论文题目 基于发布/订阅机制的传感器数据分发系统的设计与实现

工程领域	软　件　工　程
指导教师	朱　红　副教授
作者姓名	徐　涛
学　　号	200890209069

分类号_____密级 _____

UDC [注1] _____

学　位　论　文

基于发布/订阅机制的传感器数据分发系统的设计与实现

徐　涛

（作者姓名）

指导教师姓名　朱　红　副教授

电子科技大学　成　都

付成群　副教授

工程兵工程学院　南　京

（职务、职称、学位、单位名称及地址）

申请专业学位级别　硕士　专业学位类别　工 程 硕 士

工程领域名称　软　件　工　程

提交论文日期　2011.3　论文答辩日期　2011.5

学位授予单位和日期　电 子 科 技 大 学

答辩委员会主席_____

评阅人 _____

2011 年　　月　　日

注 1：注明《国际十进分类法 UDC》的类号

独 创 性 声 明

本人声明所呈交的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。据我所知，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，也不包含为获得电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

签名：_____ 日期： 年 月 日

论 文 使 用 授 权

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技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应遵守此规定）

签名： 导师签名：

日期： 年 月 日

摘 要

随着传感技术、无线通信技术和计算机技术的不断革新，传感器网络近年来得到了大力发展。因传感器网络是由大量的具有通信和计算能力的低功耗微小传感器节点组成,可用于实时感知和采集网络覆盖区域内的感知对象的信息。因此，在军事国防、医疗健康、环境监测、抢险救灾、防恐反恐、动物习性以及交通流量监测等领域都具有广阔的应用前景，特别适合应用于人不宜到达的恶劣环境和危险区域。西方一些军事强国，则将传感器网络技术广泛应用于军事信息网中。国内各个领域虽有相关应用，但存在着传感器单一，探测距离有限等不足，不能有效地充分挖掘其应用潜力。针对此问题，本论文提出一种基于发布/订阅机制的传感器数据分发系统，并以该方案应用于某军事信息网为例，论述了该传感器数据分发系统的主要功能。

本文首先分析了传感器数据分发系统的应用背景，针对军事信息网带宽资源紧张的现状，提出“面向业务，按需分发”的设计思路。传感器数据分发系统需要具备数据源自动发现和动态扩展能力，从而实现传感器数据实时按需分发的服务自动发现、动态扩展和事件过滤特性，使其适合用于设计和实现传感器的实时按需分发。

其次，以面向服务框架 SOA 为软件设计指导思想，以发布/订阅机制为基础，通过分析传感器数据实时分发业务流程，将传感器数据分发系统分解为注册服务、注册匹配服务、发布服务和订阅服务。针对系统按需实时分发要求，充分利用 DDS 规范面向数据的特性，灵活运用其 Qos 策略，对基于主题的发布/订阅机制进行了内容方式的扩展，实现更小粒度的按需分发；采用非中心直连通信架构，实现数据实时分发。

再次，在 X86/WindowsXP 平台上移植了 ACE、TAO 和 OpenDDS 中间件，构建了软件中间件层。在中间件层上实现了注册服务、注册匹配服务、发布服务和订阅服务等主要服务和功能组件，使用这些服务和功能组件开发了注册服务器、传感器代理、作战子网网关软件。

最后，验证了发布/订阅机制应用于传感器数据分发系统的可行性，为下一步的工作打下基础。

关键词： 发布/订阅；按需分发；中间件；DDS 规范

Abstract

Recently, traditional platform-centric warfare is changing to Network Centric Warfare. Such transition brings great opportunities to modernize the existing combat system. Tactical Data Link (TDL) and the Tactical Internet (TI) play an increasingly important role in the modern war. But they have some deficiencies, such as limited detection range, single-sensor, and so on. Such deficiencies make the TDL and TI uncompatable for long-range early warning and combat mission coordination. The potential of sensors which have been deployed widely, aren't fully tapped. To advance TDL and TI 's operational effectiveness, the deployed sensors' data are disseminated to the TDL and TI through the Military Information Network (MIN).

In my thesis, the background of Sensor Data Dissemination System (SDDS) is analyzed firstly. To encounter the problem of limited bandwidth of MIN, the idea of "Business-oriented, On-demand Dissemination" is introduced to design the SDDS. SSDS is required to have the feature of scalability and the capability of discovering data source dynamically and automatically, which enable the SDDS to disseminate sensor data real-time and on-demand. Because the Publish/Subscribe mechanism has the features above, it is chosen in my work to design and implement the SDDS.

Second, the Service-oriented Architecture (SOA) is adopted as the guidance for software design, and the Publish/Subscribe mechanism is the basis for the system. By analyzing the processes of dissemination business, the system is divided into four dominant services: Registry Service, Registry Matching Service, Publishing Service and Subscribing service. According to real-time and on-demand disseminating requirements, with Data-centric characteristics and Qos policies of DDS, the Topic-based Publish/Subscribe mechanism is expanded with the Content-based style. This make the SDDS obtain fine-granularity and on-demand matching ability, which can locate sensors more suitably. With the decentralized communication framework, the system can disseminate the sensor data in real-time.

Third, with migrating the ACE, TAO and OpenDDS middlewares to X86/WindowsXP platform successfully, a software middleware layer is built. Based on

such middleware layer, the Registry Service, Registration Matching Service, Publishing Service, Subscribing Service, other services and some components is implemented. Based on these services and components, the Register Server, Sensor Agent, and Operational Subnet Gateway's software are implemented too.

Finally, fundamental functions and performances for the SDDS are tested, the feasibility of using the Publish/Subscribe mechanism in the SDDS is validated.

Key word: Publish/Subscribe; On-demand Dissemination; Middleware; DDS

目 录

第一章 绪 论.....1
1.1 课题的背景.....1
1.2 课题的目标.....5
1.3 课题的主要工作.....6
1.4 论文的组织.....6
第二章 系统应用和发布/订阅研究.....8
2.1 数据分发系统应用场景.....8
2.1.1 传感器.....8
2.1.2 作战子网.....10
2.1.3 IP 骨干网.....11
2.1.4 系统需求.....12
2.2 基于 SOA 的发布/订阅.....13
2.2.1 面向服务框架 SOA.....13
2.2.2 发布/订阅基本原理.....14
2.2.3 发布/订阅的特点.....15
2.2.4 发布/订阅相关技术.....16
2.3 实时数据分发规范－DDS.....18
2.3.1 DDS 背景.....18
2.3.2 DDS 结构.....19
2.3.3 基本概念.....20
2.3.4 Qos 策略.....20
2.4 当前研究现状.....23
第三章 传感器数据分发系统设计.....25
3.1 系统设计思路.....25
3.2 系统软件结构.....26
3.3 发布/订阅框架设计.....28
3.3.1 事件声明.....28
3.3.2 匹配模式.....30

3.3.3 通信架构 31
3.3.4 小结 32
3.4 应用业务和功能 33
3.4.1 应用业务 33
3.4.2 注册服务器 35
3.4.3 传感器代理功能 35
3.4.4 子网网关功能 35
3.5 服务定义和分解 36
3.5.1 数据分发流程 36
3.5.2 注册服务 38
3.5.3 注册匹配服务 41
3.5.4 发布服务 41
3.5.5 订阅服务 43
3.5.6 其它服务和组件 44
3.5.7 小结 45
第四章 传感器数据分发系统实现 47
4.1 系统的软硬件环境 47
4.2 中间件层实现 48
4.2.1 中间件层次结构 48
4.2.2 中间件的移植 51
4.3 服务实现 52
4.3.1 传输服务 53
4.3.2 注册服务 56
4.3.3 注册匹配服务 62
4.3.4 发布服务 66
4.3.5 订阅服务 68
4.3.6 子网请求管理分发服务 69
4.3.7 时间同步服务 72
4.4 注册服务器实现 72
4.5 传感器代理实现 74
4.6 作战子网网关实现 79
第五章 数据分发系统测试 84

5.1 测试环境 84
5.1.1 软硬件配置 84
5.1.2 测试工具 85
5.2 测试内容和结果 86
5.2.1 功能测量 86
5.2.2 性能测量 88
第六章 结束语 92
6.1 论文工作总结 92
6.2 下一步工作 92
致 谢 93
参考文献 94

第一章 绪 论

1.1 课题的背景

传感器网络是由大量具有感知、计算和通信能力的低成本低功耗的微型传感器组成，用于实时感知和采集网络覆盖区域内的感知对象的信息，该网络常常由成百上千个传感器协同工作，传感器网络中的节点一般可分成两类：一是传感器节点，也称为源节点，这些节点可对观察范围内的感知对象产生原始数据，并可存储、处理和传输数据，二是网关节点，也称为节点（或基站），这些节点用于实现传感器网络与 Internet 的连接，往往数量有限，但能量能够得到补充。研究表明，传感器网络中数据通信量大、结构复杂，如何建立有效的数据传输路径，节约节点的传输能耗，延长网络生存期，设计有效的数据分发方案是至关重要的。

目前数据分发方案有很多，其分类根据源节点与节点是否可移动，数据分发可分为三类：

(1)固定源节点与固定节点间的数据分发。源节点与节点一旦配置好，它们的位置就不再改变，直到网络终止，这是使用最广泛的数据分发。

(2)固定源节点与移动节点间的数据分发。用户使用 PDA 移动设备在感知区域内移动，通过查询源节点来获取有关对象的当前状态或近来目标活动的概况。

(3)移动源节点与移动节点间的数据分发。可用于监测和跟踪移动目标的传感器网络。

有效可靠的数据分发关键在于建立数据分发的路径，通常要考虑节点的剩余能量、所处的地理位置、网络拓扑、当前环境和节点状态等因素。数据分发方案主要用于优化数据分发路径，使其尽量接近最佳路径，以减少能量消耗和网络拥塞，有效延长网络寿命。针对数据分发的三种分类，需要使用不同的数据分发方案，才能确保有效的数据传输。

(1)源与节点均固定的数据分发方案

源与节点均固定的传感器网络应用最为广泛，常用于环境监测、抢险救灾、防恐反恐以及交通流量监测等，吸引了许多研究者的关注，提出了几种比较典型的数据分发方案：基于外部存储的数据分发方案 ES(External Storage-based data dissemination scheme)、基于数据中心存储的数据分发方案 DCS(Data-Centric

Storage-based data dissemination scheme) 、基于本地存储的数据分发方案 LS(Local Storage-based data dissemination scheme)、基于索引的数据分发方案(index-based data dissemination scheme)等。不同的方案具有不同的构架形式，并适用于不同应用环境。

