产生新的结果数据时，需要及时的传送给部门 B，部门 B 在最新的数据上进行后续工作。当部门 B 如果发现部门 A 的数据不符合要求，则需要及时的反馈部门 A，否则将部门 B 产生的新的数据文件传送给部门 C。

5） 同理，部门 C 在部门 B 所给的最新数据上进行后续工作。当部门 C 如果发现部门 B 的数据不符合要求，则需要及时的反馈部门 B。部门 B 在部门 C 所给的反馈信息的基础上进行后续工作。

另一方面，现有的各专业软件系统使用的数据库不同，数据格式不统一，各专业软件系

统均有自己的数据结构，因而各专业软件系统需要交互信息，就必须进行相应的消息转换，将格式不符合的消息转换成所能接受的消息格式。异构数据的集成也是本文需要研究的问题之一。

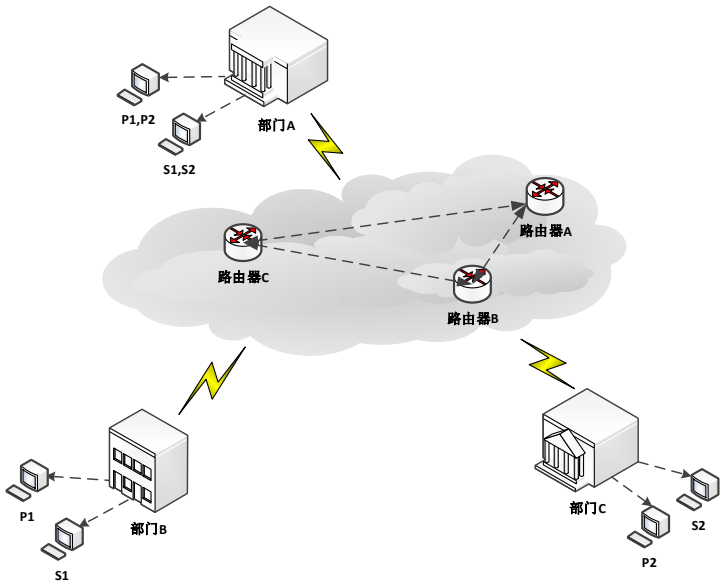

图 3.1 飞机协同设计仿真应用场景图

3.1.3 飞机协同设计功能需求

针对现有的各专业软件系统信息交互困难，导致飞机设计周期变长的问题，为了实现飞机协同设计，需要设计飞机协同设计数据服务中间件，其功能需求如下所示：

- 通信域配置： 飞机设计的过程涉及到很多部门，各部门之间可能不在同一个区域，并且某个部门只需要与其中的部分部门交互信息，不需要将自己的结果信息分发给所有其他的部门。通过信息域的设置，只有在同一个信息域内的部门可以交互信息。
- 服务质量控制：飞机设计涉及到多个学科，每个学科均有自己的需求，并且各不相同。中间件需提供丰富的 QoS 支持以及人性化的配置方式。本中间件可以动态配置主题 QoS 以满足不同用户的需求或满足相同用户不同时间段的需求，需要配置的 QoS 主要包括 DEADLINE 、 RELIABLITY 、 HISTORY 、 TRANSPORT_PRIORITY 、 以 及 LATENCY_BUDGET 等。
- 松耦合交互：由于航空事业的迅速发展，飞机设计技术将会更加先进，因而将不断会有新技术代替现有技术是必然的发展趋势，因此，飞机协同设计平台会不断增加新的分支或者减少某个分支，因而各专业软件系统应尽量松耦合的结合在一起。
- 信息共享的及时性：飞机设计中设计到很多部门相互协作，这些部门可能分布在不同的地方，因而信息交互困难。但各分支之间关系复杂，需要相互协作才能完成，因而各专

业软件系统需要及时交互信息，这样可以有效的缩短飞机设计周期。

- 异构数据的集成：现有的各专业软件系统使用的平台和数据库各不相同，各专业软件系统之间想要交互信息，那么异构数据是亟须解决的问题之一。

3.2 ACD_DSM 设计

3.2.1 ACD_DSM 结构设计

ACD_DSM 框架结构如图 3.2 所示，划分为 3 个层次，从上到下分别为：应用层，中间件层，数据资源层。中间件层是整个系统的核心层，它将应用程序与底层的数据资源层隔离开来，用户不必关心底层数据传输使用什么样的协议以及其它节点使用什么样的操作系统。

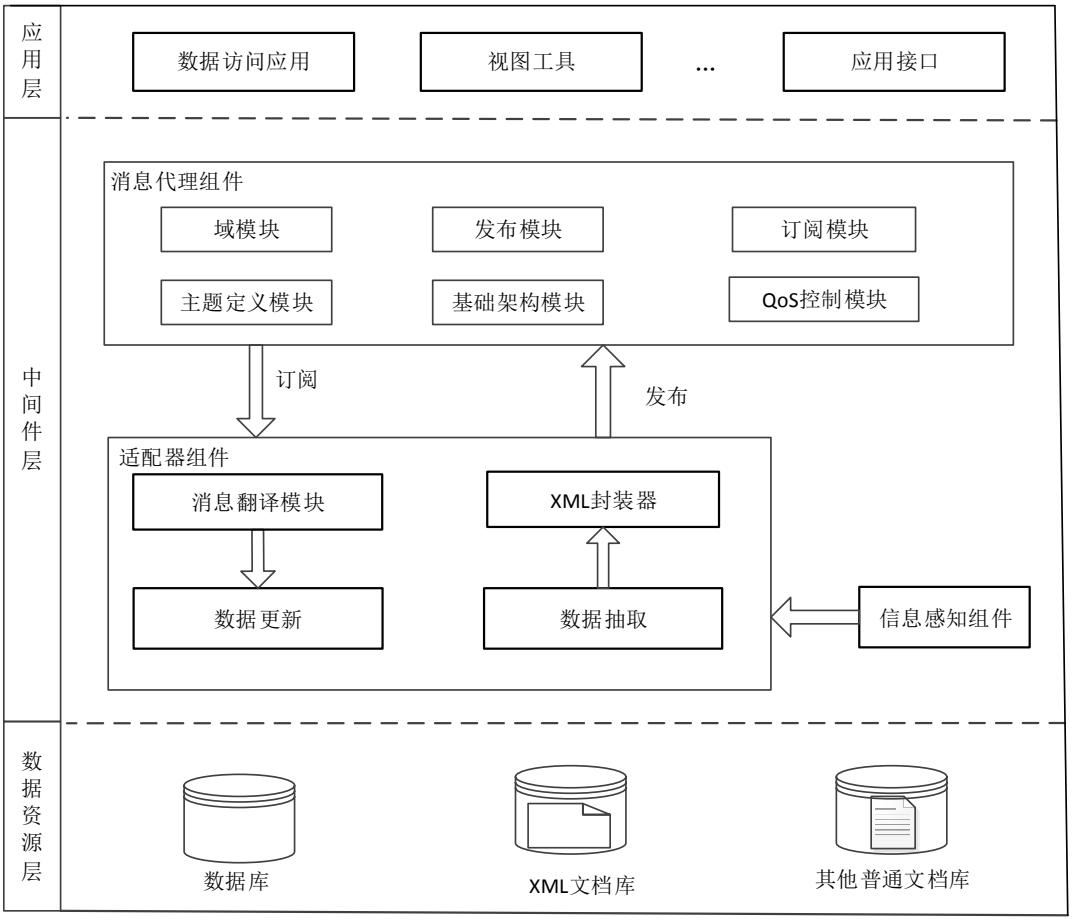

图 3.2 ACD_DSM 框架图

- 应用层：应用层在中间件层之上，是用户和应用程序接口，提供统一的数据调用接口供用户使用，负责将处理逻辑层返回的结果以统一的视图界面显示给用户。
- 中间件层：数据集成中间件层主要包括三个组件，分别为信息感知组件、适配器组件和消息代理组件。信息感知组件负责监听专业软件系统是否有新的结果数据产生。适配器

组件负责数据格式的转换和数据库相关操作，主要包括消息翻译模块、数据更新模块、XML 封装器模块、数据访问模块。数据访问模块的功能是对数据资源层的数据提取。XML 封装器模块按照框架中数据传递的格式标准,对源数据进行格式转换,以统一的数据结构交予消息代理进行消息传递和交互。消息翻译模块负责将消息代理组件接收的消息进行翻译，转化成本专业系统所能接受的格式。数据更新模块负责将新的结果数据更新到本地数据库中。

基于 DDS 的消息代理是中间件的核心，负责各专业软件系统的信息交互，发布/订阅的通信机制保证了消息实体间无连接、异步和松耦合的消息传输，同时 DDS 强大的 QoS 策略保证了消息传递的可靠性和持久性。消息代理组件主要包括域模块、发布模块、订阅模块、主题定义模块、基础架构模块和服务质量控制模块。域模块实现了域相关功能；发布、订阅模块实现了消息代理组件的信息发布订阅的功能；主题定义模块主要实现对飞机协同设计平台各数据的主题定义；基础架构模块负责消息代理组件的通信；服务质量模块负责对消息代理组件进行相关 QoS 配置。

- 数据资源层：数据资源层位于体系结构的最底层，是数据的存储层，系统运行数据的提供层。主要存储飞行器协同设计平台中参与设计的各功能软件的数据，包括各功能软件的本地数据库、XML 文档和其他普通文档。

3.2.2 ACD_DSM 总体流程设计

ACD_DSM 的总体流程步骤如下：

Step 1. 系统启动，各专业软件系统订阅自己感兴趣的主题；

Step 2. 信息感知组件开始对本专业软件系统的相应数据表进行监控；

Step 3. 当对应数据库表里面的数据发生改变时，调用适配器组件对变化数据进行抽取；

Step 4. 适配器将从数据库中抽取出来的数据转化为飞机协同设计系统的统一消息格式；

Step 5. 通过消息代理组件将消息发布出去，并进入 DDS 发布订阅流程；

Step 6. 等待远程系统发布消息，当接收到远程系统发布的消息时，转到 step7，否则一直处于等待状态；

Step 7. 将接收到的消息翻译成本专业软件系统所能接受的消息，并存入数据库。

ACD_DSM 的总体流程图以及 DDS 发布订阅流程如图 3. 3、图 3. 4 所示。

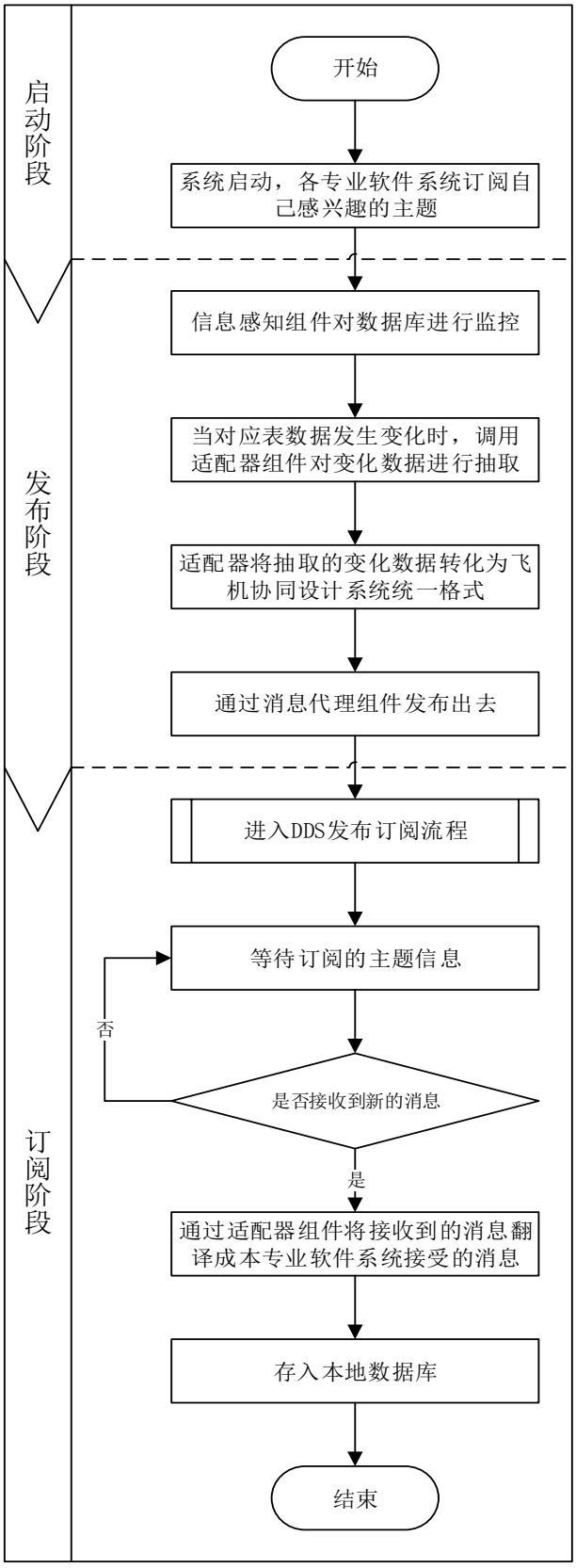

图 3.3　ACD_DSM 总体流程图

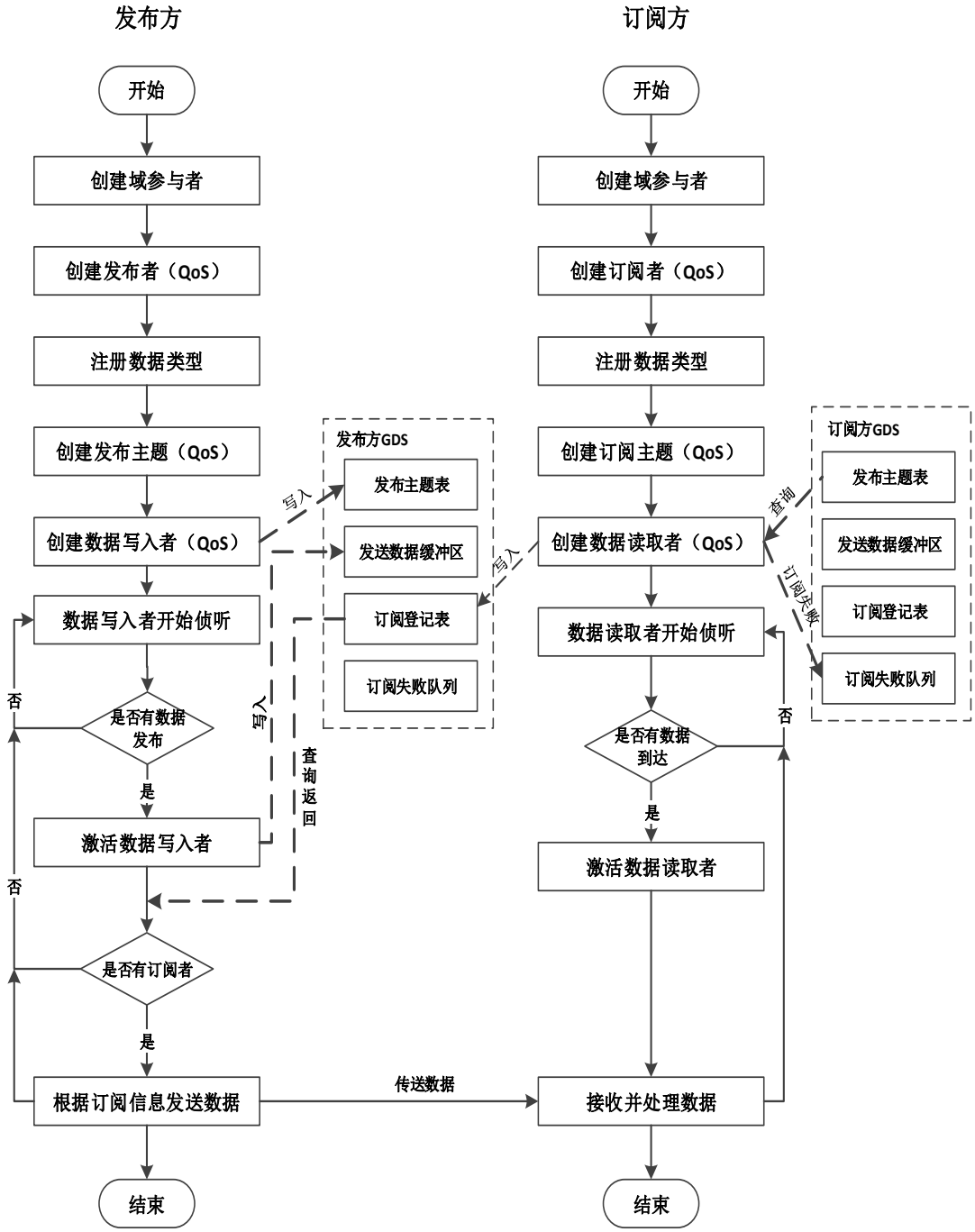

图 3. 4 DDS 发布/订阅流程图

3.3 OpenDDS 分析以及优缺点

3.3.1 OpenDDS 分析

为了实现消息代理组件，本文选用开源项目 OpenDDS 作为开发基础，在 OpenDDS 的代码实现的基础上改进完成。作为 OMG 数据分发服务的一种开源实现的 OpenDDS[57]，谨遵实时系统 1.2 版本的 DDS 规范[3]和实时公布/订阅互操作性通信协议 2.1 版本的 DDS-RTPS 规范[58]。