(2)源固定而节点不固定的数据分发方案

源固定而节点可以移动的传感器网络应用广泛。每个移动的节点，只要定期报告它们的当前位置，就可从源节点处接收到感知数据。但频繁的位置更新需要消耗大量能量，而带宽和电池能量是严格受限的。针对这种传感器网络的特点，研究者提出了一些数据分发方案，其中比较典型的有以下几种：TTDD(Two-Tier Data Dissemination)数据分发方案、EDDA(Efficient Data Dissemination and Aggregation)数据分发方案、ODDD(On-Demand Data Dissemination)数据分发方案、HDDS(Hierarchical Data Dissemination schemes)数据分发方案等。该类方案大多将监测区域分成均匀的网格单元，利用靠近网格边界的传感器转发数据，但同时存在网络带宽和能量使用效率较低等缺陷，应用环境也相对有限。

(3)源与节点均不固定的数据分发方案

当源节点和节点可以移动时，前面介绍的数据分发方案都不是很有效。基于动态代理树的数据分发方案(Dynamic Proxy Tree-based data dissemination scheme)，其基本思想是每个源节点或节点对应于一个固定的节点，称为源代理或代理，监测同一目标的源代理和查询该源的代理形成一棵代理树，通过代理树，源节点可以把感知数据定期发送给它的代理，继而再转发给代理树中的多个代理，每个节点可以查询它的代理来获取数据。这种数据分发方案的关键是当源或代理改变时，如何有效重建代理树，根据代理树的分布式不同有两种方案：一种是最短路径方案 SP(Shortest Path-based scheme)；另一种是跨度范围方案 SR(Spanning Range-based scheme)。由于 SR 需要的控制消息比 SP 少，因此，SR 优于 SP。

美国海军经过 20 多年的论证，针对防御敌方远程巡航导弹对其水面舰船的攻击而提出的课题，在美军的国防军事领域提出了多平台军事信息网络。其主要思想是综合利用位于不同位置、具有不同(或相同)探测能力的传感器数据，使所有作战单元能够实时、精确的共享多传感器的数据，在本地进行分布式融合处理，形成统一的态势图，然后直接提供给本地作战单元的武器系统和指控系统使用[1]。

为了达成上述作战应用需求，就需要改变传统的单平台、独立的战场态势感知手段，充分利用多传感器的信息，能将多源、多维、多平台的传感器信息实时地分发给作战系统。作战系统根据应用需求对来自多个传感器的数据采取不同的

处理方式。对远程预警应用，作战系统期望借助其它传感器的探测能力来扩展自身的探测范围。对多平台协同作战应用，作战系统期望同时获取多个传感器的数据，进行融合处理以获得统一的战场态势图。综合使用多个传感器的数据，能够扩大作战系统的作战范围、提升作战效能。在文献[2]中，给出了单传感器条件下和多传感器协同条件下武器效能的对比图，分别如图 1-1 和图 1-2 所示。

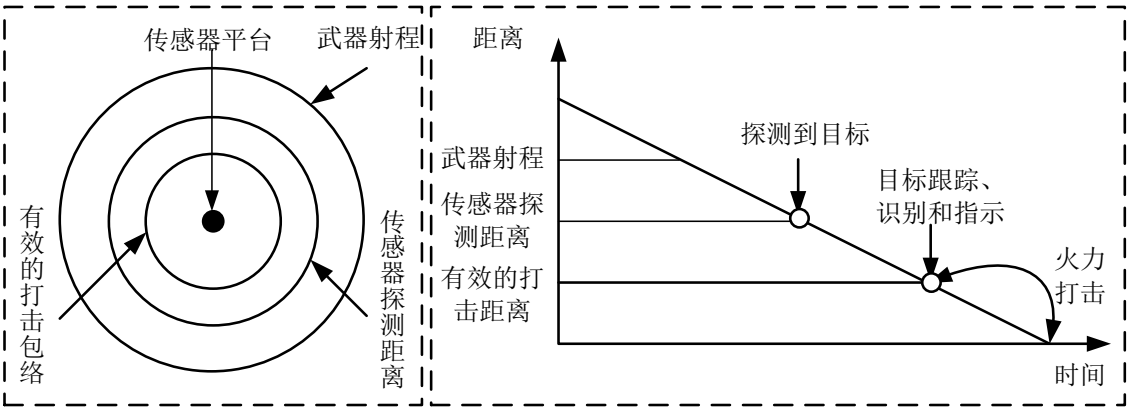

图 1-1 单传感器武器效能说明图

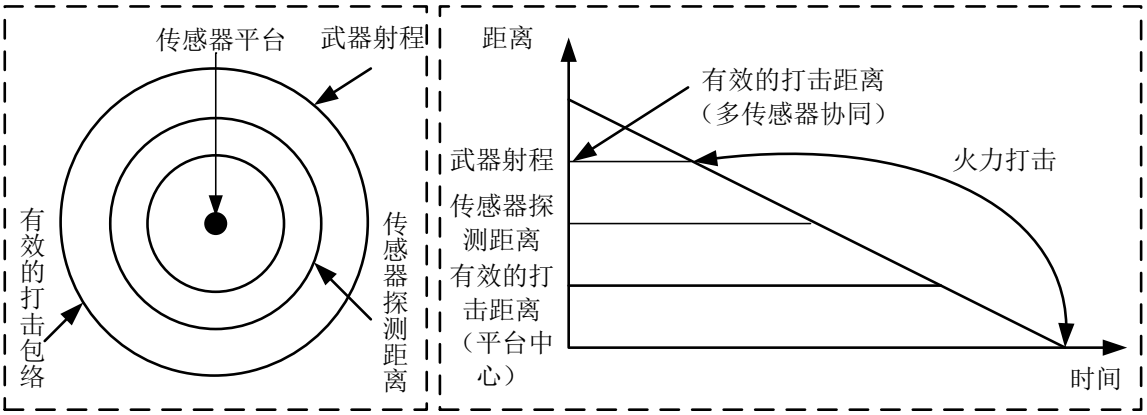

图 1-2 多传感器协同武器效能说明图

美国军方提出的该传感器数据传输网络发挥着越来越重要的作用，但是也存在着一些不足：

一、网络中的传感器单一，探测距离有限，不能满足远程预警、协同作战应用需求。美军 E-2C 预警机的雷达探测距离大概在 400 公里，且雷达源单一，因此有必要利用网外的传感器扩展其探测范围、增强探测效能。

二、现有战术数据链和战术互联网系统对传感器数据的使用不够灵活。战术数据链和战术互联网中的作战平台通常只能使用加入网络中的传感器。现有的军

事信息网络能够作为信息传输平台和网络互连环境，将传感器节点和战术互联网/战术数据链互连在一起，实现传感器数据分发。图 1-3 是网外传感器和战术数据链/战术互联网的互连关系图。

目前我军的军事信息网络的建设已经初具规模，为军队信息化转型奠定了基础，为新军事变革带来了新的发展机遇。与此同时，军事信息网络也面临着新的挑战：网络通信对通信带宽有着巨大的需求，而且对带宽的需求是呈指数增长的[3]。

在 2000 年科索沃战争的崇高铁砧行动（Operation Noble Anvil）中，美军对通信带宽的需求是 1991 年伊拉克海湾战争沙漠风暴行动（Operation Desert Storm）时的 2-2.5 倍，2002 年阿富汗的持久自由行动（Operation Enduring Freedom）的带宽需求是海湾战争的 7 倍， 2003 年伊拉克自由行动（Operation Iraqi Freedom）

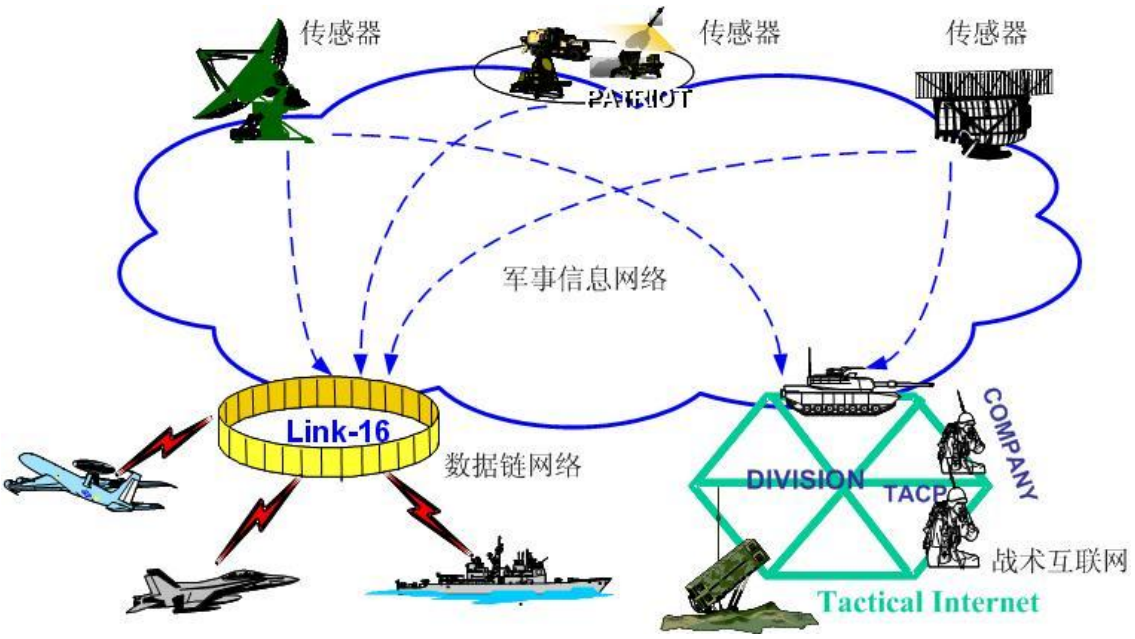

图 1-3 网外传感器与战术数据链/战术互联网的互连关系图

的带宽需求则是海湾战争的 10 倍[4]。

带宽的需求还将继续增长，而且还将远远超出网络可提供的带宽。图 1-4[4] 是美国兰德公司 Arroyo 研究中心在 2003 年对美陆军一个旅级作战单位当时的带宽能力和未来的带宽需求所做的评估，从图中可以看出，带宽需求远远超过了当时或近期的网络带宽能力，两者大致相差数百 Mbps。