OpenDDS 由 OCI（Object Computing Inc）公司设计和维护，可从 OpenDDS 的社区门户中寻求到帮助，目前最新版本为 v3.5 版本。

OpenDDS 是用 C++语言针对对象管理组织关于实时系统数据分发服务规范的开源实现。虽然 C++语言是 OpenDDS 所采用的实现语言，但它同时也提供 JAVA 和 JMS 这两者开发的接口，所以，对于 OpenDDS 的使用并不是仅限于 C++语言的。

OpenDDS 的建构基础为 ACE（Adaptive Communication Environment 即自适应通信环境，它是一套基于 C++语言的开源网络可开发库），自适应通信环境作为实现跨平台以及可移植性的保障。此外，为了给自身提供 DCPS 信息仓库，OpenDDS 还运用了 TAO（The ACE ORB，是基于 ACE 基础上的 CORBA 实现框架）所具有的 IDL 编译器等功能。

为了实现让参与者在分布式应用程序中有效地分发数据的目的，DDS 规定了一个不是特定于 CORBA 的服务。这个规范界定了两种具体的模型：平台无关模型（PIM）以及将 PIM 映射到 CORBA IDL 实现的平台相关模型（PSM）。此服务又被分为两层接口：以数据为中心的发布/订阅（DCPS）层以及可选择的数据本地重构（DLRL）层。DCPS 层通过透析与数据主题、发布者以及与订阅者相关的 QoS 约束条件，把发布者提供的数据传递给数据订阅者。DLRL 层应允远程局部对象以访问本地数据类似的方式进行分布式数据的共享。其中，DLRL 层是以 DCPS 为基础的。OpenDDS 现在所实现的 DCPS 层致力于遵循 OMG DDS v1.2 规范，并没有涉及 DLRL 功能的具体实现。

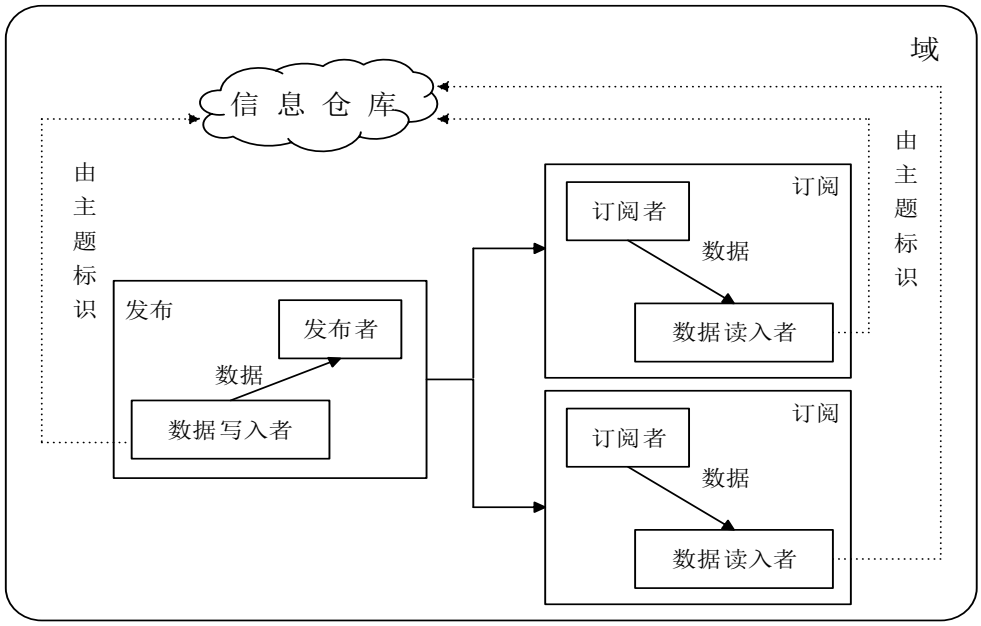

图 3. 5 OpenDDS DCPS 实体关系图

图 3. 5[57]所示是 OpenDDS DCPS 实体关系图，与第二章中的 DDS 标准有相同有域、域参与者、发布者、订阅者、数据写入者、数据读取者，主题等实体，同样各实体之间需要通过主题来匹配。与 DDS 标准中不同的是 OpenDDS 除了提供 RTPS 协议的点对点发现方式外，还提

供了基于信息仓库的匹配方式，该方式为集中式的发现方式。

3.3.2 OpenDDS 的优缺点

本文选用 OpenDDS 作为消息代理组件，OpenDDS 现在实现的 DCPS 层遵循 OMG DDS v1.2 版本，提供了 DDS 规范中 DCPS 层所有的功能，我们可以直接使用，不需要重新实现，这是使用 OpenDDS 开发的最大的好处。

但使用 OpenDDS 开发也有不足之处，主要有三点。第一点是代码上手比较困难，所给的材料仅有使用手册，而 OpenDDS 是基于 COBRA 的一种实现，其中使用的技术很多，包括 ACE/TAO、COBRA，Perl 等，因而首先需要对这些技术有大体的了解，并且项目代码量极大，这对使用和学习均有一定的难度；第二点是代码实现上是遵循 OMG 规范实现的，但规范给出的是最基本的 DDS 实现，而在实际应用中有时不能满足实际的需要，例如多媒体文件的传输 OpenDDS 是不提供的；第三点是 OMG 规范提供的自动发现协议及其算法还存在很多不足之处，后续有很多学者在其基础上改进。

3.4 ACD_DSM 的信息感知组件设计

现有的专业软件系统已经很成熟，内部具体功能比较复杂，但现需要各专业软件系统之间能够信息交互，如果选择重新编码的话，工作量极大，将会耗费大量的人力物力。因而我们应该选择一种实现方式，这种方式可以在不改动原有的各专业软件系统的自治性的情况下实现各专业软件系统之间的信息共享。

信息感知组件的引入，很好的保证了原有各专业软件的自治性。信息感知组件主要实现对各专业软件系统新数据的感知，当有新的结果数据产生时，信息感知组件可以调用适配器组件，完成对新数据的提取工作，进而通过消息代理组件发布出去。现有的各专业软件系统数据主要存储于数据库中，因而采用对数据库的检测来实现对新数据的监控。

现有的对变化数据的捕获方法主要有：

● 触发器法：在数据库对应的数据表中创建相应的触发器[48]，如果有对对应的数据表进行修改、插入和删除等 MDL 相关命令时，此时创建的触发器将会被唤醒，然后将变化数据提取出来，这种方法简单容易实现，但该方法将占用许多系统资源。

● 日志法：通过分析各专业软件的数据库系统日志的信息来获取数据库中对应数据表的所有变化序列，鉴于数据库系统日志的格式并非是公开的，因而只能利用特定厂家提供的专属的对数据库日志进行分析的工具或接口来完成，否则需要开发一套日志变化的捕获程序，这种程序的开发难度相对较大。

● API 法：即在应用程序以及数据库之间引进一种类别的中间件，由该中间件来提供系列的 API 接口。在实现应用程序对数据库操作的同时，这些中间件也将数据库表的具体变化序列

记录下来，从而达到捕获变化的目的。该方法需要原有系统的 API 的支持，并且如果一些不经过 API 操作的，也是无法捕获的。

结合现有的各专业软件系统的特征，各系统使用的数据库不同，因而日志法使用难度较大，而各专业软件系统没有相应的 API 支持，因而本文最终选用触发器法完成信息感知组件。具体实现方式参见第六章。

3.5 本章小结

本章第一小节首先进行了需求分析，给出了飞机协同设计的特点、应用场景和功能需求；在第二小节中给出了基于 DDS 的飞机协同设计中间件的总体设计；第三小节分析了开源项目 OpenDDS，并给出了不足之处；最后一小节给出了信息感知组件部分的设计思路，消息代理组件部分和适配器组件部分为为主要研究对象，将在后面两章中分别介绍。

第四章 ACD_DSM 的消息代理组件的设计

基于 DDS 的消息代理是整个数据交互框架的消息总线，需要灵活、可靠地进行信息交互。本章首先介绍了发布订阅的消息中间件的原理，其后对 ACD_DSM 的主题文件进行了设计，然后重点对 OpenDDS 现使用的自动发现协议进行了改进，提出了基于 DCF 的自动发现协议，进一步降低网络负载，最后 ACD_DSM 在使用 OpenDDS 的基础上，给出了 XML 文件 QoS 控制的控制方式。

4.1 发布/订阅通信模式

发布/订阅通信模式可以被视作是新的分布应用通信模式当中的一种，该模式将匿名性完好地提供给了通信实体，也就是说通信实体并不要了解消息经由谁生产或者被谁消费，只要迎合实体所感兴趣的主题将这两者自然地联系起来。发布/订阅通信模式具备异构性、异步性、动态性和松耦合的特点，也能够达到分布式系统现阶段的具体需求。MOM 因为具备在广域网的环境中松耦合系统间异步、动态的数据传输以及优良的可扩展性等优越性能而成为学术领域研究的热门。

发布/订阅消息中间件的具体原理如图 4.1 所示。消息发布者以及订阅者将分别与 MOM 展开通信，消息发布者主要负责把包含各种不同主题内容的消息发布到 MoM，而消息订阅者则向 MoM 提出自己感兴趣的具体主题的订阅需求。而 MOM 的具体工作则是将接受到的由双方提供的主题逐一进行匹配，如果主题一致即匹配成功后就会不断将与订阅者感兴趣的主题一致的消息推送到订阅者处，这个过程将一直进行除非订阅者提出取消订阅的申请。

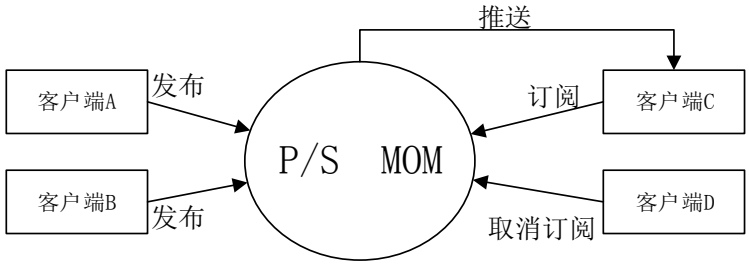

图 4.1 发布/订阅原理图

从这样的过程中可以知道，应用 MOM 完成了发布者以及订阅者双方在时间、空间、流程这几方面的解耦：

- 时间解耦：即发布方和订阅方之间进行消息传输的前提条件不再是双方同时在线，这种

异步传输能力的产生主要是消息中间件应用存储转发的途径得以实现；

● 空间解耦：即发布方和订阅方现在没有必要了解对方的空间信息，例如其物理地址以及端口、对方的逻辑名字等等；

● 流程解耦：即发布方和订阅方同时执行发送或者是接收数据的操作，这样的操作并不会影响甚至是扰乱到双方的控制过程，其内在的逻辑顺序不会因此而改变。

上述的模式有效地确保了建立于发布/订阅通信模型基础的消息中间件具备良好的可扩展性、灵活性，而且基于发布订阅的通信是一种打破被动、破处时间阻碍的信息交流方式。一旦消息发布者发布了动态更新的数据，基于发布/订阅的消息中间件就会通过发布事件来告知消息订阅者有新的可用的数据，不再需要消息订阅者展开频繁的查询工作，极大程度上减少了时间成本的浪费。因此，可以将该通信模型应用于对实时性、异构性、异步性、动态性和松耦合等方面有一定需要的应用。目前，发布订阅通信模型已然被 OMG 组织作为一种相对标准的企业应用集成通信模型进行推广。

4.2 主题文件的定义与设计

主题是通信双方进行数据连接的桥梁，根据本文第三章的分析，现有的各专业软件系统之间需要进行数据传输和共享，因而需要为各种需要交互的信息定义成各种主题。本文的消息代理组件选用 OpenDDS 作为开发基础，因而主题文件类型需采用 IDL 文件类型定义。

DDS 是基于主题匹配的，主要依据端点之间的主题信息是否相同以及 QoS 策略是否兼容。本文定义数据类型的 IDL 文件如图 4. 2 所示。

```
Module  XMLMSG
{
    #pragma DCPS_DATA_TYPE  " XMLMSG:message"
    #pragma DCPS_DATA_KEY   "XMLMSG:message subject_id"


    struct message
    {
        string from;
        string subject;
        long subject_id;
        string xmltext;
    };
};
```

图 4. 2 主题类型文件定义

通过以上 IDL 文件定义了 ACD_DSM 消息代理中消息传输格式，每个消息的主题名为结构体

message 的属性 subject，每个 DDS 端点通过该主题名来匹配。而传输的内容通过 xmltext 来传输，为了解决数据的异构问题，需将传输的消息转化成 XML 格式消息，其后利用 string 类型来发送。具体如何将各专业软件的信息转化为 XML 消息见第五章。

4.3 发布订阅自动发现机制的分析与改进

发布订阅自动发现机制是数据分发服务的核心技术。DDS 是基于主题匹配的，主要依据端点之间的主题信息是否相同以及 QoS 策略是否兼容。OpenDDS 是基于 OMG RTPS 标准设计的，其还是使用的基于 SDP 的自动发现协议，该协议存在很多缺陷，不能适用于飞机协同设计。本小节首先对 SDP、SDPBloom 自动发现协议进行相关分析，然后给出自己的基于 DCF 的自动发现协议 DCFSDP，最后通过性能分析和实验验证了有效性，有效地减少网络负载。

4.3.1 基于 SDP 的自动发现协议

2009 年 RTPS 标准 2.1 版本[58]发布指出 DDS 自动发现机制的实现至少需要支持 SDP 协议，而标准 DDS 自动发现机制就是采用一种基于 SDP（Simple Discovery Protocol）协议的自动发现算法。RTPS 标准规范中将发现过程分为两个阶段：简单参与者发现阶段和简单端点发现阶段，如图 4. 3 所示。数据分发服务中的自动发现机制使用内置数据写入者/数据读取者实现，只有当数据写入者和数据读取者的数据主题信息以及 QoS 相互兼容时，他们之间才可以发送和接收数据。

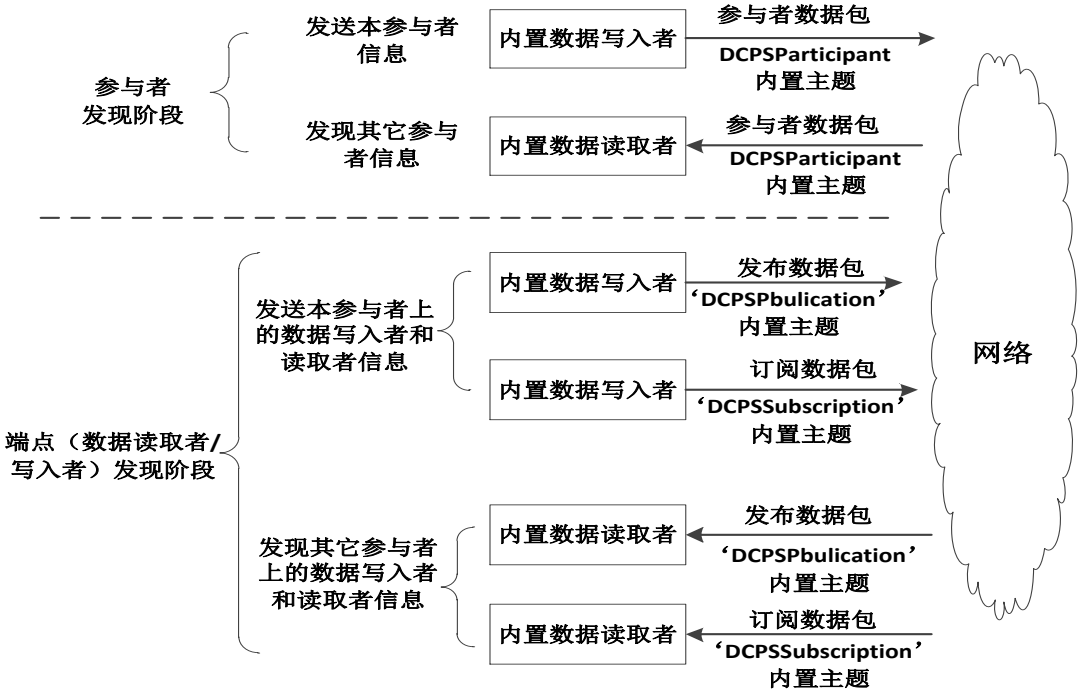

图 4.3 基于 SDP 自动发现协议两阶段

1)参与者发现阶段。通过一个内置数据写入者来发送本地域参与者数据包给远程域参与者，一个内置读取者读取远程域参与者数据包。这个数据包包括参与者的全局唯一标识号 GUID（Globally Unique Identifier），传输位置（地址和端口号）以及它使用的 QoS 策略等。这些消息通过高效（Best Effort）组播传输方式定期地被进行发送。参与者通过周期发送数据包来维持域参与者的生存情况。同时，当域参与者的 QoS 发生变化或参与者被删除时，这些消息也将需要被发送。

2）端点发现阶段。应用程序的数据读取者/数据写入者相关信息，如标识号 GUID、QoS 配置等，将通过发送发布方/订阅方声明的数据包进行交换，该阶段使用可靠的（Reliable）传输方式。当发现一个远程数据写入者/数据读取者时，中间件决策本地应用程序是否有匹配的数据读取者/数据写入者，只有当数据写入者和数据读取者有同样的主题、数据类型以及兼容的 QoS 策略时，局部实体与远程实体才能进行匹配，匹配成功后，两端才能进行相互通信。

基于 SDP 协议的自动发现算法中，每个参与者需要将自己的端点（数据读取者和数据写入者）信息发送给其它所有的远程参与者并接收其它远程参与者的端点信息。此外，每一个参与者需要存储系统中所有的参与者端点信息，以便新节点加入系统时进行匹配。当一个新的参与者或端点加入时，一条消息将会被发送给系统所有的参与者，然而，在一个大型网络中，对于一个给定的参与者来说，大多数端点信息它是不需要，它只会关心与自己有数据交互的参与者端点信息，因此，对于它来说存储了大量不必要的信息，造成内存浪费，同时还产生了大量不必要的网络数据传输。在文献[65][66]指出执行 SDP 算法需要的内存的消耗以及产生的网络通信量完全依赖于整个系统的端点个数，即随着网络中端点个数的增加，该算法的缺点越明显，该协议可扩展性较差，无法应对广域网的要求。

4.3.2 基于 Bloom Filter 的自动发现协议 SDPBloom

4.3.2.1 布隆过滤器 Bloom Filter

Bloom Filter[59]是上个时期 70 年代由 Howard Bloom 提出的二进制向量数据结构，使用位数组来表示一个集合，并判断该集合是否包含某个元素时具有相当高的正确率。Bloom Filter 使用相互独立的哈希函数，将集合中的所有元素映射到一个二进制过滤器向量中；开始向量中的所有二进制位均初始化为 0，然后使用哈希函数计算，然后把计算结果对应的数据位设置为 1[60]。从 Bloom Filter 提出开始，就开始被广泛用于摘要缓存（Summary Cache）[61]、开放的服务发现体系结构 OSDA（Open Service Discovery Architecture）[62]以及应用到分布式数据库中进行查询， Bloom Filter 最早是被应用于拼写检查，即在过滤器内存放的是有效单词列表[63]。

Bloom Filter 相对于经典的哈希函数最大的好处在于空间效率[64]上。另一方面，由于 Bloom Filter 不用采用任何措施去处理碰撞，因此，无论等待插入元素集合的大小为多大， Bloom Filter

处理元素的时间跟哈希函数的计算时间是相同的，同样对于查询一个元素是否在集合内所用时间亦为哈希函数计算时间。本文将通过一个简单例子具体说明 BFV 插入和查询以及误报是如何发生的。假定一个 BFV 为 8 位，装入的集合 S 的元素个数为 2 个，以及哈希函数的个数为 3 个，初始状态时 BFV 每一位都为 0，集合 S 为空集；如图 4. 4 所示：

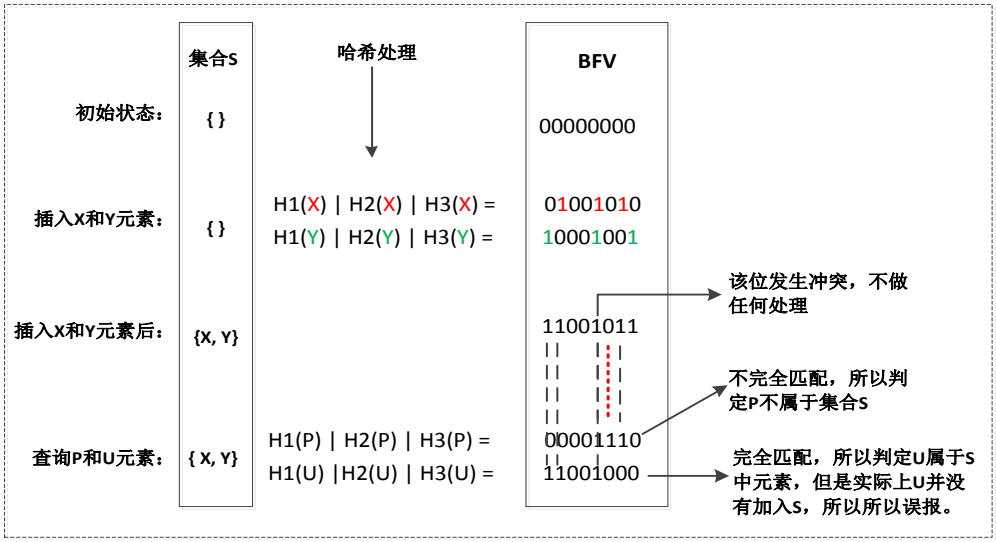

图 4. 4 BFV 插入和查询样例

4.3.2.2 SDPBloom 算法描述

SDPBloom[66]协议主要思想是在参与者发现阶段，各个参与者将自己所有包含的端点信息映射到 BFV 中，然后随着参与者数据包发送给远程的其他域参与者，描述信息主要包括端点的主题相关信息，因 BF 不支持区域查询，因而无法包含 QoS 信息。当本地参与者收到远程参与者数据包时，根据定义的包格式解析数据包并获取远程端点信息，根据得到的信息与本地端点进行匹配，若匹配成功则将本地端点数据包（不再仅仅是描述信息，包括主题数据和主题 QoS）发送到远程端点进行再匹配，若匹配成功则端点之间即可建立通信连接，进行数据间通信；若匹配不成功则返回类似“本节点不存在该端点信息”的数据包。SDPBloom 算法描述如图 4. 5 所示：

```
SDPBloom 算法：
    输入：参与者信息
    输出：发现结果
1：Build Bloom Filter BF
2：while Participant is enabled do
3:        if Endpoint deleted then
4:                Rebuild BF
5:        end if
6:        for all New Endpoint E do
7:                Build E key Ek
8:                insert Ek in BF
9:        end for
10:       Add BF to Participant DATA message
11:       for all Remote Participant filter r do
12:             for all Local Endpoint key Ek do
13:                   if Ek ϵ r then
14:                         {try to start publication or subscription}
15:                         send Endpoint information(SSEDP message) to the remote Participant
16:                         if Ek’s SSEDP desired message is not received then
17:                               matched Endpiont Ek is a false positive
18:                         end if
19:                   end if
20:             end for
21:       end for
22: end while
```

图 4. 5 SDPBloom 算法描述

总的来说，SDPBloom 算法较 SDP_ADA 算法的改变主要是参与者数据包的变化，在原有数据包的基础上加上一个 BFV，对于本地参与者来说，意味着从“给我你所有的消息”转变为“给我一些信息让我知道你有什么”，这样本地不需要存储远程参与者发来的所有消息，从而可以降低内存消耗；同时，远程参与者仅需要和匹配上远程参与者交互，从而也降低了网络数据传输量。在文献[66]末尾提到不支持 QoS 信息的处理，在各实体 QoS 多样的情况，过滤器的筛选效果会很差，因而无法适用于飞机设计的多样性需求。

4.3.3 基于 DCF 的自动发现协议 DCFSDP 设计

4.3.3.1 Dynamic Count Filters

标准的 BloomFilter 数据结构相对简单，它只有两种操作：插入和查询。如果集合是静态的，此时标准 Bloom Filter 的性能比较好，但当需要表达的集合经常变动，标准 Bloom Filter 性能就会下降，这是由于它无法处理删除工作，需要删除元素是，就需要重新去构造，这将会耗费大量的系统资源。

为了解决了该问题，人们提出了 Counting Bloom Filter[67](CBF)，主要思路是将标准 Bloom Filter 位数组的每一位扩展成多位，形成一个个小的计数器（Counter），当插入元素时，将对应

的 Counter 的值加 1，删除元素时减 1。 CBF 是用空间换效率的思想的一种体现。Counting Bloom Filter 原理图如图 4.6 所示。

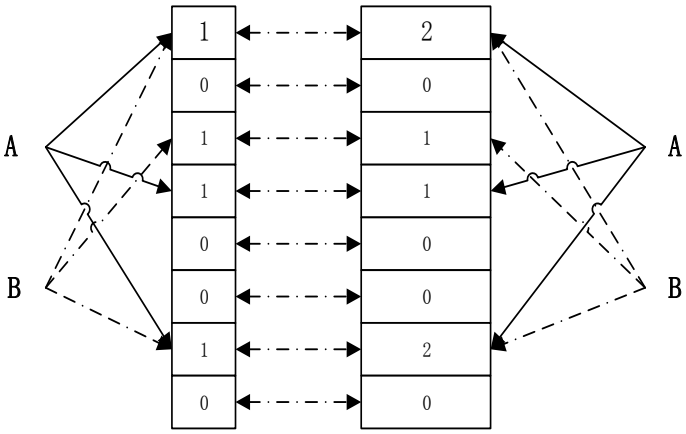

图 4.6 Counting Bloom Filter 原理图

Dynamic Counting Bloom Filter（DCF）[68]主要思想是通过将 CBF 的计数器使用两个数组来存储，分别是基本的 CBF 和 OFV。其中 CBF 计数器的长度固定为 log(集合元素个数/集合中不同元素的个数)； OFV 长度并不固定，用来处理计数器溢出。

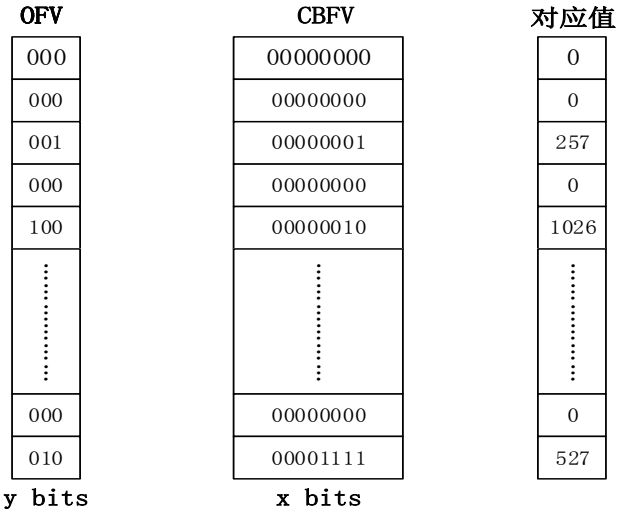

图 4.7 DCF 示意图

图 4.7 中给出计算原理。当查询一个计数器时，DCF 需要访问两次内存，分别读取 CBFV 和 OFV 的值。假设读到的 CBFV 和 OFV 对应的值是 C_j 和 OF_j，那么计数器的值就能够通过公式 $V_j = (2 \times OF_j + C_j)$得到 。例如图 4.7 中的第 5 个计数器对应的 CBFV 和 OFV 的值分别是 100 和 00000010，通过公式计算得到计数器是 1026。

4.3.3.2 DCFSDP 协议描述

SDPBloom 算法虽然通过标准的 Bloom Filter，在一定程度上减少了内存消耗以及网络通信量，但是该算法在参与者阶段并没有将端点 QoS 考虑进去。因为 BF 不支持区域查询，而 QoS 兼容的条件是满足 RxO 特性，因而 BF 不能处理该情况，因而 SDPBloom 算法在网络中存在大量 QoS 不兼容时，效果就会很差。本文提出了一种改进的自动发现协议 DCFSDP，该协议在 SPDBloom 的基础上，通过制定添加元素的规则，可以把 QoS 信息考虑到过滤器向量中，从而在一定程度上较少因 QoS 不匹配导致的发送无效的数据包，为了减少因为端点离开而需重构过滤器向量导致的过多资源消耗，将 BF 升级为 DCF。

表 4. 1 DDS 支持的 QoS

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

表 4. 1 给出了 DDS 规范中的 22 种 QoS 策略，其中有 8 种需要满足 RxO 特性，即该 QoS 策略需要满足 Request vs Offered 模式。对这 8 种 QOS 的匹配条件分为以下 4 种情况处理：

1.若该 QOS 策略适用范围不包括数据写入者和数据读取者，此时就无需将其考虑到 DCFV 中，因为其对端点发现阶段的筛选没有帮助意义。如 PRESENTATION 策略。

2.若该 QOS 策略需要精确匹配，此时只需要像构建主题信息一样，将其加入到 DCFV 中。如 OWNERSHIP 策略等。

3.若该 QOS 策略的值是几个具体的值，并且这几个值有大小关系，此时将与该端点的 QOS 值的能匹配的情况全部加入到 DCFV 中。如 RELIABILITY 策略等。

4.若该 QOS 策略的值属于一个范围区间。此时根据实际情况选择一个值作为该区间的分点（如该值选择默认值），将范围划分为两个区间，把区间作为一个整体，这样就变成第 3 种情况。如 DEADLINE 策略等。

通过以上方法，就可以把 QOS 的信息考虑到 DCFV 中，在参与者发现阶段，通过 DCFV 进行粗略判断，从而可进一步减少网络负载和内存消耗。

```
DCFSPD算法：
要求：域参与者处于激活状态
1：Build DCFV
2：while Participant is enabled do
3:      if Endpoint E deleted then
4:            build E keyList EkList
5:            for all key Ek in EkList do
6:                Delete Ek from DCFV
7             end for
8:      end if
9:      for all New EndPoint E do
10:           build E keyList EkList
11            for all key Ek in EkList do
12:               Insert Ek in DCFV
13            end for
14:     end for
15:     Add DCFV to ParticipantDATA message
16:     for all Remote Participant filter r do
17:           for all Local Endpoint key Ekl  do
18:               if r contain Ekl then
19:                   {try to start publication or subscription}
20:                   Send SEDP message to the Remote Participant
21:                   if Ek’s SEDP desired  message is not received then
22:                        Matched Endpoint Ekl is a false positive
23:                   end if
24:               end if
25:           end for
26:     end for
27: end while
```

图 4. 8 DCFSPD 算法

对 QoS 进行以上处理，一个端点的信息需要变成很多个端点信息插入到 DCFV 中，即需要插入 DCFV 的信息会变成原来的几十到几百倍，这样处理的时间会变长。DCFSDP 中使用 DCF