要解决带宽紧张的现状，不能仅仅通过提高通信带宽的方式解决。加拿大陆军中校 P.C. Cooper 在其为文献[4]撰写的推介文章“Future Army Bandwidth Needs and Capabilities”[5]中引用了一位通信军官的观点：“Bandwidth is like beer you can

never get enough of it”，用户对通信带宽的需求是无止境的。新的技术可以暂时提高通信带宽，但永远也不能满足未知的需求。

文献[4]提出了通过信息带宽需求再评估、增加带宽、应用结构革新、带宽管理等思想来解决带宽需求矛盾。信息带宽需求再评估是一个漫长的过程，将会受到多种因素的制约，对缓解当前带宽资源紧张的现状不适合。增加带宽对缓解带宽资源不是长期有效的解决方案，一方面如前面所说，带宽提升的速度总是低于带宽需求增长的速度，另一方面是因为部队通信装备的更新换代是一个缓慢的逐步提升的过程，设备的通信带宽在短期内难有大幅的提高。因此在目前情况下，要有效的缓解带宽资源矛盾，需要通过应用结构革新优化应用结构，降低应用对带宽需求；通过带宽管理合理分配带宽资源。

在图 1-4 中，还可以看到传感器数据对带宽的需求占到总需求的绝大部分，而态势感知、语音、指控数据以及其它的应用只是总需求的一小部分。因此有效地降低传感器信息对带宽的需求对缓解带宽矛盾具有十分重要的意义。

Figure 1.7
Initial Estimates Suggest Future Requirement (Demand) Will Exceed Existing Supply

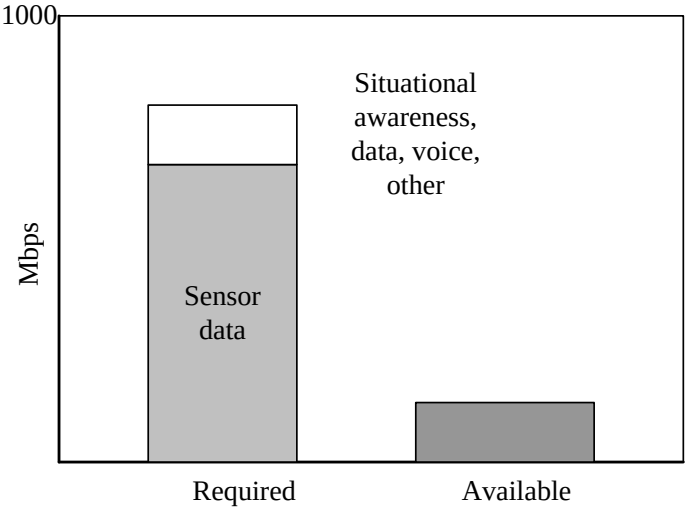

图 1-4 RAND 报告可用/带宽对比图

1.2 课题的目标

本课题旨在寻求一种能够节约使用带宽资源，将多个传感器的数据以按需分发的方式，实时地分发给战术数据链和战术互联网中的用户终端，使战术互联网和数据链中的用户终端能够获得更精确、更广阔、统一的战场态势图，达成远程

预警和多平台协同作战能力，提升数据链和战术互联网作战效能。本文基于“面向业务，按需分发”的思想来达成上述目标，将所设计和实现的系统命名为传感器数据分发系统。

传感器数据分发系统设计开发应该遵循以下原则：

一、数据分发以实时按需分发作为最高的指导原则，数据分发过程必须是实时的，在满足实时要求的前提下再保证可靠性。

二、实现传感器数据按需分发，保证合适的数据在合适的时间内传送给合适的数据使用者。

三、具备数据源自动发现，不需要数据使用者自己在网络中查找数据源的具体位置。

四、传感器和数据使用者可以动态加入和退出系统，具备动态扩展能力。

五、设计实现方案不能对已部署的应用系统做大的改动。

1.3 战场环境虚拟仿真研究的现状与发展趋势

本课题的主要工作是设计一个传感器数据分发系统，它具备传感器数据源自动发现能力，能够实现传感器数据按需分发，保证传感器数据分发过程实时，支持传感器和数据用户动态地加入和退出，能够满足多种应用需求。

针对课题目标，结合当前的实际情况，课题主要有以下五个方面的工作需要完成：

一、针对传感器数据分发应用，规划符合实际情况的系统应用场景。

二、寻找能够满足实时按需分发需求的软件框架。

三、结合系统设计原则和应用需求，设计和实现传感器数据分发系统。

四、测试传感器数据分发系统的主要功能和性能。

五、针对课题中发现的新问题，提出下一步的研究方向。

1.4 本文研究的背景、内容和意义

第一章介绍课题的背景、课题的目标、课题目前的研究现状、课题的主要工作以及论文的组织情况。

第二章结合实际情况规划了传感器数据分发系统的应用场景，介绍了发布/订阅机制的基本原理，研究了主要相关技术，最后研究了实时数据分发规范 DDS。

第三章首先明确了传感器数据分发系统的设计思路；结合传感器数据分发系统的应用场景和系统需求，设计了符合实际情况的软件结构和发布/订阅框架；根据“按需分发”要求规划了传感器代理、子网网关、注册服务器的功能；依照 SOA 方法，通过分析数据分发业务流程，将整个系统分解为独立的服务和功能组件，使用这些服务和组件，实现传感数据分发系统的数据分发功能。

第四章设置了传感器数据分发系统的软硬件环境，介绍了 ACE、TAO 和 OpenDDS 中间件在 X86/Windows XP 平台上的移植工作，构建了中间件层；详细阐述了发布服务、订阅服务等主要服务的具体实现，最后介绍了注册服务器、传感器代理和作战子网网关的实现工作。

第五章介绍了传感器数据分发系统的功能和性能测试工作。

第六章总结了本课题的主要工作和创新点，指出了下一步研究和改进的方向。

第二章　系统应用和发布/订阅研究

将传感器数据分发给战术互联网和战术数据链，需要将现有的军事信息网作为信息承载网络。发布/订阅是面向服务框架 SOA (Service Oriented Architecture)的一种实现技术，具备服务自动发现、动态扩展和事件过滤等特性。DDS 是 OMG 组织针对数据实时分发应用而制定的数据分发规范，它以数据为中心，使用发布/订阅机制实现数据实时分发，具有丰富的 Qos 策略，适合用于设计实现传感器数据分发系统。本章规划了传感器数据分发系统的应用场景，研究了 SOA 实现技术－发布/订阅，以及实时数据分发规范 DDS。

2.1 数据分发系统应用场景

传感器生成的数据通过现有的军事信息网分发给战术互联网和战术数据链网络是一种符合实际、成本低廉、组网快捷的方案。遵循系统设计开发不能对已部署的系统做出大改动的原则，规划了图 2-1 所示的系统应用场景。整个数据分发系统划分为三大部分：传感器（数据源）、作战子网（数据用户）和 IP 骨干网。本文使用 IP 骨干网（IP Backbone Network）指代军事信息网，它是一个军事化的 Internet，使用 TCP/IP 协议族作为网络互连协议，是传感器数据的信息承载网；作战子网指代战术互联网和战术数据链，子网用户终端通过子网网关接入 IP 骨干网。它们之间的关系可以描述为：传感器生成的数据源通过 IP 骨干网到达作战子网网关，作战子网网关将传感器数据分发给子网内的用户终端。

2.1.1 传感器

在本文中，将传感器限定于空中目标探测雷达，有火控雷达、红外雷达、远程预警雷达等多种制式。传感器数据指空中目标探测雷达生成的航迹数据。传感器在地域上是分散的，假定其不受限于地域的限制，能够任意地接入 IP 骨干网，向网络中注入传感器数据。

对远程预警应用，用户终端可以通过获取远程预警雷达或者那些能够探测到兴趣目标的雷达生成的数据来满足应用需求。对协同作战需求，用户终端请求某一或者某些传感器向其提供数据，然后在本地生成局部统一的战场态势图，统一

的战场态势图是实现协同作战的前提。美军的 CEC[10][11]和 TCN[12]系统就是通过多传感器数据融合以获取全局统一的战场态势图。

为了不改变现有传感器节点结构，在传感器前端设置传感器代理，它部署于 IP 骨干网的边缘，代理传感器接入 IP 骨干网。传感器数据分发由传感器代理上实现，它具有以下三方面功能：

一、进行协议转换，将传感器信息封装成 IP 分组。传感器代理采用双连接模型，提供内向接口和网络侧接口。内向接口有并口、串行接口（RS232C、RS422）、以太网接口等多种方式[13]，接收来传感器的数据；网络侧接口是以太网络接口，用于接入 IP 骨干网。传感器代理需要实现内向接口协议向网络侧接口协议的转换，将传感器数据封装进 IP 分组，以便在 IP 骨干网中传输。传感器数据有三种封装方法，具体参看文献[14]。

二、对传感器数据进行标准化处理。由于传感器制式和传感器厂商的不同，存在多种传感器数据格式，如 Asterix、RVP、ARUP。对用户终端来说，识别并处理多种传感器数据格式是一件困难而艰巨的任务，需要设计多个传感器数据识别和处理模块，当有新的传感器数据格式产生时，就需要添加新的识别处理模块。本课题由传感器代理对数据进行标准化处理，将其转换成一种标准的数据格式，这样用户终端只需将标准的数据格式转换为本地特定的数据格式即可，降低用户终端的复杂度。

三、实现传感器数据分发功能。将封装了标准传感器数据的 IP 分组通过 IP 骨干网分发给战术互联网和战术数据链的网内节点。数据分发必须是实时的，同时还要满足按需分发的需求。

本课题采用实验室自行开发的雷达模拟器来模拟传感器，其输出的数据格式如表 2-1。简单起见，此数据格式也作为标准的传感器数据格式。

表 2-1 雷达模拟器/标准数据格式

序号	数据项	
1	传感器类型（Sensor_type）	
2	传感器编号（Radar_ID）	
3	航迹号（Track_No）	
4	目标属性标识 Target_type	
5	时间（Timestamp）	
6	目标位置	纬度 Latitude

	（Target Position）	经度 Longitude
		高度 Height
7	目标速度 （Velocity）	Vx
		Vy
		Vz

2.1.2 作战子网

本文使用作战子网代指战术数据链和战术互联网，子网内的用户终端是传感器数据的最终使用者。作战子网在网络边界处置一个或者多个作战子网网关，汇集子网内用户终端的数据请求，然后向 IP 骨干网请求数据，最后将接收到的数据分发给子网内的用户终端。作战子网内的用户终端通过作战子网网关接入 IP 骨干网。在一个作战区域内，可能同时部署了多个作战子网，它们通过作战子网网关接入 IP 骨干网中。作战子网既可以是数据的请求者，也可以是数据的发布者，从长远来说，作战子网之间也应具有互操作能力，本课题暂不考虑作战子网间的互操作，先实现传感器数据向作战子网的分发。