作为过滤器，在软件启动后，就不再需要重新去构造 DCF，由于过滤器增加集合元素的时间均为哈希函数的计算时间，因而启动所使用的时间也是可以接受的。在软件运行过程中，当有新的端点加入或者端点的离开，仅仅是由原来的一个信息变成几十到几百个，对于现在计算机处理能力，其引起的时间变化用户基本上是感觉不到的；当接收到远程参与者发来的 DCFV，此时仅需要查询是否有本地端点包含在 DCFV 中即可。

结合 DCFV 构建策略，图 4.8 给出了 DCFSDP 算法伪代码描述。从图 2 中看出，DCFV 仅在开始时需要构造，在循环中，只需要对新加入的端点或者删除的端点构造一个符合条件的集合，然后对 DCFV 进行相应的插入和删除工作即可。

值得注意的是，算法中需要构建键值 key 共 3 处，第 1、2 处为当有端点加入或者删除时，该情况需要对本地 DCFV 更改，第 3 处为接收到远程参与者发送的 DCFV，该情况是本地对每一个端点构造键值查询是否包含在 DCFV 中。在这两种情况的构造策略是不同的，第一种情况是想告诉接收者哪些是和本端点匹配的，需要把所有符合条件的情况均构造键值，因而一个端点对应一系列的键值，第二种情况是查询本端点是否在这符合的条件中，因而一个端点对应一个键值；另外一个不同点就是对于同一个端点，构建的键值中的身份信息（即该端点是数据读取者还是数据写入者）在两种情况下是相反的，例如该端点是数据读取者，第一种情况下键值中相应位置是用数据读取者标注，而第二种情况下键值相应位置用数据读取者标注，因而在伪代码中用 Ek 和 Ekl 区分。

图 4.9 给出 DCFSDP 协议的参与者发现阶段和端点发现阶段的步骤描述。

DCFSDP的参与者发现阶段具体步骤如下：
Step 1：构建DCF向量DCFV；
Step 2：将所有本地端点描述信息（包括QoS信息）按照制定的策略加入到DCFV中，并将DCFV放到本地参与者数据包中；
Step 3：向所有远程参与者发送本地参与者数据包，同时接收远程参与者发来的数据包信息。
Step 4：当收到远程参与者发来的数据包时，获取DCFV字段，与本地端点信息进行比较；若与本地端点相匹配，则将该远程参与者信息加入到本地信息库，并进入到端点发现阶段。
Step 5：整个参与者发现过程中，当参与者或参与者创建的端点信息发生变化时，将转到step3继续执行。

DCFSDP的端点发现阶段具体步骤如下：
Step 1：当发现过程进入端点发现阶段后，向所有本地信息库中的远程参与者发送本地端点数据包，并接收远程端点数据包。
Step 2：当本地参与者接收到远程端点数据包，将与本地端点信息进行再匹配；如果匹配成功则执行step3，否则执行step4。
Step 3：将远程端点信息加入到本地信息库，建立双方之间的数据通信。
Step 4：向远程参与者发送类似于“本地没有相关的端点”的信息，说明发生了误报情况。

图 4.9 DCFSDP 协议阶段描述

根据整个发现过程两阶段的分析，其实整个发现过程两阶段是相互交叉的，即对端点 A 进

入到端点发现阶段时，也可能进入到端点 B 的参与者发现阶段。整个自动发现过程的总流程设计如图 4. 10 所示：

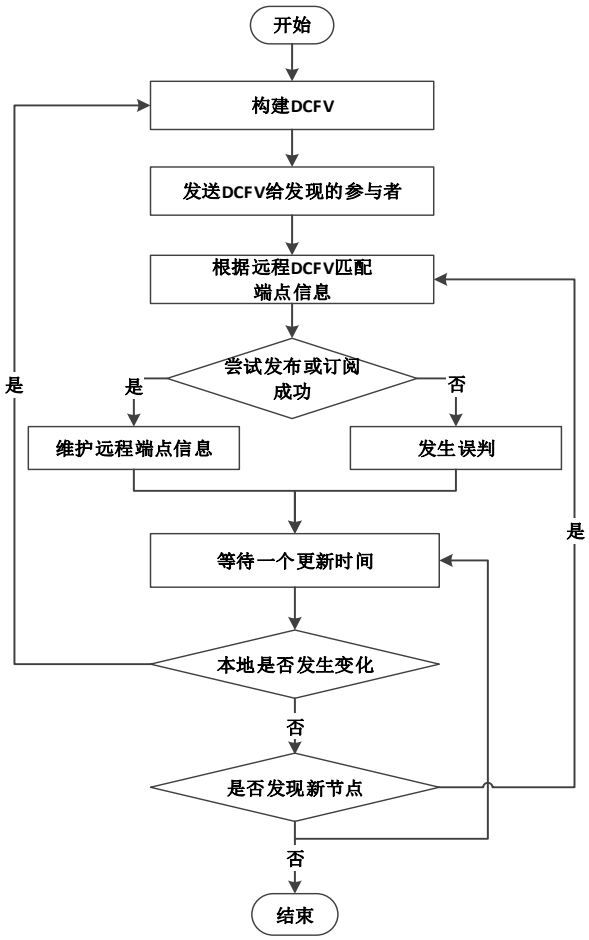

图 4. 10 DCFSDP 流程图

4.3.4 性能分析以及仿真实验结果分析

4.3.4.1 性能分析

本小节主要是从理论上对 SDP，SDPBloom，DCFSDP 三种发现协议在内存使用率和 CPU 使用率上作对比分析。内存使用率主要与一个域参与者需要存储多少端点信息相关，其次还与需要保存的连接数有关。CPU 使用率与需要处理的数据量有关。

假定在域中有参与者的个数为 P，端点的总数为 E。为了简化分析，做如下两个规定：第一，端点是一致分布在域中的参与者中，即每一个参与者均有 E/P 个端点；第二，心跳和确认信息不考虑在内。

参考文献[65],定义 ME_{BF} 为在 SDPBloom 协议中通过 BF 匹配上的总端点数，ME_{DCF} 为在 DCFSDP 协议中通过 DCF 匹配上的总端点数。ME_{BF}、ME_{DCF} 控制着在端点发现阶段需要发送

和接受的信息数。在 SDP 中，每个端点信息都需要发送并保存，所以其需要发送的信息数为 E。如果我们控制 BF 与 DCF 有相同的误报率，此时因为 DCFSDP 相对于 SDPBloom 协议考虑到 QoS 情况，所以在相同情景下，ME_{DCF} 应小于等于 ME_{BF}。综上得出式（4.1）。

$$E \gg ME_{BF} \geq ME_{DCF} \tag{4.1}$$

在不使用广播的情况下，在 SDP、SDPBloom、DCFSDP 三种发现协议中每一个参与者需要发送和接受的消息数分别见式（4.2）[65]、式（4.3）[65]和式（4.4）：

$$N_{P-SPD} = 2 \bullet (P-1) \bullet \frac{E}{P} \sim 2 \bullet E \tag{4.2}$$

$$N_{P-BF} = 2 \bullet (P-1) \bullet \frac{ME_{BF}}{P} \sim 2 \bullet ME_{BF} \tag{4.3}$$

$$N_{P-DCF} = 2 \bullet (P-1) \bullet \frac{ME_{DCF}}{P} \sim 2 \bullet ME_{DCF} \tag{4.4}$$

在不使用广播的情况下，在 SDP、SDPBloom、DCFSDP 三种发现协议中整个网络中需要处理的消息数分别见式（4.5）[65]、式（4.6）[65]和式（4.7）：

$$N_{t-SPD} = P \bullet (P-1) \bullet \frac{E}{P} \sim P \bullet E \tag{4.5}$$

$$N_{t-BF} = P \bullet (P-1) \bullet \frac{ME_{BF}}{P} \sim P \bullet ME_{BF} \tag{4.6}$$

$$N_{t-DCF} = P \bullet (P-1) \bullet \frac{ME_{DCF}}{P} \sim P \bullet ME_{DCF} \tag{4.7}$$

在使用广播的情况下，在 SDP、SDPBloom、DCFSDP 三种发现协议中每一个参与者需要发送和接受的消息数分别见式（4.8）[65]、式（4.9）[65]和式（4.10）：

$$N_{mP-SPD} = \frac{E}{P} + (P-1) \bullet \frac{E}{P} = E \tag{4.8}$$

$$N_{mP-BF} = 1 + (P-1) \bullet \frac{ME_{BF}}{P} \sim ME_{BF} \tag{4.9}$$

$$N_{mP-DCF} = 1 + (P-1) \bullet \frac{ME_{DCF}}{P} \sim ME_{DCF} \tag{4.10}$$

在使用广播的情况下，在 SDP、SDPBloom、DCFSDP 三种发现协议中整个网络中需要处理的消息数分别见式（4.11）[65]、时（4.12）[65]和式（4.13）：

$$N_{mt-SPD} = P \bullet \frac{E}{P} = E \tag{4.11}$$

$$N_{mt-BF} = P \bullet \frac{ME_{BF}}{P} = ME_{BF} \tag{4.12}$$

$$N_{mt-DCF} = P \bullet \frac{ME_{DCF}}{P} = ME_{DCF} \tag{4.13}$$

在 SDP 中每一个参与者需要保存所有端点信息，而在 SDPBloom，DCFSDP 需要保存在与

者发现阶段附带的过滤器向量和本省自带的端点信息，以及通过过滤器向量匹配上的端点信息。因而在 SDP，SDPBloom，DCFSDP 三种发现协议中每个参与者需要保存的端点信息数分别见式（4.14）[65]，式（4.15）[65]和式（4.16）：

$$M_{P-SPD} = E \tag{4.14}$$

$$M_{P-BF} = P + \frac{E}{P} + ME_{BF} \tag{4.15}$$

$$M_{P-DCF} = P + \frac{E}{P} + ME_{DCF} \tag{4.16}$$

ME_{BF} 和 ME_{DCF} 是对端点信息使用过滤器之后得到的匹配端点数，每个参与者还有 E/P 个端点，因而 ME_{BF},和 ME_{DCF} 均小于等于(P-1)E/P,当且仅当所有端点信息刚好全部匹配时才会相等，从而得出 E/P+ME_{BF},E/P+ ME_{DCF} 均小于等于 E。SDPBloom 和 DCFSDP 相对于 SDP 需要多存储 P 个过滤器向量，依据 2.2 节对过滤器的相关分析，P 个过滤器向量的内存占用是很小的。因此可得出以下结论：在一般情况下，SDPBloom 和 DCFSDP 的内存使用均小于 SDP，仅当网络中所有的端点信息刚好全部匹配，此时 SDPBloom 和 DCFSDP 也仅比 SDP 多了存储过滤器向量的空间，但这仅相当于端点存储空间很小一部分。

通过以上分析得出：通过使用过滤器，相对于 SDP 协议，SDPBloom 和 DCFSDP 两种协议很有效的减少了网络负载和内存消耗。DCFSDP 协议将 QoS 信息考虑到过滤器向量中，在主题类似但 QoS 各异时，ME_{DCF} 会明显小于 ME_{BF}，此时相对于 SDPBloom 协议会进一步减少网络负载。

4.3.4.1 仿真实验分析

本文实验环境基于开源项目 OpenBloom Filter[69]和 Open DDS[57]，DCF 是在 OpenBloom Filter 的基础上修改实现。网络传输数据使用 TShark[70]收集分析。试验中共有两种 IDL 文件，具体定义见图 4. 11。

```
数据类型定义：
struct Simple{
        string msg;
};

struct Complex{
        string from;
        string subject;
        long subject_id;
        string msg;
        long count;
};
```

图 4. 11 IDL 定义描述

实验分两组，每组实验中有两个应用程序，每个程序有 50 个主题，即有 50 个端点，设置 BF 与 DCF 需要插入的元素个数为 50 ，误报率为 0.05%。因为只有当两个端点之间主题信息部分匹配并且 QoS 属性相互兼容时，两个端点之间的连接才是有效的。而无效的端点之间进入端点发现阶段，他们之间发送数据包都是不必要的。根据以上特性，设置的第一组实验的环境是主题信息各异，但各端点均采用默认 QoS 设置，即各端点的 QoS 属性是相互兼容的。第二组实验的环境是主题信息部分相同，但 QoS 策略各异，存在不兼容的情况。

现给出两组实验中 ME、ME_{BF} 和 ME_{DCF} 的含义。表 4. 2 和表 4. 3 中 ME 代表在该场景下，各参与者的 50 个端点中最终应该建立通信的端点的个数，这是通过手动设置的达到的；ME_{BF} 和 ME_{DCF} 代表如果使用 SDPBloom 协议和 DCF_SDP 协议时，进入端点发现阶段的端点个数，这是在设置的场景下实验得来的。由于过滤器存在误报率，所以必然有 ME_{BF} 和 ME_{DCF} 大于 ME。而本文设置误报率为 0.05%，端点数量较少，所以出现误报的情况很少。

第一组实验分为 4 个实验场景，全部实体使用默认 QoS 设置，主题信息匹配的个数决定了 ME 的大小。场景描述信息以及实验结果见表 1。

表 4. 2 实验 1 场景描述及实验结果

实验 1 场景描述:均使用默认 QoS 设置				
	场景 1	场景 2	场景 3	场景 4
ME	10	20	30	40
过滤器匹配结果				
ME_{BF}	10	20	30	40
ME_{DCF}	10	20	30	40
实验结果：网络传输比特数				
SDP	85824	86116	86578	87242
SDPBloom	23170	41456	56 964	70224
DCFSDP	23542	41858	57248	70556

图 4. 12 为第一组实验结果的柱状图。通过实验结果分析，我们可以得出以下两点结论：第一点是在 SDP 中，各参与者需要将所有的端点描述信息发送给远程参与者，而 SDPBloom 协议和 DCFSDP 协议通过过滤器向量筛选无效的端点，减少了网络流量；第二点是在使用默认 QoS 设置的情况下，DCFSDP 与 SDPBloom 效果差不多，DCFSDP 传输的比特数比 SDPBloom 稍微多是因为 DCFV 比 BFV 使用空间较大。

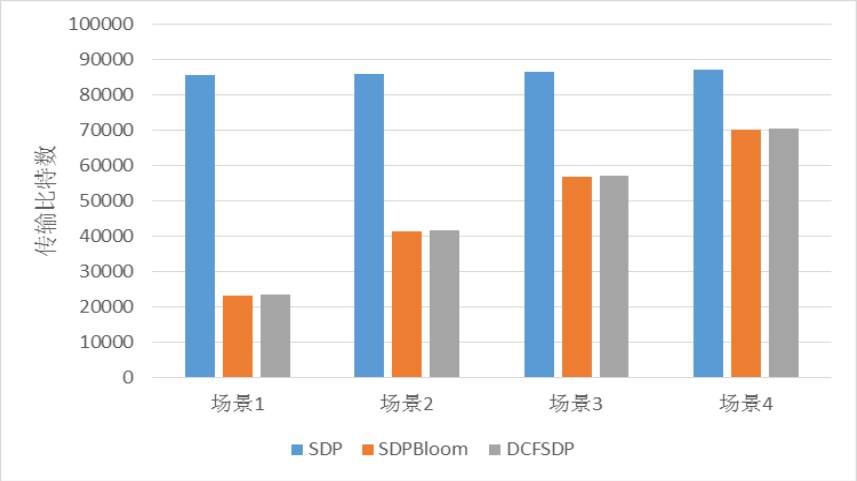

图 4.12 实验 1 结果柱状图

第二组实验分为 4 个实验场景，两个应用程序的主题相关信息全部匹配，QoS 属性兼容的情况最终决定了 ME 的大小，场景描述信息见表 4.3。

表 4.3 实验 2 场景描述及实验结果

实验 2 场景描述:均使用通用主题信息				
	场景 1	场景 2	场景 3	场景 4
ME	10	20	30	40
过滤器匹配结果				
ME_{BF}	50	50	50	50
ME_{DCF}	11	21	32	44
实验结果：网络传输比特数				
SDP	87 226	88110	88 706	89 134
SDPBloom	87 614	88 418	88 998	89 486
DCFSDP	27 112	45 364	60 240	69 878

图 4.13 为第二组实验结果的柱状图。通过实验结果分析，我们可以得出以下结论：在主题信息部分全部匹配，QoS 属性兼容决定端点是否匹配的情况下，SDP 协议和 SDPBloom 协议在端点发现阶段参与者需要将所有的端点信息发送给远程参与者，而 DCFSDP 可以通过过滤器提前判断，筛选无效的端点，有效的减少网络负载。而 ME_{DCF} 会比 ME 多一些是因为规则 4 中是将区间分成两段，当两个端点的对应 QoS 属性值在同一个区间，DCFSDP 协议认为是匹配的，但可能这两个不满足兼容条件，提供方的值小于需求方。

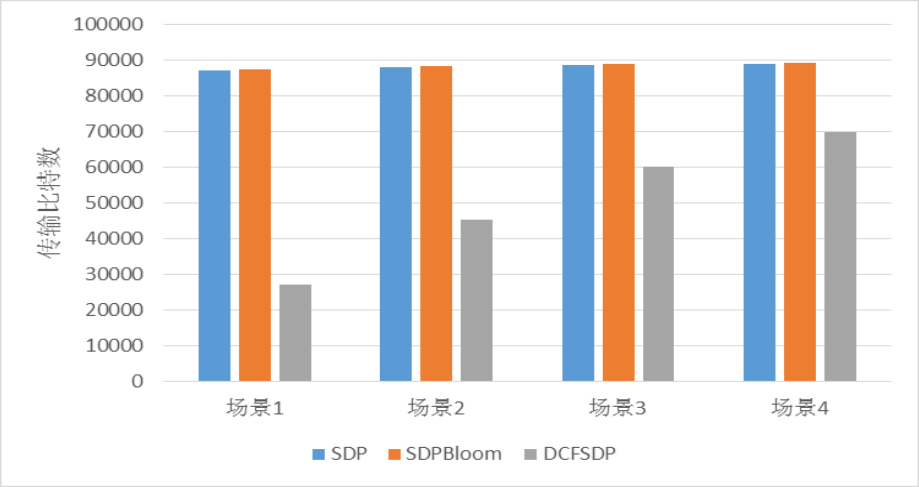

图 4.13 实验 2 结果

综上实验结果分析可得到以下结论：在使用默认 QoS 的情况下，SDPBloom、DCFSDP 相对于 SDP 可以很好的减少网络负载；在使用通用主题，但 QoS 设置各异的情况下，SDPBloom 相对于 SDP 基本没有效果，但 DCFSDP 依然可以很好地减少网络负载。

4.4 ACD_DSM 的 QoS 策略控制

ACD_DSM 使用 OpenDDS 来实现消息代理组件。OpenDDS 遵循 OMG DDS 标准的 DCPS 层实现，已提供了 22 种 QoS 策略的实现，因而直接使用 OpenDDS 提供的 QoS 设置接口即可。图 4.14 给出了代码直接控制 QoS 的代码。

```
DDS::DataWriterQos  dw_qos;
pub->get_default_datawriter_qos(dw_qos);

dw_qos.history.kind = DDS::KEEP_ALL_HISTORY_QOS;

dw_qos.reliability.kind = DDS::RELIABLE_RELIABLITY_QOS;
dw_qos.reliability.max_blocking_time.sec=10;
dw_qos.reliability.max_blocking_time.nanosec=0;

dw_qos.resource_limits.max_samples_per_instance = 100;

DDS::DataWriter_var dw =
   pub->create_datawriter(topic,
                          dw_qos,
                          0,
                          OpenDDS::DDS::DEFAULT_STATUS_MASK);
```

图 4.14 OpenDDS QoS 配置示例

OpenDDS 提供的直接控制方法实现比较简单，但灵活性比较低，也不利于用户配置。为了提高用户体验，ACD_DSM 还提供了 XML 文件配置方式，用户可以通过使用 XML 配置文件来

配置 ACD_DSM 的 QoS 策略。该方法具有一般性以及较高的灵活性，尤其适合发布方或订阅方对数据非功能型需求经常变更的情况。但是使用该方法需要开发人员对 XML 技术能够有一定的了解，充分发挥 XML 的优势，以及需要能够正确的进行参数配置，如 XML 文档当前存放的路径等。通过这种方式用户不需要重新编译应用程序即可对实体的 QoS 进行控制。图 4. 15 给出了一个表示服务质量的 XML 文件，该文件具体内容可以通过用户手动修改或通过用户界面选择设置并修改。

```
<?xml version="1.0" encoding="ISO8859-1"?>
<!-- A XML configuration file -->
 <aocd-dds version = 1.0>
       <qos_profile name="StrictReliableCommunicationProfile">
       <datawriter_qos>
          <history>
              <kind>DDS_KEEP_ALL_HISTORY_QOS</kind>
          </history>
        </datawriter_qos>
        <datareader_qos>
          <history>
              <kind>DDS_KEEP_ALL_HISTORY_QOS</kind>
          </history>
          <reliability>
              <kind>DDS_RELIABLE_RELIABILITY_QOS </kind>
          </reliability>
        </datareader_qos>
        </qos_profile>
        <qos_profile name="DeadlineProfile">
        <datawriter_qos>
           <deadline>
              <sec>10</sec>
              <nonasec>0</nonasec>
           </deadline>
</aocd-dds>
```

图 4. 15 QoS XML 配置文件

4.5 本章小结

本章第一小节首先介绍了发布订阅通信模式，其后三小节在 OpenDDS 的基础上，完成 ACD_DSM 的主题文件的设计、发布订阅自动发现协议的改进和 QoS 控制方法改进。本章重点是对自动发现机制的研究，针对数据分发服务中现有的自动发现机制的不足，对 DDS 的自动发现机制进行改进，给出了基于 DCF 的自动发现协议，有效的降低了网络负载。

第五章 ACD_DSM 的适配器的研究与设计

现有的各专业软件系统使用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。欲让各专业软件系统之间能够信息共享，需要使用适配器，将信息转化为飞机协同设计系统统一的数据格式，并通过消息代理组件发布并订阅得到自己需要的数据。

5.1 适配器的功能与分类

适配器[71]作为与信息系统的接口，其目的是以结构化的方式访问数据或应用软件的功能层。它隐藏了应用软件的翻译、传递消息或者调用接口的复杂过程。适配器是外部系统与消息代理连接的纽带，可以通过适配器将不同应用系统连接起来，解决数据异构的问题。适配器主要工作是将各软件系统的数据转换成全局统一的数据格式，并且对接收到的信息进行翻译，将其转换为本系统可以理解的数据。

从应用层次角度可以将适配器分成技术适配器以及应用适配器。技术适配器主要作用在于实现不同通讯协议之间的转换，进而实现不同技术之间的互操作；应用适配器则是要实现业务系统与消息代理之间的连接，从而使各业务系统可以信息交互。结合我们的需求，我们主要研究的是应用适配器。

应用适配器从功能上可以分为通用应用适配器和专用应用适配器两大类。通用应用适配器能够提供与框架的接口，完成消息验证等通用服务，以及与消息代理组件的信息交互；专用应用适配器完成与具体的应用相关的工作，负责相应的消息转化和翻译工作。

传统应用适配器连接如图 5.1 所示。我们可以看到应用适配器在消息总线和信息子系统之间起到了一个桥梁的作用。它与消息总线之间通过公共的服务请求接口进行交互,与信息子系统之间采用一对一的专用接口连接。

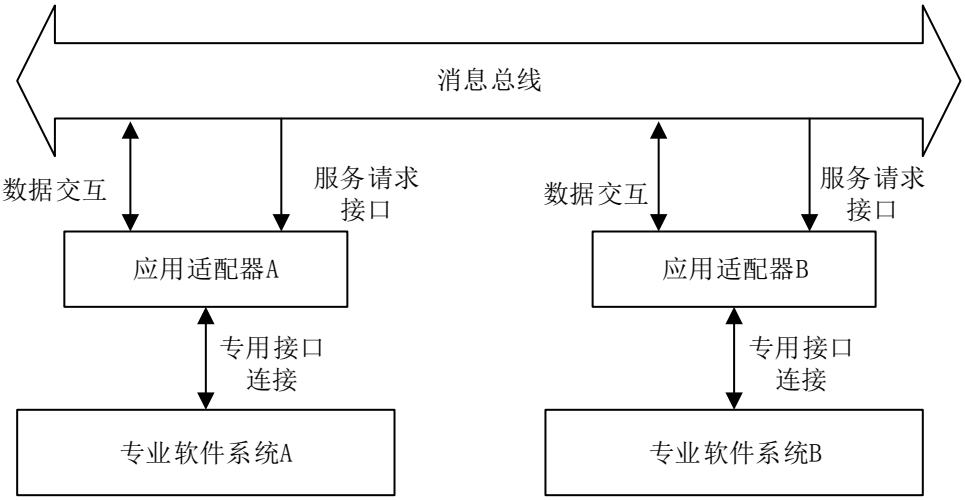

图 5.1 传统应用适配器示意图

5.2 ACD_DSM 应用适配器结构设计

结合飞机协同设计需求，主要的目的是将现有的各专业软件系统能够信息交互，从而各专业软件系统可以协同开发。现使用信息感知组件来对新的结果数据进行监控，因而无需提供公共服务请求接口。基于 DDS 的飞机协同设计数据服务中间件的应用适配器的体系结构如图 5.2 所示。

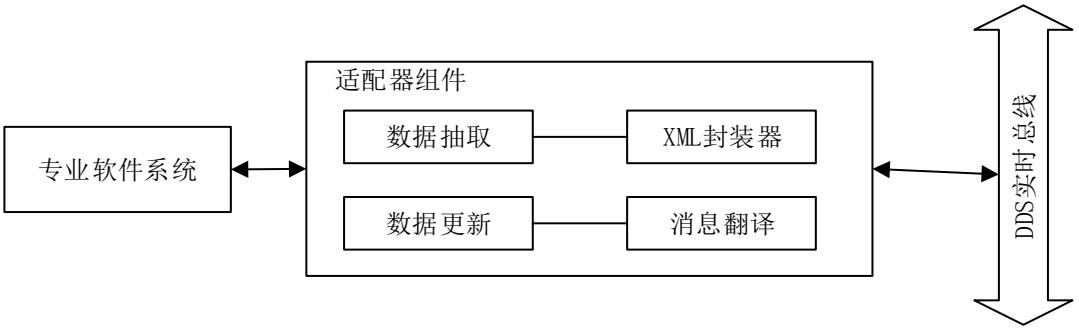

图 5.2 应用适配器结构图

数据抽取模块：用于连接专业软件系统的相关数据文件，主要是数据库。当信息感知组件检测到数据库表对应的数据发生变化时，数据抽取模块负责从数据库中将变化的数据抽取出来，提交给 XML 封装器模块。

XML 封装器：用于把专业软件系统数据封装成飞机协同设计过程中统一的消息格式的消息，并将该消息发布到 DDS 实时总线上。

消息翻译模块：各专业软件系统订阅自己感兴趣的主题，当其他专业软件系统发布该主题时，此时就会从 DDS 实时总线上收到相应的主题信息。消息翻译模块的主要工作就是将统一格式的消息翻译成本专业软件系统所能接受的消息，并交于数据更新模块。

数据更新模块：用于将接收到的新的结果数据更新到本专业软件系统中去。

5.3 ACD_DSM 应用适配器核心技术

各专业软件系统之间交换信息，需要解决消息格式不同等问题。现需要按照某种机制进行转化与翻译工作，将本专业软件系统的消息转化为飞机协同设计系统的统一消息格式，当接收到消息代理组件的消息时，将飞机协同设计系统的统一消息格式转化为本系统所能接受的消息格式。本中间件定义全局统一的消息格式为 XML 消息。

图 5.3 给出消息通过适配器和消息代理组件在各专业软件系之间交互的过程。应用适配器将专业软件系统的私有消息用 XML 封装器转化为统一消息格式，即 XML 消息。然后发布到 DDS 消息总线上；当从 DDS 消息总线接收到 XML 消息，适配器翻译成本系统所能接受的消息。

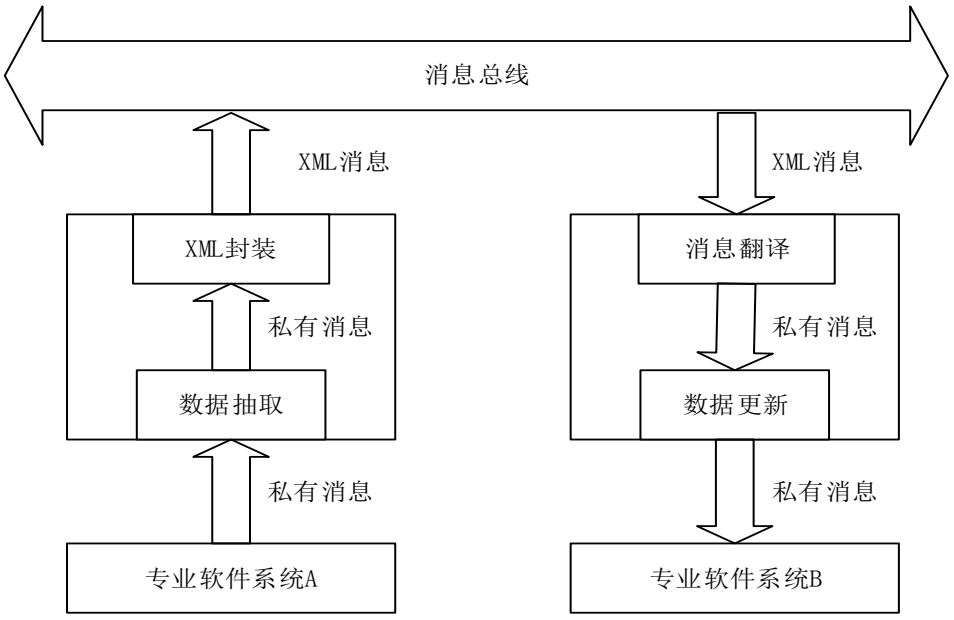

图 5.3 飞机协同设计消息流向

通过以上分析，适配器的核心技术主要有统一消息格式的生成，消息解析，消息翻译，本小节对这三个核心点简要介绍。

5.3.1 统一消息格式生成

信息感知组件对各专业软件系统的数据库的对应数据库表进行监听，当对应表发生数据变化时，就使用适配器对数据库中变化数据进行数据提取，此时需要将数据转化为 XML 消息。因而 XML 封装器的主要功能就是将关系数据库到 XML 的映射转换。

文献[72]、文献[73]和文献[74]分别给出了关系数据模型和 XML 模式的形式化描述和转换规律的研究，并给出了转换规则。飞机设计过程中，每计算出来的一个结果都是很重要的，后面

计算出的结果需要和前面的作比较，因而每种数据的版本会有很多，因而飞机协同设计系统一个很大的技术难点就是数据的版本管理。为了实现数据版本管理（版本管理的具体实现不是本文考虑的问题），数据库表中会有相关字段用于记录版本。同一个数据在专业系统内部和外部是不同的版本号，例如从 A 系统订阅到版本为 a 的数据，而针对该数据，可能会产生许多新的不同版本的数据，这些数据在本专业软件系统中对应不同版本号，而需从中选择一个比较好的发布出去，此时不能用这个系统内部的，需要用版本 a 来标记发布，外部这样才可以建立关联。本小节在以上文献的基础上，结合版本管理的需求，完成了 XML 封装器的设计工作。参照文献[74],需将版本号这一属性特殊对待，因而定义关系数据模型为一个六元组

$$R=(D，T，CR，P，PK，FK)$$

其中：

(1)D 表示将数据库 db 映射为数据库详细信息 d 的函数，D(db)=d，d 为一个四元组，d=(DBN，DBA，VN，VC)，DBN 表示数据源名称，DBA 表示数据源地址，VN 表示访问对象名称，VC 表示访问密码；