本课题的战术数据链以美军的 Link16 作为研究对象，战术互联网以美军的 21 世纪部队旅和旅以下作战指挥系统 FBCB2（Force XXI Battle Command-Brigade and Battle）作为研究对象。战术数据链 Link16 通过 J/TDMA 实现组网，战术互联网通过 VMF/IP 实现组网[15]。

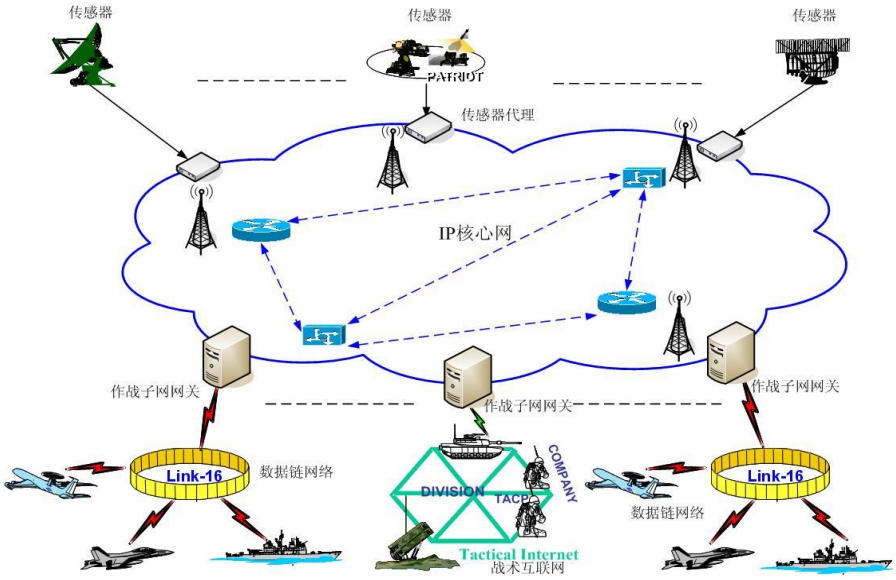

图 2-1 传感器数据分发系统应用场景

一、战术互联网

战术互联网是指在战术条件下、以互联网技术为支撑，以实时准确传输指挥信息及战场态势信息为基本目标，由战术无线电台、野战地域网及其它通信设备、计算机及相关网络设备综合而成的一个战术级数据通信网[16]。战术互联网使用成熟且广泛应用的 TCP/IP 协议族作为节点互连协议，以 IP 分组作为基本传输单元，构成一个具有统一地址的信息网络。

为了满足作战应用需要，适应战场环境，战术互联网采用 VMF（Variable Message Format）消息来表示各类信息，关于 VMF 的具体情况可参看文献[15]。为了适应战术互联网网络带宽受限及无线链路的特点，VMF 采用了 UDP 传输协议标准，由应用层（47001 协议）的分段/重组协议保证报文数据的可靠传输。

战术数据终端是战术互联网的主要用户设备，具有为指挥员和战斗员提供战场态势等诸多功能。课题目标之一是将传感器的数据以按需的方式分发给在战术数据终端并在终端界面上显示出来。

二、战术数据链

战术数据链是链接数字化战场上指挥中心、各作战部队、武器平台的一种信息处理、交换和分发系统[15]。

Link16 数据链是美国国防部用于指挥、控制和情报的主要战术数据链，支持侦察数据、电子战数据、任务执行数据、武器分配和控制数据交换。它是一种高速视距 UHF 数据链，支持战斗群各分队之间的综合通信、导航和敌我识别，用于交换联合战术数据，使用具有抗干扰能力的 UHF 无线电设备。许多平台已经装备或即将装备 16 号数据链(如机载侦察和情报系统、指挥控制系统、战斗机和轰炸机、地对空导弹系统等)，Link16 数据链已被美国和北约作为战区导弹防御的主要战术数据链[17]。

Link16 数据链使用联合战术信息分发系统 JTIDS（Joint Tactical Information Distribution System）数据终端，支持 Link11 数据链和 Link4A 号数据链的功能，并增加了如话音、相对导航和扩展的电子战性能等功能。

Link16 链采用 J 序列报文传递信息,它是美国国防部指定用于指挥控制系统的标准报文格式，是美军联合互通的基础。关于 J 序列报文的介绍参看文献[15]。

2.1.3 IP 骨干网

在本文中，用 IP 骨干网指代是军事信息网，它为传感器数据分发系统提供一

个通信业务承载平台，将传感器、作战子网通过 TCP/IP 技术互连成一个网络系统。

本文中的 IP 骨干网工作于战场环境，通信信道类型多样，可能是窄带宽的无线信道、卫星信道通信，也可能是高带宽的有线光纤信道等，网络带宽从 Kbps 到 Mbps 不等。IP 骨干网中的局部可能是低速率、高时延和频繁断链的链路的 Disadvantaged 网络[6]，不具备通常所说的高带宽特征。但是无论具体情况如何，就如 1.1.3 节所述，带宽资源总是紧张的，需要本着节约的原则使用网络带宽资源。

2.1.4 系统需求

在上述应用场景下，本文对传感器数据分发系统提出下列需求：

一、实时性需求。在传感器数据分发系统中，传感器数据是一种时效性强的应用数据（Time-critical），过期的数据毫无应用价值还可能会产生不可预知的后果，因此传感器数据分发的实时性是系统设计时需要首要考虑的因素。

二、可靠性要求。可靠性是指数据能够有序、无差错、无丢失地到达子网用户终端。实时性和可靠性通常是相互矛盾的，满足一定的实时性是以牺牲部分可靠性作为代价，反之亦然。因此在系统设计中，需要根据业务的实际需求权衡考虑。对传感器数据，满足其实时性要求是第一位的，在满足了实时性的前提下，可以采用多种方法提高数据的可靠性。

三、多种数据应用需求。

1、单源选择分发：多个传感器探测覆盖的作战区域重复，用户终端只选择其中之一作为该节点的数据源。这样的分发模式可用于远程预警应用。

2、多源选择分发：与 1 相同的场景，作战平台同时需要获取探测相同区域或目标的多个传感器数据，然后进行传感器数据融合处理，以获取更高精度的数据，如 CEC 应用[11][18]，将多源的传感器数据融合生成更高质量的航迹数据。

3、多向分发：传感器探测到的数据信息同时提供给多个作战平台。

四、动态扩展。新入网的作战子网能够从网络中获得所需的传感器数据，新入网的传感器节点能够将数据以最短的时间发送给有需求的作战子网。

五、系统的健壮性（Robustness）。系统运行过程中会受到各种不确定因素的影响，当出现异常情况时，系统能够保持正常运行或快速从故障中恢复。

六、软件跨平台通用性。传感器数据分发系统软件可以运行在多种软硬件平台上，具备平台通用性、可移植性。

2.2 基于 SOA 的发布/订阅

面向服务架构 SOA（Service Oriented Architecture）是一种信息资源共享框架结构，框架中的资源或组件被抽象为独立的服务，通过逐步优化升级单个服务的方式实现演进和升级。发布/订阅机制是一种 SOA 实现技术，它具备满足按需分发要求的潜力，适用于传感器数据分发系统的设计和实现。

2.2.1 面向服务框架 SOA

面向服务架构 SOA（Service Oriented Architecture）既是一种软件结构框架，也是一种分布式软件设计和开发方法，软件的设计和开发围绕着服务进行[19]。在 SOA 中，服务是最核心的概念，根据应用需求和业务流程发现服务、定义服务和实现服务。

一、服务发现。服务发现是进行面向服务分析和设计的第一步，服务发现的主要任务是针对应用需求和现有系统寻找可能成为服务的候选者列表。服务发现的方法有自上而下法、自下而上法和中间对齐法。

二、服务定义。服务定义的主要任务是规范性地描述服务各个方面的属性，包括服务输入/输出消息等功能性属性，服务安全约束和响应时间等服务质量约束，以及服务在业务层面的诸多属性。

三、服务实现。

一个 SOA 应用系统的组成如图 2-2 所示，包含三种角色：服务提供者、服务消费者和服务注册中心，在此之外还有抽象的服务和服务总线。

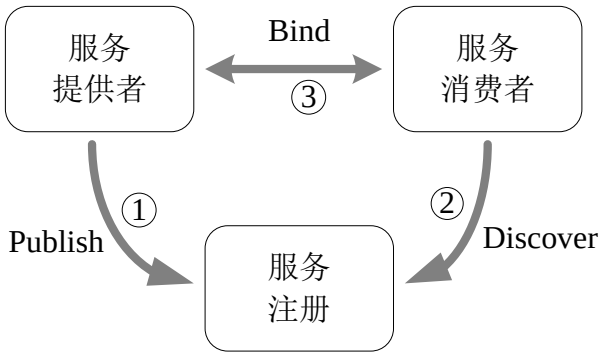

图 2-2 SOA 系统组成

（一）服务提供者（Service Provider）：通过 Publish 操作向服务注册中心注册所提供服务的功能信息和接口信息。

（二）服务消费者（Service Consumer）：通过 find 操作向服务注册中心查询所请求的服务，然后通过 bind 操作使用该服务。

（三）服务注册中心（Service Registry）：也称之为服务代理（Service Broker），用于接收、存储服务的描述信息并提供服务查询，使得服务使用者能够以技术透明和位置透明的方式来使用服务提供者提供的服务。

四、服务：网络中可被利用的应用功能或数据资源。

五、服务总线是服务使用者和服务提供者实现交互的软件总线，比如 Web Service、CORBA 等。

传感器数据分发系统的组成结构与 SOA 组成结构相符，传感器数据是服务对象，传感器代理是数据服务的提供者，作战子网网关是数据服务的消费者。传感器数据分发系统需要为作战子网网关数据服务消费者（作战子网网关）找到合适的数据服务提供者（传感器代理）。

2.2.2 发布/订阅基本原理

发布/订阅是一种使分布式系统中各个参与者，能以发布/订阅语义[20]进行交互的分布式框架。发布/订阅框架两个最基本的组成元素是发布者 Publisher 和订阅者 Subscriber，二者之间交互的对象称之为事件 Event。事件包括信息数据 sample 和元数据 Metadata。发布者是数据的生成者，通过发布操作将数据分发给感兴趣的订阅者。订阅者是数据的消费者，接收自己感兴趣的数据。发布/订阅的原理图如图 2-3[21]所示。