(2)T 表示表名字的有序集合，如表 B 的 Foreign Key 是引用表 A 的 Prime Key，那么表 A 在表 B 之前；

(3)CR 表示从表名 t∈T 到一组列 c∈C 的映射，C 表示列名集合；

(4)P 表示将列名 c 映射为列类型Ψ的函数，P(c)=Ψ，这里列类型Ψ为：(w，k，ζ，n，μ，ch,vn)。其属性表示如表 5. 1 所示。

表 5. 1 列类型 Ψ 属性表

元素	含义	值
w	是否是外关键字	~W:不是;W:是
k	是否被引用了的主关键字	~K:不是;K:是
ζ	字段的数据类型	如 char
n	字段是否可空	~N:不可空, N:可空
μ	字段是否唯一	~U:不唯一, U:唯一
ch	域约束	~C:无约束, C:有约束
vn	是否为版本号列	~V:不是, V:是

（5）PK 表示表之间的 PrimeKey 约束；表示表名 t∈T 到一组列 c∈C 的映射，且 PK(t)⊆CR(t)

（6）FK 表示表之间的外键约束，表示为{ci⊆cj}。

使用文献[74]中定义 2. XML 模式为一个六元组

$$S=(NS，Ro，\ Le，Br，type，KR)$$

其中：

(1) NS 表示文档的命名空间信息；

(2) Ro 表示 XML Schema 的 root 节点；

(3) Le 表示 XML Schema 的子节点集合，其中每个元素 Lei 包含了子节点的名称、相关属性的类型、以及用户自定义约束等信息；

(4) Br={Le_1，…，Le_i，…，Le_n}表示一个树干，其中 Le_i 表示叶子节点，树干中的叶子节点之间存在一种嵌套的关系；

(5) type 表示将节点 N 映射为节点类型{Em,Ar}的函数，type(N)={Em,Ar}，其中 N={Lei∪Ro}，Em 是元素的集合，Ar 是属性的集合。

(6) K_R 是有序对（Lei，Lej）的集合，表示 Lei 和 Lej 之间是 key 关键字与 keyref 关键字引用的关系。

下面给出关系数据模型到 XML 模式的映射规则：

映射规则一：确定 XML 模式的命名空间以及目标模式空间等信息，作为 XML 模式头信息描述。存于 NS 中。

映射规则二：将数据库名 DBN 映射为 XML 模式的根元素节点，定义为 complextype，DBA，VN，VC 作为根元素的属性，数据库包含的表作为根元素的子元素。

映射规则三：将关系数据模型中的表名字有序集合 T 映射到 XML 模式的 Br 中，即若 T 中表 A 在表 B 之前，则映射为 Br 中 A 下嵌套 B，B 作为 A 的子元素。

映射规则四：将关系模式 C 中的列信息Ψ映射并存储到 XML 模式的节点 Le 中，其中对于列信息为版本号时，需需要调用版本管理中间件提供的结构进行版本转换。

映射规则五：当外码 w 为 W 时，用属性节点表示，存放于 Le 中，即 type（w）=attr，主码 k 为 K 时，用元素节点表示，存放于 Le 中，即 type（k）=elem;；当要指明对应主表的某个主键时，用扩展元素属性 fref 表示，如“fref=B.id”表示此外键对应表 B 的主键 id；关系模式中的字段的唯一性 u 采用扩展元素属性“unique=yes/no”方式表示；可空 n 为 N 时，用属性 minOccurs=“0”表示；域约束 ch 用属性 check=“条件”表示

映射规则六：将关系模式中的 FK 主键外键之间的关系，映射到 XML 模式 KR 中。

根据上述定义的模型以及映射规则，本文设计的全局模式生成算法分为两个个阶段，统一格式生成算法描述如图 5. 4 所示。

输入：关系数据模型R；

输出：XML模式。

第一阶段：关系模式提取阶段

Step1：从数据资源层数据库中读取数据源中的元数据，

Step2：从元数据中获取关系数据模式中的数据源类型、数据源地址，访问名称、访问密码等

Step3：从本地数据库中获取数据表名T、数据表名和数据列之间的相关约束信息C等信息。

Step4：通过查询数据库系统表获取约束定义，数据列类型P，包括其中的可空性n、唯一性约束μ 等，获取表之间的主键约束PK、外键约束FK；

第二阶段：模式映射阶段

Step5：进行XML模式头描述；

Step6：依据映射规则二创建对应于关系模式的XML文档的根节点，

Step7：依据映射规则三创建根节点的各子节点，形成XML模式树型结构，

Step8：依据映射规则四、五完成各子节点所包含的属性，丰富XML模式树；

Step9：全局统一标准XML模式生成。

图 5.4 统一消息格式算法

5.3.2 消息解析

ACD_DSM 使用 XML 消息作为统一消息格式，因而对消息的解析指的就是对 XML 消息的解析。针对 XML 消息的解析主要是使用 DOM 或 SAX 对 XML 文件展开解析,同时利用 XML Schema 文件对 XML 消息中的数据作验证。

所谓的 DOM 方案就是把 XML 文档保存于一个分级对象节点树当中，而该分级对象节点树能够同 XML 文档的结构和信息开展匹配,这样一来,我们就可以通过查询节点树来获取自己想要的信息。我们使用 DOM 解析器的时候，首先将整个文档全部读入,接着再构建一个具有驻留内存功用的树结构，并使其生成 DOM 树上的全部 Ndoe 节点，其后用户就能够利用 DOM 接口来实现对这个树结构的具体操作。该方案的最大亮点在于文档树整个地留存于内存中，操作简单方便，它还具有支持修改、删除、重新排列等若干功能；而缺点是它需要在内存中调入整个文档(不排除一些无用的节点),因此浪费了有限的内存空间,延长了文档解析的处理时间,降低了文档处理的效率。

而 SAX 则是一种以事件驱动的非重量级的方案，也就是说它在具体应用时无需读入整个文档，而是从解析器发现元素开始、元素结束又或者是文本文档的开始以及结束时再发送事件。接着，借助于编写响应这些事件的代码的方式实现对 XML 数据的处理和保存。该处理方式的好处在于对文档的分析可以随时随刻开始，并不需要等待全部数据的处理。另外,鉴于应用程序仅仅在读取数据的同时进行数据的检查,所以没有必要将全部数据都存储到内存中，因此 SAX 更

适合解析大型 XML 文档。实际上，因为应用程序能够在某个特定条件得到满足时即刻停止解析所以它完全没有解析整个文档的必要。通常来讲,SAX 的速度要领先于 DOM，但不容置喙的是其缺点显而易见，即存储性极差，它往往不会使用任意一种方式对数据做存储处理,因此导致使用 SAX 无法随机进行 XML 文档的访问，在修改数据方面也存在十分较大的难度。

结合飞机协同设计的需求，解析后的信息可能与本系统的私有消息格式不统一，需要转化为本专业软件系统的私有消息，需要用到消息翻译，而消息翻译使用的是 XSL 完成，需要整个文档树，因而我们选择 DOM 作为消息解析方式。

5.3.3 消息翻译

当某专业软件系统接收到远程专业软件系统发布的主题消息时，接收到消息格式为全局统一消息格式，与本专业软件系统的消息格式可能不同，因而需要将该消息格式转化为本专业软件系统所能接收的私有消息格式。数据服务中间件使用 XML 消息作为统一的消息格式，因而消息翻译的主要职责是解决语义上的冲突，我们选用扩展样式语言 XSL 来解决。

扩展样式语言 XSL(extensible Style sheet Language)是以 W3C 作为标准，对 XML 文档执行标准的格式化。主要包括两个标准：格式和转换。格式标准界定了一系列格式对象所适用的格式语义,通过转化标准逐步发展为一种语言,实现从一个 XML 文档到另外一个 XML 文档的转变。这里的语言指的就是扩展样式转换语言 XSLT(eXtensible Stylesheet Language for Transofrmations)。

XSLT 界定了 XML 文档的转换机制。依照样式表里的静态映射信息，如果把某 XML 文档转变为其他的 XML 文档，则可以同时处理完 XML 的显示以及将数据转变成不同的 DTD 或者是 Schema 格式的 XML 文档这两个问题。借助运用 XSL 样式表对这两种模式间的映射的界定,加上辅助实施 XSLT 转换就能够轻松便捷地执行并完成转换。

多数数据共享应用中，人们通常会将 XSLT 转换应用到数据转换的过程当中，而 XSLT 通常根据所需要的数据格式来转换 XML 数据。例如，产生于同一个数据源的 XML 数据标记以及数据的类型应该在一定程度上相似于消费者对数据提出的格式要求，但是也要在某些方面体现出不尽相同。这样，在不更改源数据的前提下，我们可以借助运用 XSLT 模板以及转换器的方式向消费者生成并提供十分有效的数据。或者，如果我们需要较为简洁的数据模式，就可以借助 XSLT 缩减某一数据集，当然也可以采用重新整合源文档的方式。这对于由数据库映射出的 XML 数据集实施处置大有用处。

由于 XML 文档是结构化的,因此，在使用 DOM 解析器分别解析 XML 消息文档和映射文档时，获得消息文档中信息在映射文档中所对应的信息后,依据消息文档格式建立新的 XML 消息文档，就能够完成对 XML 文档内容进行的转化。XML 转换示例如图 5.5 所示。

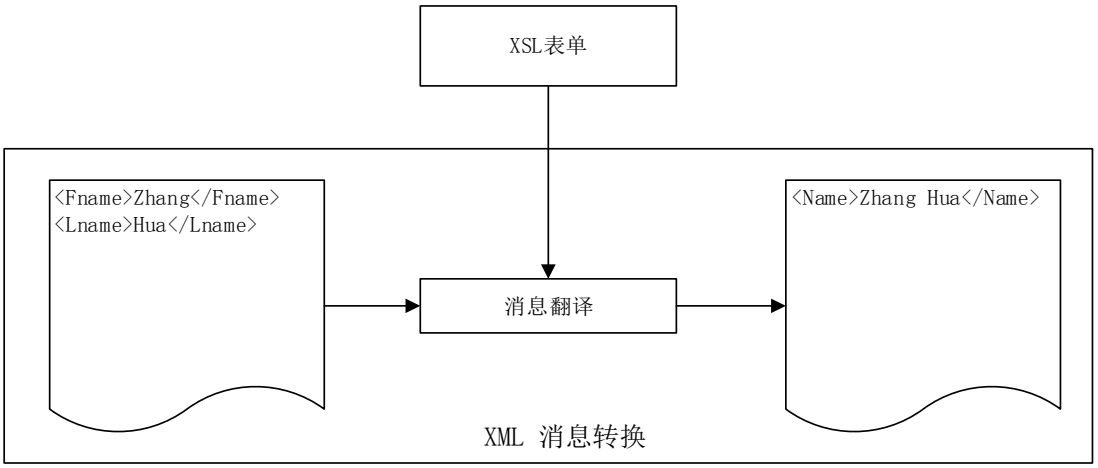

图 5.5 XSL 示例

5.4 本章小结

本章首先对适配器的功能与分类做了简要介绍，其中对传统应用适配器做了分析，然后结合本文的需求，设计了数据服务中间件的应用适配器。最后介绍了应用适配器中的三个核心技术：统一消息格式生成，消息解析、消息翻译。

第六章 ACD_DSM 在飞机协同设计系统中的应用

本文所设计的 ACD_DSM 应用到了飞机协同设计系统中，并取得了良好的效果。它为原有的各学科使用的专业软件系统提供了有效的信息共享方式，有效地提高了各部门之间的协作能力，缩短了飞机设计周期。

6.1 飞机协同设计系统概述

目前计算机相关技术已经在飞机设计领域得到广泛应用，随着飞机设计行业的迅速发展，其设计的相关学科均按照自己学科的特色，结合本学科的业务需求，均已有一些本学科的专业软件。但在飞机设计的过程中，需要多个学科协作共同完成，这些部门单位分布在区域不同，因而部门之间交互信息困难，飞机设计周期一般较长，以年为单位。

分析以上需求，需要解决的最大问题就是现有的各专业软件之间以及设计人员之间的信息交互与共享。因而最好的方案是在保持原有各专业软件系统的自治性的条件下，完成各专业软件之间的信息交互。

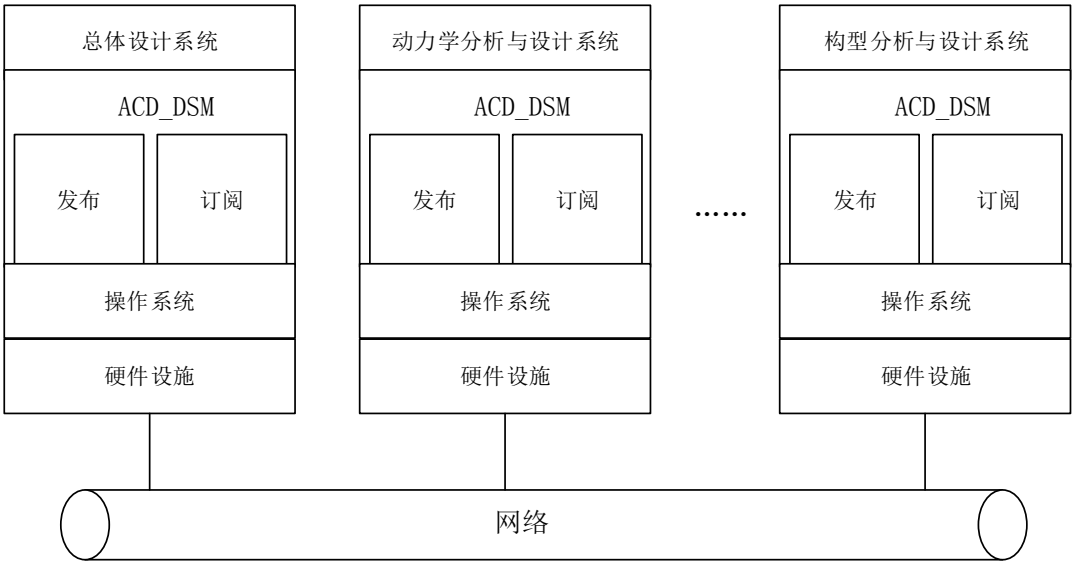

图 6.1 嵌入 ACD_DSM 的飞机协同设计系统示意图

本文设计的基于 DDS 的飞机协同设计中间件嵌入到各专业软件系统后，整个飞机协同设计系统如图 F6.1 所示。飞机协同设计系统为各专业软件系统需要的结果数据构建相应的主题（Topic），当有新的结果数据产生时，由消息感知组件调用适配器，抽取对应数据库表变化的

数据，并将该数据转化为飞机协同设计平台统一的消息格式。然后数据发布者经由数据写入者把 XML 消息发送至全局数据空间当中；如果数据订阅者需要相关的数据信息，可以到全局数据空间中去订阅相关主题，然后通过数据读取者监听并且读取需要的数据，获取消息后进行相应的解析工作，并翻译成本系统所能接受的私有消息，并存入数据库中。本文中通过消息代理组件实现各专业软件系统的信息传递。为了保证服务质量，数据发布者、数据订阅者、数据读取者、数据写入者和主题等可以利用平台提供的 QoS 参数配置服务进行相关的 QoS 参数配置。通过这种方式，各专业软件系统保持了原有的自治性，信息交互的工作交给中间件完成。

6.2 信息感知组件实现

信息感知组件主要功能是对新的结果数据进行感知，为了保持原有专业软件系统的自治性，选用对专业软件系统的数据库进行监测，经过 3.4 小节中对各项技术的比较，本文选用触发器法实现信息感知组件。

为了实现数据的可追溯性，因而飞机设计过程中对于重要的数据是不支持删除或者更新的操作，当有新的结果数据产生时，通过插入添加数据，并由数据版本管理中间件进行版本管理，版本管理不是本文所要处理的问题，因而就不做介绍。通过以上分析，对于需要的结果数据的表仅需要有 Insert 触发器即可。

下面以重量重心软件为例，其编码语言为 C#，使用数据库为 Mysql。因为可能我们需要监控的数据不仅仅在一个表中，为了减少后期的分析，我们额外的新建一张数据表 log，用于记录关注的数据表的更新记录，具体表结构如表 6.1 所示。

表 6.1 数据表 log 表结构

字段	类型	备注
Id	Interger	次序
TableName	Varvhar	操作的数据表
Data	Text	操作的数据

由于触发器运行在数据库服务器上，调用外部脚本或程序要求比较苛刻，因而提供两种方式实现。通用的方法就是定时查询记录表，这种方法不能达到实时性；另一种就是如果数据库支持调用外部函数时，可以设置相应的接口调用，从而实现该功能，使用这种方式，当有新数据产生时，就会立即进行数据的抽取。图 6.1 给出了 Insert 触发器的描述。当对应的数据表产生新的数据时就会向数据表 log 里面插入一条记录，可以用以上方法进行抽取，如果想调用外部函数，需要使用相应的函数，Mysql 可以使用 udf 函数调用函数。

```
DROP TRIGGER trigger_tbls;
CREATE TRIGGER trigger_tbls AFTER INSERT ON hive_metastore.TBLS
FOR EACH ROW
BEGIN
DECLARE @m long varchar;
DECLARE @n integer;
DECLARE @s varchar(5);
DECLARE @rq varchar(10);
SET @s = cast(inserted.number as varchar(5));
SET @rq=cast(inserted.rq as varchar(10));
SET @m = ‘keyname=code,keytype=char(2),
oldcontent=NULL,newcontent=’ +inserted.code+’ ;
Columnname=type,columntype=char(10),
oldcontent=NULL,newcontent=’ +inserted.type+’ ;
Columnname=description,columntype=char(50),
oldcontent=NULL,newcontent=’ +inserted.description+’ ;
Columnname=number,columntype=decimal,
oldcontent=NULL,newcontent=’ +@s+’ ;
Columnname=rq,columntype=datetime,
oldcontent=NULL,newcontent=’ +@rq+’ ;
Select count(*)  into @n from log;
Insert into log(id,tablename,data) values(@n+1,tablename,@m)
 END;
```

图 6.1 Insert 触发器

6.3 消息代理组件的实现

ACD_DSM 的消息代理组件是在 OpenDDS 的基础上实现。OpenDDS 是基于 ACE/TAO 实现的，如果想使用 OpenDDS，首先必须得配置好 ACE/TAO。以下给出配置方法。

6.3.1 配置 ACE/TAO

运用 TAO 可以实现在多种操作系统和多种环境下的系统开发，开发人员可以只需要专注于业务领域本身的实现。下面对 TAO 编译的过程进行详细的论述。

(1) 准备工作

首先我们需要下载并安装 perl 语言的运行环境。Windows 用户可以使用 active perl。

(2) 下载源代码

进入 ACE 的下载页面，点击 Obtainning TAO（获取 TAO）下载链接，开始下载源文件。DOC 提供了各种版本，以及以不同压缩方式的压缩版本。此处我们选择 6.2.1 版本 Zip 文件的发布包，只包含源代码的版本。下载后我们可以得到 ACE/TAO 的压缩包，并将文件解压至硬盘。

(3) 设置环境变量（以 Windows 8 为例）

编译 TAO 之前需要对环境变量进行设置。添加 ACE_ROOT 和 TAO_ROOT。分别设置为

ACE_ROOT = C:\Users\xiongbiao\Desktop\ACE_wrappers\ ; TAO_ROOT = C:\Users\xiongbiao\Desktop\ACE_wrappers\TAO 。修改 Path 为 PATH = %ACE_ROOT%/bin;%ACE_ROOT%/lib;

(4) 配置编译选项

打开到目录%ACE_ROOT%/ace 下，添加一个同文件，命名为 config.h， 图 6.2 给出了里面的文件内容。

```
#include "ace/config-win32.h"
  //config.h内容如下
     // -*- C++ -*-
#ifndef ACE_CONFIG_H
#define ACE_CONFIG_H
#define ACE_DISABLE_WIN32_ERROR_WINDOWS
#define ACE_DISABLE_WIN32_INCREASE_PRIORITY
#define ACE_HAS_MFC 1
#include "ace/config-win32.h"
#endif /* ACE_CONFIG_H */
```

图 6.2 config.h 描述

(5) 编译生成

编译顺序是先 ACE，完成之后再编译 TAO 工程，建议选择命令行的形式来编译，如果直接使用 VS2010 编译时时间会很久。 编译 ACE，首先打开 Visual Studio 2008 命令提示框（类似 CMD，路径为 程序->Microsoft Visual Studio 2008->Visual Studio Tools->Visual Studio 2008 命令提示）,然后输入命令 msbuild %ACE_ROOT%/ace/ace_vc9.sln “Debug|Win32”。编译 TAO ， 先 编 译 %TAO_ROOT%/TAO_IDL/TAO_IDL_vc9.sln, 然 后 再 编译%TAO_ROOT%/tao/tao_vc9.sln。也可以直接使用一下批处理文件直接编译，编译语句如图 6.3 所示。

```
cd D:\Program Files (x86)\Microsoft Visual Studio 9.0\VC\vcpackages

vcbuild.exe /rebuild  %ACE_ROOT%\ace\ACE_vc9.sln "Debug|Win32" >c:\ace_log.txt
vcbuild.exe /rebuild  %ACE_ROOT%\apps\gperf\src\gperf_vc9.vcproj "Debug|Win32"
vcbuild.exe /rebuild  %ACE_ROOT%\TAO\TAO_IDL\TAO_IDL_vc9.sln "Debug|Win32"
vcbuild.exe /rebuild  %ACE_ROOT%\TAO\tao\tao_vc9.sln "Debug|Win32"
vcbuild.exe /rebuild  %ACE_ROOT%\TAO\orbsvcs\orbsvcs_vc9.sln "Debug|Win32"
vcbuild.exe /rebuild  %ACE_ROOT%\TAO\TAO_ACE_vc9.sln "Debug|Win32"
vcbuild.exe /rebuild  %DDS_ROOT%\DDS_vc9.sln "Debug|Win32"
vcbuild.exe /rebuild  %DDS_ROOT%\dds\dds_vc9.sln "Debug|Win32"
%ACE_ROOT%\bin\mwc.pl -type vc9 -features qt4=1
```

图 6.3 ACE/TAO 编译 bat 文件

6.3.2 配置 OpenDDS

在配置了 ACE/TAO 的基础上，OpenDDS 的配置相对比较简单。首先需要配置环境变量 DDS_ROOT，设置 DDS_ROOT 的路径为 DDS 的所在路径，然后再 path 中添加%DDS_ROOT%/bin 和%DDS_ROOT%/lib。最后使用编译器编译编译 DDS 的相应的工程即可。此处建议不直接使用 visual studio 编译，因为工程中包含项目太多，编译速度很慢，如果使用批处理文件编译速度会快很多，具体语句参见 ACE/TAO 批处理文件。

6.3.3 发布订阅过程实现

发布订阅的过程主要需要完成以下步骤，通过这些步骤可以将本来没有联系的发布者和订阅者之间产生一种逻辑关联的关系，从而实现发布者和订阅者之间的通信，完成了发布订阅的实现。

1. 首先，创建一个 MessageTypeSupportImpl 对象，然后使用 register_type( )操作与类型名称注册类型。

```
Messenger::MessageTypeSupport_var mts = new Messenger::MessageTypeSupportImpl();
if (DDS::RETCODE_OK != mts->register_type(participant, ""))
{
        std::cerr << "register_type failed." << std::endl;
        return 1;
   }
```

2. 接下来，从类型支持对象获得注册的类型名称，并通过将类型名称到参与者的 create_topic()操作创建主题。

```
CORBA::String_var type_name = mts->get_type_name ();
DDS::Topic_var topic =
participant->create_topic ("Movie Discussion List",
                           type_name,
                           TOPIC_QOS_DEFAULT,
                           0, // No listener required
                           DCPS::DEFAULT_STATUS_MASK)
```

3. 已经准备好创建与默认的发布 QoS 发布者。

```
DDS::Publisher_var pub =
    participant->create_publisher(PUBLISHER_QOS_DEFAULT,
                                  0, // No listener required
```

```
                                        DCPS::DEFAULT_STATUS_MASK);
```

4. 有了发布者，创建数据的写者。

```
// Create the datawriter
DDS::DataWriter_var writer =
        pub->create_datawriter(topic,
                               DATAWRITER_QOS_DEFAULT,
                               0, // No listener required
                               DCPS::DEFAULT_STATUS_MASK);
```

当创建的数据写入者通过主题对象的引用，默认的 QoS 策略，以及一个空监听器的参考。现在缩小数据写入引用 MessageDataWriter 对象的引用，所以可以使用特定类型的发布业务。

```
Messenger::MessageDataWriter_var message_writer =
        Messenger::MessageDataWriter::_narrow(writer);
```

5. 创建域参与者工厂以及参与者的创建。

```
 DDS::DomainParticipantFactory_var dpf = TheParticipantFactoryWithArgs(argc, argv);
 DDS::DomainParticipant_var participant =
        dpf->create_participant(10, // Domain ID
                                PARTICIPANT_QOS_DEFAULT,
                                0, // No listener required
                                OpenDDS::DCPS::DEFAULT_STATUS_MASK);
```

6. 接下来，初始化消息类型和主题。如果该主题已经在这一领域进行初始化具有相同的数据类型和兼容的服务质量，在 create_topic()调用返回对应于现有主题的参考。如果在 create_topic()调用中指定的类型或 QoS 不匹配，现有的主题则调用失败。还有一个 find_topic()操作用户可以用它来简单地检索现有的主题。

```
 Messenger::MessageTypeSupport_var mts = new Messenger::MessageTypeSupportImpl();
 if (DDS::RETCODE_OK != mts->register_type(participant, ""))
 {
        std::cerr << "Failed to register the MessageTypeSupport." << std::endl;
        return 1;
 }
 CORBA::String_var type_name = mts->get_type_name ();
 DDS::Topic_var topic =participant->create_topic("动力学数据",
                                                 type_name,
```

```
                                                    TOPIC_QOS_DEFAULT,
                                                    0, // No listener required
                                                    DCPS::DEFAULT_STATUS_MASK);
```

7. 创建订阅者。

```
// Create the subscriber
  DDS::Subscriber_var sub =
        participant->create_subscriber(SUBSCRIBER_QOS_DEFAULT,
                                       0, // No listener required
                                       DCPS::DEFAULT_STATUS_MASK);
```

8. 现在，可以创建数据读取器，并创建侦听器对象进行关联。

```
// Create the Datareader
  DDS::DataReader_var dr =sub->create_datareader(topic,
                                                    DATAREADER_QOS_DEFAULT,
                                                    listener,
                                                    DCPS::DEFAULT_STATUS_MASK);
```

9. 监听器类实现了 DDS 规范中定义的 DDS:: DataReaderListener 接口。该 DataReaderListener 是 DCPS:: LocalObject 内包裹它解决隐约继承成员。该接口定义了一些操作必须实现，每个调用，通知不同的事件。该 DCPS:: DataReaderListener 定义了特殊需求，如断开和重新连接事件的更新操作。

```
void DataReaderListenerImpl::on_data_available(DDS::DataReader_ptr reader)
{
        num_reads_ ++;
        try {
                Messenger::MessageDataReader_var reader_i =
                    Messenger::MessageDataReader::_narrow(reader);
                if (!reader_i)
                {
                        std::cerr << "read: _narrow failed." << std::endl;
                        return;
                }
    }
```

10. 下面的代码会从消息读取下一个样品。如果获取成功，返回有效数据，然后输出每个

消息的字段。

```
Messenger::Message message;
DDS::SampleInfo si ;
DDS::ReturnCode_t status = reader_i->take_next_sample(message, si) ;
if (status == DDS::RETCODE_OK) {
        if (si.valid_data == 1) {
            std::cout << "Message: subject = " << message.subject.in() << std::endl
                      << " subject_id = " << message.subject_id << std::endl
                      << " from = " << message.from.in() << std::endl
                      << " text = " << message.text.in() << std::endl;
        }
 }
```

6.4 适配器消息翻译模块实现

飞机协同设计系统为每个专业软件需要交互的数据定义不同的主题，每种主题的信息都是从数据库中提取出后转化为 XML 格式的信息，但可能语义上与本专业软件系统不符合，此时需要使用 XSL 进行相应的转化。本小节以 5.3.3 小节的示例，给出消息翻译的示例。

从图 5.5 可以看出，xsl 需要完成的工作是将 Fname 属性和 Lname 属性里面的值放入 Name 属性中，并在两个值之间加入空格。完整的 XSL 文件图 6.4 所示。

```
<?xml version=”1.0” encoding=”UTF-8”?>
<xsl:stylesheet version=”1.0”
xmlns=”http://www.w3.org/1999/XSL/Transform”>
<xsl:template match=”/Name”>
        <Name>
                <xsl:value-of select=”Fname”/>
                <xsl:text></xsl:text>
                <xsl:value-of select=”Lname”/>
        </Name>
</xsl:template>
</xsl:stylesheet>
```

图 6.4 XSL 文件

将以上文件命名为 Transform.xsl。然后仅需要在原 XML 文档中应用该语句即可，具体语句为<?xml-stylesheet type=”text/xsl” href=”Transform.xsl”?>。

6.5 系统运行测试

本小节给出基于 DDS 的飞机协同设计数据服务中间件的运行结果。首先需要配置数据服务的各参数，具体配置界面如图 6.5 所示。需要配置的信息主要有本地的数据库连接信息，轮询时间间隔，日志存放路径，以及 DDS 的一些配置属性。数据库连接信息用于连接本地数据库使用，从而达到监控本地数据的目的；轮询时间的设定用于对数据表 log 查询的时间，如果时间间隔设为 0，此时使用的是触发器直接调用脚本完成；日志存放路径用于设置系统的日志存放路径；DDS 相关的配置属性主要有通信域的设定以及 QoS 的设定。通信域的设定决定了该节点只能与该域中的节点通信，而此处的 QoS 设定主要是设置默认 QoS，便于其他时候调用。

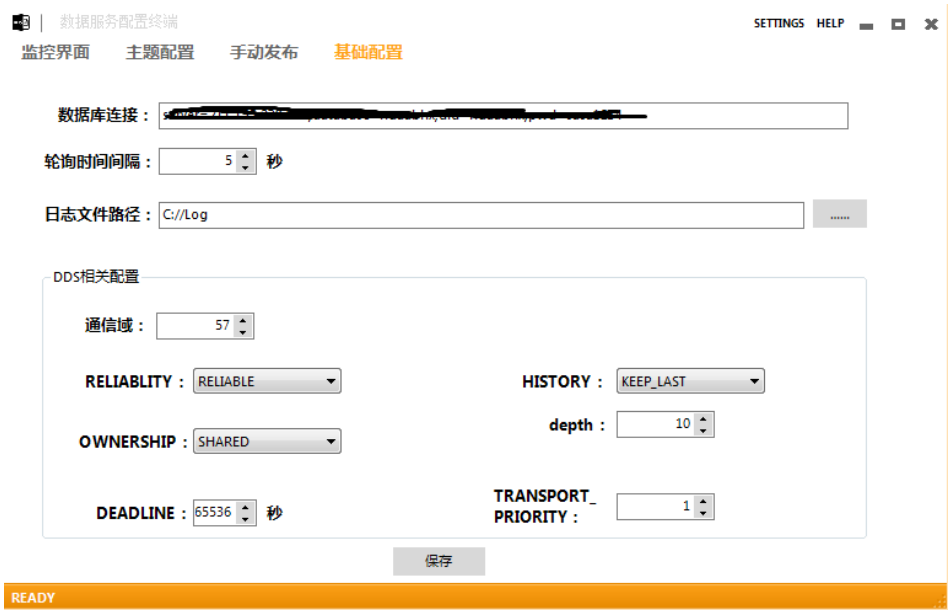

图 6.5 基础配置界面

图 6.6 所示的是主题配置界面。在该界面下可以完成主题的创建，修改，删除的工作。主题的相关属性主要包括主题名，操作类型，QoS,该主题对应的数据库表，以及如果是订阅的话，可能需要上传想用的 XSL 文件用于转换 XML 消息。

图 6.6 主题配置界面

图 6.7 所示的手动发布界面，对于有些主题消息不是数据库里面的消息，此处可提供发布的方式。主题为主题配置界面的主题，我们通过下拉框选择已经创建的主题。消息正文必须按照 XML 消息格式编写，发布之前会进行格式检查。