发布者通过 Publish()操作向发布/订阅服务发布事件；订阅者通过 subscribe()向发布/订阅服务提出订阅条件，即对感兴趣的事件的描述，订阅者可以通过 unsubscribe()终止前面的订阅；发布/订阅服务调用 notify()将订阅者感兴趣的事件分发给订阅者。图 2-3 是基本的发布/订阅模型，对于扩展的发布/订阅模型，发布者具有 advertise()和 unadvertise()方法。advertise()方法是发布者在调用 publish()方法之前将其拟发布的事件的描述信息（发布条件）提供给发布/订阅服务，unadvertise()方法则用于终止发布条件。

发布/订阅服务为发布者和订阅者提供匹配服务，为订阅者找到合适的发布者或者为发布者找到合适的订阅者。发布/订阅服务还负责发布者和订阅者之间的交互，有推模式（Push Mode）和拉模式（Pull Mode）两种模式可供选择。

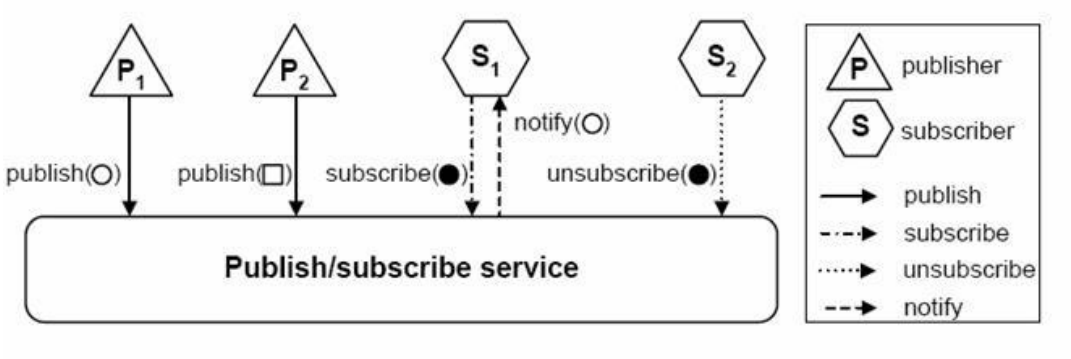

图 2-3 基本的发布/订阅原理示意图

2.2.3 发布/订阅的特点

一、服务自动发现特性

在发布/订阅框架中，发布者可以映射为 SOA 中的服务提供者，而订阅则可以映射为服务消费者。服务自动发现特性指的是在发布/订阅框架中，订阅者可以通过发布/订阅服务自动地发现符合其订阅条件的发布者，即服务的提供者。发布者可以通过发布/订阅服务自动地发现对其事件感兴趣的订阅者。服务的自动发现特性使得系统具有动态扩展性（Scalability）。具体到传感器数据分发系统，服务自动发现特性使得作战子网网关能够在 IP 骨干网中找到合适的传感器数据源。特别是网络中有多个数据源时，服务自动发现特性使得发布/订阅机制的优势更加明显。

二、动态扩展特性

动态扩展是发布/订阅框架一个很重要的特性，这是由发布/订阅的服务自动发现特性所带来的。系统运行过程中加入的节点可以立刻参与到系统中，节点退出不会对系统正常运行状态造成任何影响。当有新的发布者加入时，发布者向发布/订阅服务进行注册，发布/订阅服务将其与现有的订阅者进行匹配，如果有对其事件感兴趣的订阅者，发布者就会将其发布的事件分发给订阅者。当有发布者退出系统时，发布者会停止发布事件并调用 unadvertise()方法终止发布条件。相似地，当有新的订阅者进入系统时，订阅者向发布/订阅服务进行注册，根据匹配的结果，符合订阅条件的发布者会将事件分发给订阅者，订阅者可以调用 unsubscribe()终止订阅条件。无论是新的发布者或者新的订阅者加入和退出，都不会对系统中其它的发布/订阅过程产生影响。

动态扩展特性对传感器数据分发系统来说，是非常重要的，因为系统中随时都会有新的传感器代理（发布者）和作战子网网关（订阅者）的加入和退出，发

布/订阅框架的动态扩展特性正符合传感器数据分发系统动态扩展的应用需求。

三、事件过滤特性

发布/订阅框架还具有事件过滤特性，订阅者通过订阅条件声明自己感兴趣的事件并只接收自己感兴趣的事件，发布者将自己发布的事件分发给感兴趣的订阅者集合。事件过滤特性可以降低网络中的冗余数据，符合第一章所提出的按需分发思想。在后续章节中，将在发布/订阅框架事件过滤的基础上，结合传感器数据分发应用，从应用层面进一步减少冗余数据的传输，将按需分发实现得更加细致。

在文献[21] 中，提到了发布/订阅具有时间松耦合、空间松耦合、同步松耦合三种松耦合特性，其中时间松耦合和同步松耦合是消息中心模式的发布/订阅框架所特有的。本课题为传感器数据分发系统设计的发布/订阅框架，这两种松耦合特性体现得不明显，因此对其不做深入研究。空间松耦合性是指发布者和订阅者无须知道对方的通信地址，由发布/订阅服务为发布者和订阅者建立联系，服务自动发现特性就是由空间松耦合引入的。

通过对发布/订阅框架优点的介绍，以及结合传感器数据分发系统应用的分析，可以认为，发布/订阅作为 SOA 的实现技术，适合于传感器数据分发系统实现按需分发需求。

2.2.4 发布/订阅相关技术

发布/订阅框架的实现包含四个方面：事件声明方式、匹配模式、发布/订阅的通信架构、发布/订阅 QoS。本节先介绍前三方面的内容，关于 Qos 留待后面结合 DDS 规范进行研究。

一、事件声明方式

事件声明方式是指发布条件和订阅条件对事件的描述方式，决定着后续的匹配算法。目前有三种事件声明方式：主题方式、内容方式和类型方式[21]。

1、主题方式（Topic-based）

最早的发布/订阅实现是基于主题的，主题的英文翻译是 Topic 或者 Subject，在本文中统一采用 Topic 表示“主题”。目前许多发布/订阅商用实现都是基于 Topic[22][23][24]。Topic 以关键字（keyword）刻画事件内容，区分不同的事件，关键字通常是字符串的形式。发布条件以关键字的形式描述发布者发布的事件，订阅条件以关键字的形式描述感兴趣的事件，发布/订阅服务通过比较发布条件和订阅条件中的关键字，获得匹配结果。

基于 Topic 方式的发布/订阅框架是一种静态的框架，即 Topic 需要提前规划，一经规划好之后，所有的发布/订阅都只能围绕这些规划好的 Topic 进行。Topic 的另外一个不足之处是 Topic 以关键字的方式描述事件，其对事件的描述能力十分有限。

尽管 Topic 方式存在上述两点不足之处，但是它实现简单，目前许多成熟的发布/订阅中间件采用的均是 Topic 方式，包括后面将要研究的 DDS 规范（Data-Distribution Service for Real-Time Systems）采用的就是 Topic 方式。

2、内容(Content-based)方式

针对 Topic 方式表达能力弱的不足，出现了内容方式 Content-based。内容方式对事件的描述不是按照事先已经定义的标准，能够根据需求进行动态的扩展。事件的属性可以是封装事件的数据结构或者是与事件相关的元数据（meta-data）[25]。发布条件和订阅条件以内容过滤器的方式描述事件，内容过滤器由 name-value 对和比较符（＝<>）组成， name 代表属性名，比较符和 value 值构成限定条件。发布条件和订阅条件通过增加过滤器条目不断地扩展，增强对事件的描述能力。

基于内容的方式具有很强的事件描述能力，能够根据需求不断地扩展，但是基于内容方式的发布/订阅框架，其匹配算法实现困难，匹配效率低，不适用于大规模的分布式系统。正是由于内容方式的高开销，当事件主要属性值是有限的离散值时，应该优先选用静态的 Topic 方式。

3、类型(Type-based)方式

类型方式[26]是以数据结构来区分和刻画事件，这样的方式更易于编程语言和中间件的集成。但是基于类型方式的缺点是显而易见的，对不同质的事件，它的数据结构却有可能是相同的，所以单独使用类型方式不可行，它通常与主题方式结合使用。本文后面研究的实时数据分发规范 DDS 将类型方式与 Topic 方式相结合，类型使用 CORBA 的接口定义语言 IDL（Interface Define Language）定义。

在下一章中，将结合传感器数据分发系统的特点和应用需求选择一种符合需要的事件声明方式。

二、匹配模式

研究了发布条件和订阅条件的描述方式之后，下面研究发布/订阅的匹配模式，解决发布条件和订阅条件匹配方式。目前有两种基本的模式可供选择：集中注册匹配模式和分布注册匹配模式。

集中注册匹配模式通常是设置一个中心服务器，发布者和订阅者以注册的方式将发布条件和订阅条件发送给中心服务器，中心服务器接受注册，登记发布条

件和订阅条件，根据匹配算法对发布条件和订阅条件进行匹配，为发布者找到符合发布条件的订阅者，或者为订阅者找到符合订阅条件的发布者。在本文中，将中心服务器称为注册服务器。

分布注册匹配模式不需要任何中心节点，发布者直接向网络中广播其发布条件，订阅者将接收到的发布条件与本地的订阅条件进行匹配，如果匹配，匹配者就找到了合适的数据源，反之亦然。这样发布条件和订阅条件的匹配会分布在各个节点完成，无须中心节点的参与。

三、通信架构

发布/订阅框架第三个方面的内容是通信架构，发布者和订阅者如何进行通信交互。目前基本的通信架构是：中心代理架构（Centralized Architecture）和非中心直连架构（Decentralized Architecture）。

中心代理架构是在发布者和订阅者之间引入一个中介，在实践中通常是设置信息服务器，信息服务器接收发布者发布的事件，然后根据订阅者的订阅条件对事件过滤之后，将事件转发给订阅者。采用这种架构的系统通常对数据的可靠性、数据的一致性和事务（Transaction）支持有着较高的需求，而对数据吞吐量则没有很高的要求[21] 。电子商务系统通常采用中心代理架构实现。

非中心直连架构不需要第三者的参与，在发布者和订阅者间直接进行通信，数据的存储和转发分布在各个节点实现，TIBCO Rendezvous[23]采用的就是这样的架构。这种架构的主要特点在于不存在系统瓶颈和单点故障。对于要求实现数据快速高效分发的应用系统，这种分布式框架十分适合[21]。

2.3 实时数据分发规范－DDS

传感器数据是一种对实时性要求高，时效性强的数据对象，用户实时地获取传感器数据是十分关键的问题。将合适的传感器数据在合适的时间分发给合适的用户是传器数据分发系统的设计原则之一，其中合适的时间是指传感器数据的实时分发，合适的传感器数据和合适的用户是指传感器数据的按需分发。

2.3.1 DDS 背景

在 DDS 规范发布之前，没有哪种标准能够满足实现上述三个合适的要求。CORBA 的事件服务（Event Service）[27]、CORBA 的通知服务（Notification Service）

[28]以及 JMS(Java Message Service)[29]虽然也是发布/订阅语义，但是它们缺少对实时性的支持，具备很少或者根本没有 Qos 保证。

为了弥补这些缺陷，OMG 组织总结了发布/订阅以往应用的经验，针对工业过程控制和国防系统应用领域，制定了实时数据分发规范 DDS(Data-Distribution Service for Real-Time Systems)[30]。DDS 规范以数据为中心（Data-centric），面向数据，针对数据对象制定了诸多 Qos 策略。它使用发布/订阅机制分发数据，易于实现按需分发。DDS 已经在时间关键（Time-Critical）和任务关键（Mission-Critical）的作战系统中得到广泛应用，如美陆军的未来作战系统 FCS 以及海军的开放系统结构（OSA）。DDS 面向数据的特性以及发布/订阅机制适用于本课题所研究的传感器数据分发应用，因此选用 DDS 规范作为传感器数据分发系统设计和实现的依据。当前已经发布了两个版本，分别是 2004 年 12 月发布的 Ver1.0[31]和 2007 年 1 月发布的 Ver1.2[32]，后面关于 DDS 的研究均依据 DDS Ver1.0。

2.3.2 DDS 结构

DDS 规范分为两层,数据中心的发布/订阅层 DCPS（Data-centric Publish/Subscribe）和数据本地重构层 DLRL（Data Local Reconstruction Layer），DLRL 层建立在 DCPS 层上。DCPS 层是 DDS 的核心和基础,建立基于发布/订阅机制的分发框架,发布过程和订阅过程在 Topic、Publisher、Subscriber、Datawriter 和 Datareader Qos 设置的控制下，实现数据实时分发。DLRL 层对 DCPS 层提供的服务进行了抽象,在 DLRL 层建立了应用数据与底层服务的映射关系。DLRL 层是可选层，目前大多数的实现，包括 OpenDDS 仅实现了 DCPS 层的主要功能，因此后面的研究和讨论主要围绕 DCPS 进行。

DCPS 层引入了一个抽象的全局数据空间 GDS(Global Data Space)，发布者向 GDS 写入数据，订阅者从 GDS 读出数据，通过 GDS 共享信息。DCPS 层将用户对资源的需求和资源的可用信息都转化为服务质量 QoS (Quality of Service) 参数。在 DDS 中，QoS 参数贯穿整个 DDS 分发过程，在注册、匹配、数据分发阶段都离不开 Qos 的参与。DCPS 层的发布/订阅功能接口只需指明需要的 QoS，由 DCPS 依照 Qos 实现数据的发布和订阅。

2.3.3 基本概念

DCPS 层的发布/订阅功能是由 Topic、Datareader、Datawriter、Publisher 和 Subscriber 五个实体协作实现的，图 2-4 是 DCPS 的概念模型[31]，说明了五个实体之间的协作关系。

一、Topic：Topic 定义了可以合法写入 GDS 的数据类型，Topic 相关联的数据结构由 IDL 定义，这是类型方式的体现。相同的 Topic 肯定具有相同的 IDL 定义，IDL 定义不同，肯定对应不同的 Topic。

二、Datareader：实现数据订阅功能的实体，与 Topic 相关联，一个 Datareader 对应一个 Topic。

三、Datawriter：实现数据发布功能的实体，与 Topic 相关联，一个 Datawriter 对应一个 Topic。

四、发布者（Publisher）： Datawriter 容器，一个发布者可以同时包含多个 Datawriter，发布多个 Topic 的数据。

五、订阅者（Subscriber）：Datareader 容器，一个订阅者可以同时包含多个 Datareader，订阅多个 Topic 的数据。

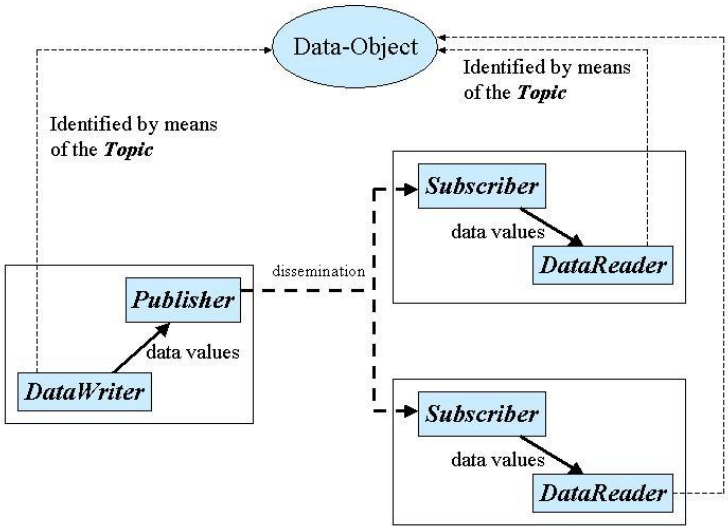

图 2-4 DCPS 层概念模型图

2.3.4 Qos 策略

DDS 规范的主要特征是它具有丰富的 Qos 策略,是其面向数据特性的主要体现。依赖于这些 Qos，DDS 能够有效地控制和管理网络带宽、内存空间等资源的

使用，控制数据的生存时间（Persistence）、数据可靠性和实时性。DDS Ver1.0 总共制定了 21 种 Qos 策略，DCPS 中所有的实体对象（Topic、Datareader、Datawriter、Publisher、Subscriber 和 DomainParticipant）均有与其相应的 Qos。Publisher/Subscriber Qos 应用于 Publisher /Subscriber 实体内所有 Datawriter 实体，有 4 种；Datawriter/Datareader Qos 用于数据分发控制，Datawriter 有 12 种，Datareader 有 11 种；Topic Qos 是对发布数据的属性描述，主要用于注册匹配，有 12 种。现将 Publisher/ Subscriber、Topic、Datawriter、Datareader 的可以使用的 Qos 总结于表 2-2 中。通过灵活使用这些 QoS 策略， DDS 不仅能在窄带的无线环境上，也能在宽带的有线通信环境上开发出满足实时性需求的数据分发系统。

表 2-2 DDS 规范诸实体可用 Qos 总结

序号	Publisher/ Subscriber Qos	Topic Qos	Datawriter Qos	Datareader Qos
1	presentation	topic_data	durability	durability
2	Partition	durability	deadline	deadline
3	group_data	deadline	latency_budget	latency_budget
4	entity_factory	latency_budget	liveliness	liveliness
5		liveliness	reliability	reliability
6		reliability	history	destination_order
7		destination_order	resource_limits	history
8		history	transport_priority	resource_limits
9		resource_limits	lifespan	user_data
10		transport_priority	user_data	time_based_filter
11		lifespan	ownership_strength	reader_data_lifecycle
12		ownership	writer_data_lifecycle	

本课题对 DDS 规范的 Qos 进行了深入的研究，按照功能进行了分类，现对其中部分 Qos 的功能和应用场合总结如下，具体内容可参看 DDS 规范[31]。

一、资源管理

1、Resource_limits 策略控制缓存资源的使用。由 max_samples，max_instances 和 max_samples_per_instance 三个子项构成，分别代表 Datawriter 和 Datareader 可管理的 samples 数量、instance 的数量、每个 instance 可管理的 sample 的数量。Resource_limits Qos 与 History、Reliability 等 Qos 配合使用。

2、Time_based_filter 指定数据发布的最小时间间隔，发布者使用 Time_based_filter 对数据进行采样，然后再发布。订阅者使用 Time_based_ filter 过滤接收的数据，小于最小时间间隔的数据将不会被接收。此策略用于限制发布者发布数据的数量。

二、实时性控制

1、Deadline 定义数据最大的到达间隔时间。发布者用 Deadline 指定数据发布周期，订阅者使用 Deadline 指定所期望的数据更新周期。此 Qos 策略应用于实现 Datareader 和 Datawriter 的匹配，当 Datawriter_Deadline<=Datareader_Deadline 时，二者的 Deadline Qos 匹配，反之则不匹配。传感器数据分发系统将发布条件和订阅条件中的数据刷新率映射为 Deadline Qos。

2、Latency_budget 制定了数据时延要求，这需要时间同步服务的支持。

三、数据有效性控制

History：Datawriter 用 History 控制发送缓存的分配，Datareader 用 History 控制接收缓存的分配。有 KEEP_LAST 和 KEEP_ALL 两种策略可供选择，OpenDDS1.0 两种策略均支持。

四、通信控制

1、Reliability 用于控制数据分发的可靠性等级。此策略依赖于 History 和 Resource_limits 的设置，因为实现可靠的分发需要充分的内存资源作为保障。DDS 规范的 Reliability Qos 等级有 RELIABLE 和 BEST_EFFORT。RELIABLE 能够实现数据的可靠重传，BEST_EFFORT 则只是尽力而为的。由于不存在重传确认操作，BEST_EFFORT 的数据分发实时性肯定是优于 RELIABLE 的。RELIABLE 是通过 TCP 传输协议实现，BEST_EFFORT 则是由 UDP 传输层协议实现。软件设计了一个可插入的传输层，根据 Reliability Qos 选择合适的传输层。

2、Destination_order 允许订阅者选择处理接收主题更新值的方式，通常有两种方式：数据源或者数据目的地的时间戳。

3、Ownership 提供了主题的写入权限。有 SHARED 和 EXCLUSIVE 两种类型可供选择。SHARED 允许多个 Datawriter 同时更新 Topic 的数据空间，这需要与

Destination_order 策略配合使用。Destination_order 确定多源数据在订阅端的组合方式。EXCLUSIVE 则只允许 Strength 值最高的 Datawriter 对 Topic 数据空间进行更新，这需要 Ownership_ strength 策略配合。Ownership_strength 设定 Datawriter 写入数据的优先级。

4、Transport_priority：使应用程序能够在传输层用不同的优先级发送消息。在传输层配置时，应用程序指定 Datawriter 上的 Transport_priority 与传输层协议之间的映射关系。