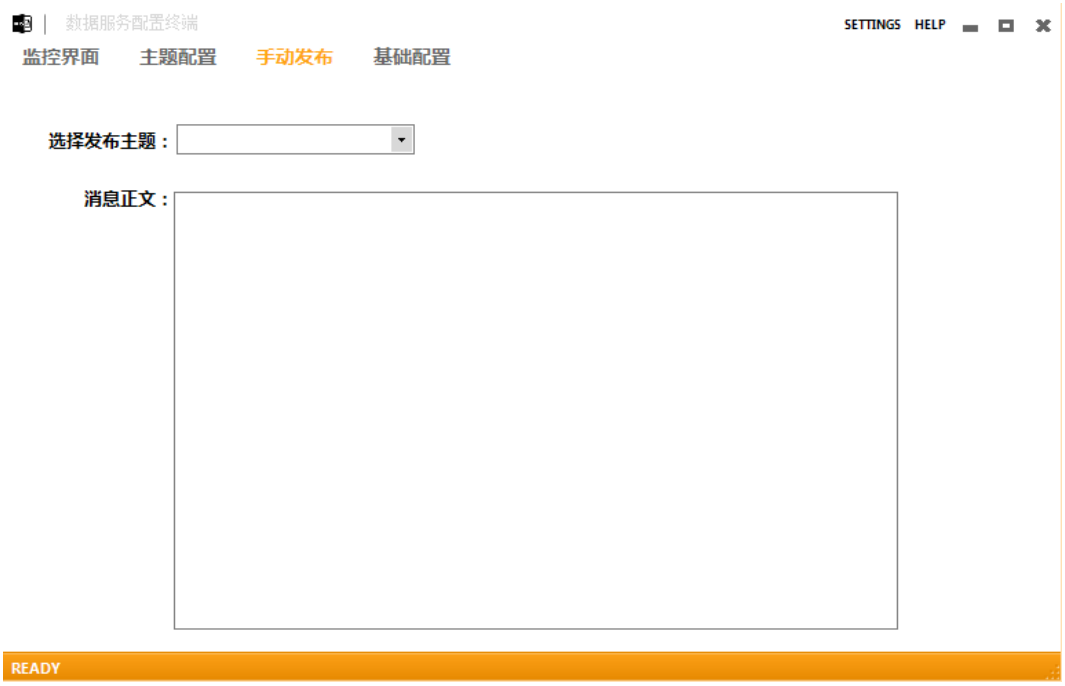

图 6.7 手动发布界面

图 6.8、图 6.9 所示是实现信息监控的界面，分别展示了当有新的数据产生进行信息发布

的界面和接收到订阅的主题信息时的界面。还可以通过查看日志文件按钮，查看保存的日志文件。

图 6.8 监控界面（发布）

图 6.9 监控界面（订阅）

6.6 总结

本文对 DDS 进行了深入的研究， 并针对飞机设计过程中，现有的各专业软件系统之间信息交互困难的问题，设计了一个轻量型的 ACD_DSM，并成功应用到飞机协同设计系统中，主要研究工作包括以下几个方面：

1) 数据分发服务技术的研究。主要研究了 DDS 相关标准、体系结构以及模块划分；并深入分析了 DDS 的开源实现 OpenDDS 的全部代码，并将其改进后应用到数据服务架构中。

2) 飞机协同设计数据服务中间件的设计。本文结合飞机协同设计的需求，设计出具有可扩展性、松耦合的数据集成框架。秉持灵活以及可扩展的原则，发挥 DDS 的特性，构建了基于 DDS 的飞机协同设计数据服务中间件框架。

3) 发布/订阅自动发现机制的研究以及改进。论文工作研究了 DDS 规范中基于 SDP 协议的发现协议、基于 Bloom Filter 改进的自动发现协议 SDPBloom，提出了一种基于 DCF 的自动发现协议,设计了相关算法，并对算法进行了性能分析和实验验证，实验结果表明，设计的局域 DCF 的数据分发自动发现协议有效地减少了网络负载和内存消耗。

4) 采用 XML 相关技术设计并实现了中间件的适配器组件部分，解决了现有的各专业软件系统的数据格式不统一的问题。

5) 基于 DDS 的飞机协同设计数据服务中间件的应用。本文设计的基于 DDS 的飞机协同设计数据服务中间件已在飞机协同设计系统中得到初步应用。测试实验结果表明该数据服务中间件能够满足飞机设计各专业软件信息交互的要求;还通过实验验证了数据分发服务 DDS 应用到飞机协同设计中的有效性。

6.7 后期研究工作

现已经初步实现了基于 DDS 的飞机协同设计数据服务中间件，虽然已经可以满足各专业软件系统之间的信息交互的需求，但还有以下几点需要改进：

1） 现在主要处理的数据是数据库数据，其他类型数据需要通过手动发布，随着飞机设计技术的发展，后期的需求会很多，如何应对其他类型文件是需要解决的问题。

2） 信息感知组件现在采用的是触发器法，占用系统资源比较多，并且调用外部脚本或程序，要求比较苛刻，是否有其他方式可以替代。

3） 本文采用 XML 技术实现了使用适配器，现在应用适配器部分还没有提供公共访问接口，主要还是通过信息感知组件对数据库数据进行监控，从而调用适配器处理，不接受其他方式的请求。后期随着需求的增多，应用适配器应提供相应的公共服务请求接口，以供外部调用。

参考文献

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

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

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

[4] 王凯. 面向飞机总体布置的协同设计关键技术研究 [博士学位论文]; 南京航空航天大学, 2010.

[5] 王磊. 飞机协同设计应用技术研究[硕士论文];西北工业大学，2007

[6] 付广磊. 飞机主机单位协同研制流程研究 [硕士学位论文]. 西北工业大学, 2007.

[7] Zachary G.Ives. Efficient Query Processing for Data Integration [Dissertation].Washington: Washington University,2002.

[8] 李俊,李勇.联邦式异构数据库应用系统的集成框架和实现技术的研究.计算机应用研究,2001,18(4):19-22.

[9] Inmon WH.Building the data warehouse.Canada:JohnWiley&Sons Ine ,1992.

[10] 张宁.数据仓库中 ETL 技术的研究[J].计算机工程与应用,2005(24):213-216.

[11] OMG, Common Object Request Broker Architecture: Core SpecificationVersion3.0.3 formal/04-03-01[S],March 2004.

[12] 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.

[13] 赵晓君. 基于 JMS 和 XML 的异构数据库集成研究 [硕士学位论文]; 武汉理工大学, 2005.

[14] 王建. 浅谈消息中间件 IBM WebSphere MQ [J],微型机与应用. 2010.

[15] 王楠.Web Service 技术研究[J].计算机与数字工程.2006.

[16] 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.

[17] 秦耀坤. 用 TongIntegrator 实现电子商务中的系统集成[J].现代电子技术.2004.

[18] 王东.InforBroker 在国税应用整合中的应用[J].计算机工程与科学。2006.

[19] 数据分发服务指南与规范[S],中华人名共和国科学技术部. 2005.

[20] RTI Connext Core Libraries and Utilities User’s Manual Version 5.0, Real-Time Innovations, Inc. 2012.

[21] 张珺，尹逊和. 基于 RTI DDS 的数据分发中间件的升级设计[J].北京交通大学学报. 2011, 35(5):

31-37.

[22] Pardo-Castellote G, Schneider S, Hamilton M. NDDS: The real-time publish-subscribe middleware[J]. Real-Time Innovations, Inc, 1999.

[23] 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.

[24] Computing O. Inc. OpenDDS Developer's Guide[J]. 2009-06-10. http://download. ociweb. com/OpenDDS/OpenDDS-latest. pdf.

[25] 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.

[26] Milsoft DDS Middleware And Using dds For Simulation In Combat Management Systems .http://dds.milsoft.com.tr/en/dds-related-documents.php.2007.

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

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

[29] 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.

[30] 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.

[31] Bloom B H. Space/time trade-offs in hash coding with allowable errors [J]. Communicatio- ns of the ACM, 1970, 13(7): 422-426.

[32] 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.

[33] 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.

[34] 翟海波.基于服务力向量向量的发布/订阅自动发现算法[J].2014

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

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

[37] 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.

[38] 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 (012): 185-187.

[39] 姚兵，蔡婷，李峻林，等. 基于 DDS 模型的数据分发中间件的设计与实现[J]. 计算机工程与设计, 2009, 30(3): 619-623.

[40] 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.

[41] 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.

[42] Giddings V. Tutorial on the OMG data distribution service[R]. Objective Interface Systems, Inc, 2005.

[43] 张大海，赖兰剑，陈鼎才. DDS 在分布式系统仿真中的应用[J]. 计算机技术与发展, 2011, 21(3): 250-253.

[44] Sheth P, Larson A, Federated database system for managing distributed, heterogencous and autonomous database.ACM Computing Surveys,1990,22(3):183 一 236.

[45] W3C Extensible Markup Language (XML) 1.0 Second Edition. W3C Recommendation 6 October 2000. http://www.w3.org/TR/REC- XML.

[46] 段友祥,岳厚光.基于 XML 的异构多数据源信息访问.微型机与应用,2002,12(6):30-32.

[47] Gerardo P C. OMG Data-Distribution Service (DDS):Architectural Update [C]. 2004 IEEE Military Communications Conference, 2004, 2:961~967.

[48] 刘晓萍,触发器在数据库中的应用[J]. 韶关学院学报,2007.

[49] W3C. Simple Object Access Protocol(SOAP) 1.1. http://www.w3.org/TR/20OO/NOTE-SOAP-20000508/.2009.

[50] W3C. WebServices Description Language (WSDL) 1.1. http://www.w3.org/TR/2001/NOTE-wsdl-20010315.2009.

[51] OASIS. Universal Description , Discovery,and Integration (UDDI) 3.0. http://www.oasis-open.org/committees/uddi-spec/doc/spce/v3/uddi-v3.0.2-20041019.htm.2010

[52] 刘伯超,马晓轩,葛声等.基于 Web 服务的软件服务体系结构的研究与实现.北京航空航天大学学报.2004,3(第 30 卷第 3 期).pages263-266.

[53] Bemstein, PhiliP. A: Middleware: A Model for Distributed Services. Communications of the ACM Volume 39 Issue 2. l996.Pages86-97.

[54] 魏勇,张权.中间件技术研究.国防科技大学电子科学与工程学院.电子技术应用.200430(11).Pages 1-4.

[55] Xiong Gang, XiongGang-Yu, Litokorpi Aki, et al. Middleware-based solution for enterprise

information integration. IEEE SymPosium on Emerging Technologies and Factory Automation, ETFA. 2001(2).Pages687-690.

[56] 王军.基于 JMS 的消息中间件设计与实现.计算机应用.2003,23(8).Pages 64-67.

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

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

[59] 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.

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

[61] Fan L, Cao P, Almeida J, et al. Summary cache: a scalable wide-area web cache sharing protocol[J]. IEEE/ACM Transactions on Networking (TON), 2000, 8(3): 281-293.

[62] 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.

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

[64] 严华云, 关佶红. Bloom Filter 研究进展[J]. 电信科学, 2010, 13 (2)：139-142.

[65] 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.

[66] Javier S´anchez Monedero. A DDS Discovery Protocol based on Bloom filters[MASTER THESIS]. Universidad de Granada.2009.

[67] Bonomi F, Mitzenmacher M, Panigrahy R, et al. An improved construction for counting bloom filters[M]//Algorithms–ESA 2006. Springer Berlin Heidelberg, 2006: 684-695.

[68] Aguilar-Saborit J, Trancoso P, Muntes-Mulero V, et al. Dynamic count filters[J]. ACM SIGMOD Record, 2006, 35(1): 26-32.

[69] A. Partow, General Purpose Hash Function Algorithms, 2009. http://www.partow.net/programming/hashfunctions/index.html.2014.

[70] W. Foundation Wireshark, 2010. http://www.wireshark.org/.2014.

[71] Jeff Sutherlnad, Willem 一 Jan van den Heuvel. Enterprise Application Integration Encounters Complex Adaptive Systems: A Business Object Perspective. Porceedings of 35th Hawaii International Conference on System sciences 一 2002,IEEE

[72] 史晔翎, 黎建辉. 关系数据库模式到 XML Schema 的通用映射模型 [J]. 计算机工程, 2009, 35(7):

[73] 侯莹, 李凤岐, 牛纪桢, et al. 关系模式到模块化的 XML Schema 的模型映射方法 [J]. 计算机工程与应用, 2011, 47(12): 122-125.

[74] Yuan Y, Zhuang Y, Huo Y. Research and Application of Data Integration in Aircraft Designing Based on SDO[J]. Journal of Software, 2014, 9(3): 689-696.

[75] Ramachandran K K, Sikdar B. A queuing model for evaluating the transfer latency of peer-to-peer systems[J]. Parallel and Distributed Systems, IEEE Transactions on, 2010, 21(3): 367-378.

[76] Pérez Tijero H, Gutiérrez J J. On the schedulability of a data-centric real-time distribution middleware[J]. Computer Standards & Interfaces, 2012, 34(1): 203-211.

[77] 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.

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

[79] 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.

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

[81] Fan L, Cao P, Almeida J, et al. Summary cache: a scalable wide-area web cache sharing protocol[J]. IEEE/ACM Transactions on Networking (TON), 2000, 8(3): 281-293.

[82] 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.

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

[84] 余雄庆, 徐惠民, 昂海松. 飞机总体设计 [M]. 航空工业出版社, 2000.

[85] 詹盼盼. 一种面向服务的消息中间件的研究与实现 [D]; 西安电子科技大学, 2010.

致 谢

两年多的研究生时光转瞬即逝，可初来乍到时的场景依旧记忆犹新，不得不慨叹光阴似箭、岁月如梭。回望逝去的日日夜夜，回忆旧时光里的点点滴滴，一路得失相伴、喜忧并行。研究生生涯终结的号角即将吹响，故借此论文完成之际，特意向我的良师益友、至亲故交致以最诚挚的感谢！

最深的谢意献给我的导师陈丹副教授和庄毅教授。在两位老师的温情关怀与悉心指导下我顺利地完成了研究生学业，并完成此论文的撰写。在学业上的进步和科研中的成就都与导师的言传身教息息相关。在生活中，她们待人宽厚、和蔼可亲，时刻为我们排忧解难；在科研工作中，她们细致严谨、求实创新，为我们开辟更宽广的求知空间，她们将是我人生中永远的导师。其次，真诚地感谢各位审稿老师。您的批评和建议是我论文质量进一步提高的保障，您的鼓励和肯定是我继续学习前进的动力，您的严格要求是我一生的追求。

再者，感谢师兄师姐给予我研究方向和论文撰写方面的建议和指导，感谢陈乙睿、叶重阳、郝刚、牛涛、朱建荣同学陪我度过短暂而又充实的研究生生活，感谢师弟师妹对我研究工作的支持和帮助，与你们在一起的每一分钟都将是我此生最美好的回忆。

最后，还要将内心深处最厚重的感激奉送给我的父亲和亲人，你们一如既往的支持与鼓励是我前行道路上的动力，你们对我的关爱是我战胜困难的中坚后盾，感谢成长的路上有你们一路相伴，希望你们此生幸福同行、安康永伴。

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. Bian H X, Xia Z X. Applied-Information Technology in Improving Bloom Filter-Based Discovery Protocol for DDS Middlewarein Consideration of QoS[C]//Advanced Materials Research. 2014, 1046: 457-460. 第一作者，已发表。
2. Bian H X, Zhuang Y, Chen D. Research on Data Distribution Service for Aircraft Collaborative Design System[C]. Communcations in Control Science and Engineering.第一作者，已录用。
3. 卞华星，陈丹，庄毅.DDS 在飞机协同设计系统中的应用[J],计算机与现代化，2015，第一作者，已录用。
4. 卞华星，庄毅，陈丹.基于 Dynamic Count Filters 的数据分发服务发现协议[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 2 月至 9 月，参与“飞机协同设计系统的研发”项目，主要负责数据部分，研究摸索 DDS 以及开源项目 OpenDDS。
2. 2014 年 2 月至 6 月，参与“某飞机飞行力学评估软件”项目，参与项目的关键技术研究，完成数理模型模块代码的编写，并顺利通过了项目的中期评审。
3. 2013 年 10 月至 2014 年 1 月，参与“飞机总体参数优化评估软件”项目，参与需求的讨论和关键技术的研究，参与软件的框架设计和代码的编写。