五、数据附加属性

Topic_data：应用程序通过此 Qos 将一些额外的信息附加给 Topic，远程应用程序会以已定义的方式使用这些额外信息。发布和订阅条件中的区域信息、精度等级、目标类型会附加在 Topic_data Qos 中，传感器类型会转换成 Topic，数据刷新率会设置成 Deadline Qos，在注册服务器对 Topic、Deadline Topic_data 等 Qos 进行匹配，实现有效的按需分发。Topic_data Qos 可以根据应用需求进行不断的扩展，这是本课题针对 OpenDDS 实现不足之处的改进点之一。

2.4 当前研究现状

信息共享是网络中心作战能力的具体体现，本课题的传感器数据分发就是希望实现数据链和战术互联网能够共享使用网外的传感器数据。信息共享的相关研究在国内外已经进行了多年，提出了许多概念和理论，如 III(Integrated Information Infrastructure)、IDM（Information Dessemination Managemeng）等。面向服务架构 SOA（Service Oriented Architecture）被认为是一种较好的信息资源共享的解决方案，它能充分利用现已部署的资源，将可用资源转换成独立的服务，被其它实体发现并使用[6]。在具体的实践中，美军远远领先于其它各国，有诸多的系统项目正在开发、逐步演进，甚至已经具备了实际应用能力。

2006 年 8 月，美国空军对 General Dynamics, Raytheon 和 SAIC 三家公司所开发的基于 SOA 的航迹信息传输系统进行现场测试。实验场景是将 Link16 数据链提供的航迹数据通过空中网络（Airborne Network Enviroment）分发给远端的数据中心，由远端的数据中心生成 COP(Common Operation Picture)。实验的主要目的是检验三家公司所开发的航迹数据分发系统的数据分发性能，通过实验来检讨现阶段存在的不足，为下一阶段的工作提供参考[7]。

General Dynamics 的 SOA 实现采用的是 Web Service 技术，航迹数据封装在

SOAP 消息中，通过 HTTP 协议传送给远端数据中心。从实验的结果来看，该方案的的数据投送率（成功接收数据报/发送的数据报）是最低的，数据的平均投送率只有 76.4%，航迹数据平均时延高达 31.206 秒。

Raytheon 的 SOA 实现方案使用基于 DDMS 标准（DoD Discovery Metadata Standard）[8]的元数据 Metadata（Metadata is often defined as being “data about data.”）实现航迹数据注册和发现服务，采用可配置的传输层实现数据传输，使用专用编码方式压缩航迹数据。该方案的数据投送率达到 92.43%，航迹数据平均时延是 1.178 秒，这两项指标均优于 General Dynamics 的方案。

SAIC 公司的 SOA 实现采用了 NDDS（Network Data Distribution Service）软件中间件[9]。NDDS 是 OMG 公司的 DDS(Data-Distribution Service for Real-Time Systems)规范的商用实现，DDS 是针对实时数据分发应用而制定的发布/订阅中间件规范，本课题对 DDS 规范进行了大量的研究工作。该方案的数据投送率在三者中是最高的，高达 97.5%，但同时也产生了较大的传输时延，平均时延高达 28.254 秒。

从上述实验的测试结果来看，将基于 SOA 设计的传感器数据分发系统应用于军事信息通信环境，在技术上是可行的，而且在一定程度上也表显现出了较好的性能。但此演示实验仅验证了通过军事信息网实现传感器数据分发的可行性，并没有过多考虑在实际应用中所面临的其它问题：

一、军事信息网带宽资源紧张的矛盾。从网络传输的角度来减少网络带宽资源浪费，减少冗余数据在网络中的传输，比如在一对多的应用场景下，可以将多个点对点的数据传输优化为组播传输。

二、数据的可控性。如何降低传感器数据在应用数据中所占的比例。从应用需求的角度来降低数据源向网络中注入的数据量。对传感器数据，可以根据用户需求缩小空域范围以降低数据量，也可以根据用户需求调整数据发送周期以降低数据量，做到“节源”。

三、数据源和用户节点的动态加入和退出。如果有新的数据源接入网络中，网络中的其它节点如何感知并使用其所提供的数据服务。反之如果现有的数据源退出网络，网络中正在使用其数据服务的节点如何应对，最大程度地降低可能带来的损失和后果。测试实验方案 3 所产生的高时延估计与用户节点的动态加入和退出有很大的关系。

第三章　传感器数据分发系统设计

发布/订阅机制的服务自动发现、动态扩展、事件过滤特性符合传感器数据分发系统的应用需求，是实现 “面向业务，按需分发”的理想方案。本章针对传感器数据分发系统的应用需求，设计了符合实际应用的软件结构和发布/订阅框架；依照“按需分发”的要求规划了传感器代理、作战子网网关、注册服务器的功能；结合“面向业务”思想和 SOA 方法分析了系统业务流程，定义了传感器数据分发系统的服务模块。

3.1 系统设计思路

一、设计方法－SOA

在 SOA 中，系统的设计和实现是围绕着“服务”进行的，应用系统由一个个独立服务按照一定的规则（应用流程）组合而成。在系统设计阶段，通过分析业务流程发现服务、定义服务；在系统实现阶段实现各个服务。服务应具备独立、自包含、可重用的特性。对传感器数据分发系统，传感器数据是数据对象，发布/订阅机制要对传感器数据提供注册服务、注册匹配服务、发布服务和订阅服务。

二、系统的核心－传感器数据

传感器数据分发系统的核心是传感器数据，分析传感器数据特征属性和组成，系统的设计和实现针对传感器数据进行。传感器数据具有很强的时效性，设计实现优先考虑数据分发的实时性，其次再考虑分发的可靠性。

三、基本思想－面向业务，按需分发

面向业务是以应用需求为驱动，分析业务流程，刻画数据对象特征属性，应用系统的设计和实现围绕应用业务进行。对本课题传感器数据分发系统，中心业务是传感器数据实时分发，数据对象是传感器数据，它具有生命周期、生成平台、目标位置等特征属性。

按需分发主要是实现带宽资源的有效利用，数据提供者根据数据请求者的需求向其分发数据，保证在合适的时间将合适的数据分发给合适的用户，尽量减少网络中对特定用户来说无用的或者冗余的信息，以此提高网络带宽资源利用率，缓解带宽矛盾。用户数据请求是依据数据对象的特征属性而定义的。实现数据按

需，首先要保证数据分发的实时性，将数据按需分发延伸为数据实时按需分发。

传感器数据的按需分发由两个过程实现。第一，注册服务器通过匹配发布条件和订阅条件为作战子网网关找到合适的传感器，这是第一个按需过程，定位符合需求的数据源。第二，传感器根据作战子网网关的订阅条件对传感器数据进行过滤，不符合订阅条件的数据就是冗余数据，不会通过 IP 骨干网分发给作战子网，这是第二个按需分发过程，分发符合需求的传感器数据。

四、兼顾其它一移植性等非功能性需求

采用发布/订阅框架完成诸多功能性需求外，在系统设计时还应该考虑软件移植性、软件升级性等非功能性需求，因此本课题采用将中间件技术、分层设计、服务模块化设计应用于系统的设计中。

3.2 系统软件结构

系统的软件结构采用了分层设计思想，将整个软件从下到上依次划分为操作系统层、中间件层和应用层，如图 3-1 所示。中间件层分为发布/订阅子层和公共服务子层，应用层也分为应用服务子层和应用子层。这样的设计方法是为了满足前面所提到的数据分发系统非功能性需求中的可移植性需求。该软件层次结构适用于传感器代理、作战子网网关和注册服务器。

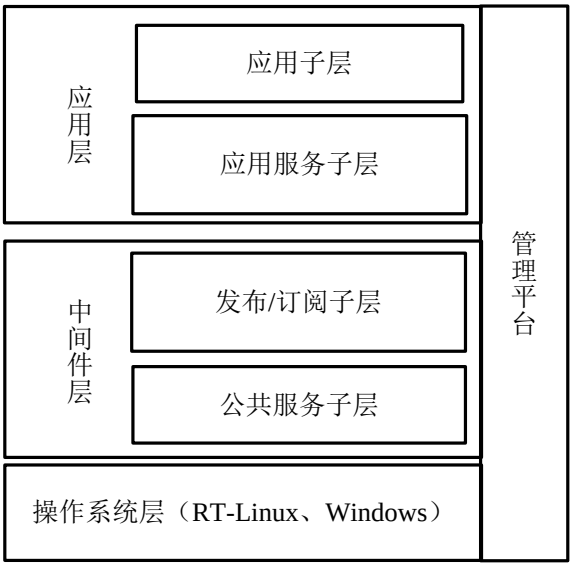

图 3-1 系统软件层次结构图

一、操作系统层：支持软件系统运行的平台，目前存在多种操作系统平台，

如 Windows 系列操作系统、Unix 类系统、RT-Linux 系统以及其它。本课题是为了验证传感器数据分发方案的可行性，建立一个验证演示系统，因此选择了应用广泛、软件开发配套工具齐全、方便易用的 Windows XP 操作系统平台。

二、中间件层：中间件层分为公共服务子层和发布/订阅子层。设计中间件层的作用有以下四点：

1、屏蔽平台差异，满足数据分发系统的系统移植性需求，使系统具有良好的移植性。

2、封装操作系统层的功能和服务，如存储器管理、I/O 管理、网络通信等，向上呈现统一的 API（Application Program Interface）接口。

3、提高软件模块的可重用性，缩短系统开发周期，节约成本。

4、提高软件质量，系统具有良好的稳定性。

关于中间件的出现、发展以及特点可以参看文献[33][34]。

发布/订阅子层将发布/订阅机制的功能封装成独立的服务模块，供上层应用直接调用。主要功能包括：

1、注册功能：发布者向网络中注册自己的数据发布能力，订阅者向网络中注册订阅需求，通过注册功能发布者和订阅者加入系统。

2、注册匹配功能：对发布者的发布能力和订阅需求进行匹配，为订阅者找到合适的发布者，或为发布者找到合适的订阅者。这是实现资源自动发现和透明定位的关键。

3、发布功能：将本地数据以合适的方式发布到网络中。

4、订阅功能：根据用户的订阅需求，接收符合本地订阅需求的数据。

公共服务器子层是支撑发布/订阅子层上述功能的中间件子层，包括传输服务、时间同步服务、存储器管理、进程管理等。

三、应用层：与中间件层相似，应用层也划分为应用服务子层和应用子层。应用服务子层包含与应用密切相关的服务或组件，如传感器数据标准化组件、消息映射组件等。应用子层综合使用中间层和应用服务子层的组件或服务实现系统的应用业务。

四、管理平台：主要对各层相关功能实施管理和监控，获得各层运行参数，处理各层产生的异常情况；提供人机接口，实现应用者对系统的控制。

3.3 发布/订阅框架设计

发布/订阅有多种实现框架，典型的如消息中间代理框架。本节结合传感器数据分发系统的应用场景和需求，从事件声明方式、匹配模式和通信框架三个方面出发，设计了符合实际情况的发布/订阅框架结构。

3.3.1 事件声明

传感器数据分发系统的事件声明采用主题、内容和类型相结合的方式，确立了主题(Topic)是核心，内容(Content)是扩展，类型(Type)是辅助的原则。事件声明方式影响着注册匹配以及按需分发的实现方式。

传感器数据分发系统采用主题、内容和类型相结合的方式，既提高了匹配效率（特别是当系统规模较大时），又解决了 Topic 方式匹配粒度过粗且不易扩展的问题，因为 Topic 通常是实现规划好的，一旦确定就不易改变，不能适应新的应用需求。Type 方式主要是为了简化编程实现。

传感器数据格式及其标准格式详细的列在表 2-1 中。对传感器应用以及传感器数据结构分析之后，将以下六点作为实现按需分发的切入点，称为按需分发条件：

1、传感器类型：生成传感器数据的传感器类型。不同的传感器，目标属性描述方式、工作范围、目标信息精度会存在着差别，不同的传感器用于满足多种应用需求。比如对远程预警需求来说，可以只请求那些探测范围广、数据精度相对不高的警戒雷达的数据；对协同作战应用来说，请求那些探测范围相对小，但数据精度高的火控雷达数据。传感器类型描述了传感器基本的能力属性，是按需分发最粗略的描述条件，是按需分发条件的第一个构成要素。

2、能力区域/目标区域：能力区域描述传感器的探测区域，对传感器，能力区域是随着地理位置变化，由当前的地理位置和平台的探测能力共同决定。用户通过目标区域描述兴趣区域，向传感器请求兴趣区域的传感器数据。能力区域和目标区域的描述有多种方式，本文使用平面二维坐标进行区域描述，高度不包括在内，这符合实际应用。本课题的演示系统，采用方格法来描述能力区域和目标区域，方格的两个对角顶点坐标唯一的确定区域。如图 3-2 所示，图中所标示的目标区域，通过｛A,B｝唯一确定。采用方格划分法不是一种最好的划分方法，但这种划分方法简单，所描述的区域不会出现重叠或者漏划的问题，在后续的开发中可以采用更优的方法，只需更改区域的描述方式和区域匹配算法即可。

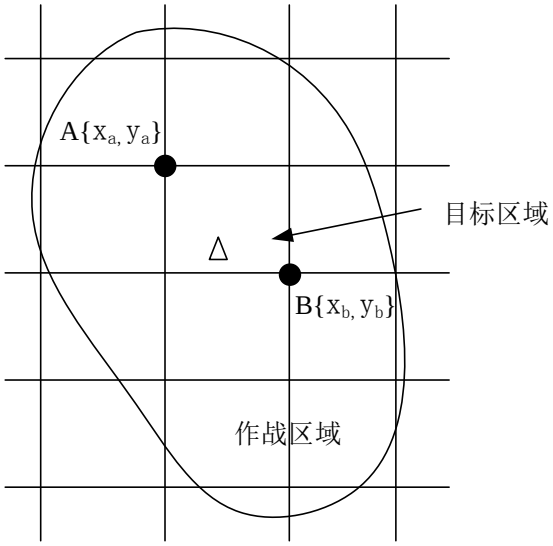

图 3-2 空域描述方法示意图

本课题的传感器数据使用地心空间直角坐标系，以{x,y,z}表征目标位置信息，但此坐标系描述的区域不方便使用，而且也不符合实际应用，在实际应用中，通常采用空间大地坐标系，以{longitude, latitude, height}表征目标位置。因此需要将目标数据由地心空间直角坐标系转换为空间大地坐标系。坐标系转换功能由传感器数据标准化组件实现，转化算法参看文献[36]。

能力区域是传感器的描述条件，目标区域是请求数据的描述条件，共同构成了按需分发条件第二个组成要素。

3、数据刷新率：是数据更新周期的倒数，是数据实时性的一方面。对机动速度高的目标，需要通过较高的数据刷新率（或者较短的数据更新周期）来更新其位置信息，对机动速度较低的目标，使用低刷新率的数据更新位置信息就可满足应用需求。战术互联网终端和数据链终端，对目标位置更新的时间要求均是 12 S，传感器能够提供刷新率大于等于 1/12 的数据就可满足要求。数据刷新率构成了按需分发条件第三个组成要素。

4、目标精度等级：描述传感器数据的质量，按照作战需求，将目标精度划分为态势级、战术级、武器级。发布者通过目标精度描述数据的质量等级，订阅者通过目标精度描述请求数据的质量等级。目标精度等级构成了按需分发条件第四个组成要素。精度等级高的数据可以满足相同等级和低等级的数据请求。

5、目标属性：目标属性指的是目标类型。在本课题的演示系统中，分为运输机、战斗机、轰炸机。用户终端可能只对特定类型目标感兴趣，而忽略其它类型的目标。目标属性仅用于订阅者描述数据请求，而不用于发布数据的描述，因为

传感器探测到的目标类型是动态变化的，不适合作为发布数据的描述，但可以作为过滤条件对源数据进行过滤。目标属性构成了按需分发条件第五个组成要素。

6、数据时延：这是数据实时性的直接体现，是数据生成时间和到达用户时间之间的差值。数据时延由数据用户在订阅条件中提出，数据发布者无法对时延做出准备的描述。数据时延构成了按需分发条件第六个组成要素。

对上述所列的六个按需发布条件，可以组合成为发布/订阅框架中的发布条件和订阅条件。表 3-1 是发布条件和订阅条件的组成，并规定了各组成项的描述方式。

表 3-1 发布/订阅条件定义及描述方式

发布条件	描述方式	订阅条件
传感器类型 Sensor_type	Topic	传感器类型 Sensor_type
能力区域 Detect_area	Content	目标区域 Intent_area
数据刷新率 Update_rate	Content	数据刷新率 Update_rate
目标精度等级 Precision_level	Content	目标精度等级 Precision_level
	Content	数据时延 Latency
	Content	目标属性 Target_type

3.3.2 匹配模式

发布/订阅有分布注册匹配模式和集中注册匹配模式两种匹配模式。

对分布注册匹配模式，系统完整性很难保证。系统完整性是指所有的发布者能够全面掌握系统中存在哪些订阅者，系统中的订阅者也能够全面地掌握系统中当前存在哪些发布者。系统不完整性主要是由于通信信道不是充分可靠的，在发布者广播发布条件时，发布条件可能不会到达所有的订阅者节点，结果就是没有收到发布条件的订阅者就不能参与到系统中，由此造成信息发布的不充分性（即发布条件没有到达所有的订阅者），从而导致系统的不完整性。另外对分布注册匹配模式，发布节点和订阅节点之间信息资源（发布条件/订阅条件）的交互、查找和路由需要占用一定的带宽资源，对带宽资源紧张的无线环境，分布注册匹配模式是不适合的。

对集中注册匹配模式，所有的发布者和订阅者都必须在注册服务器进行注册，注册服务器全面掌握系统中所有参与者的信息，系统是完整的。其次，发布条件和订阅条件直接发送给注册服务器，由注册服务器返回匹配结果，所占用的带宽

资源比分布注册匹配模式少。对于集中注册匹配模式所存在的低抗毁性、单点故障等不足，可以通过采取异地热备份或者建立数据容灾系统[36]加以解决。因此，本课题选择集中注册匹配模式作为传感器数据分发系统的注册匹配模式，接受传感器代理/作战子网网关的注册，为它们提供匹配服务。

3.3.3 通信架构

发布/订阅系统的通信架构有中心代理（Centralized Architecture）和非中心直连（Decentralized Architecture）两种架构。目前大多数发布/订阅系统采用的都是中心代理架构，但是此种框架并不十分适用于本课题所研究的传感器数据分发系统。

首先，中心代理架构需要设置专用的事件中介，这并不能有效的减少网络中的数据量，主要是由于发布者发给事件中介的数据并没有减少，网络中存在大量的冗余数据，不符合节约使用带宽的思想。

其次，由于中间代理的介入，导致发布事件到达订阅者的时间增加。文献[38]从吞吐量（Throughput）和时延的角度对比了中心代理架构和非中心直连架构，从对比结果看到，非中心直连架构的上述两种性能优于中心代理架构。传感器数据分发系统，分发的对象是传感器数据，传感器数据具有较强的时效特性，随着系统规模的动态变化，系统中需要分发的数据量可能会比较大（相对于系统的通信带宽而言）。文献[39]提出，对于实时的发布/订阅系统，应该优先考虑采用点对点的通信方式，即非中心直连架构。

第三，非中心分布式架构还具有较强的鲁棒性。集中注册匹配模式的注册匹配服务将发布者和订阅关联之后，就不再参与发布者和订阅者之间的分发过程，因此中心注册服务器失效也不会对已进行的通信过程造成影响，只会影响注册的更新和新的数据通信过程的建立。

第四，非中心直连架构便于发布者和订阅者之间的交互，可以更方便的实现数据的按需分发。

通过比较，结合传感器数据实时分发要求，本课题采用非中心直连架构作为传感器数据分发系统的通信架构，在完成注册匹配之后，传感器代理和作战子网网关直接进行通信。后面的发布/订阅框架均指的是发布者和订阅者采用非中心直连架构的发布/订阅框架，其实这种发布/订阅框架是典型的 P2P（Peer to Peer）方式。

通信框架还存在数据交互模式的问题，即主动的推模式和被动的拉模式。对传感器数据分发系统采用主动的推模式。一方面是由于数据的流向是单向的，其次主动的推模式便于通信组织，比如当有多个订阅者请求同一发布者的信息时，发布者便于采用多播等方式实现数据分发。

发布/订阅框架是层叠在 IP 骨干网之上，在 TCP/IP 协议栈上实现。根据系统的需求，还可以采用单播、多播（Multicast）等多种方式来实现系统大吞吐量、实时的数据分发要求。

3.3.4 小结

在前面的研究中，结合传感器数据分发系统应用需求，研究了对发布/订阅的事件声明方式、匹配模式和通信架构的多种实现方案进行了比较，选择了合适的方案，现总结如下：

传感器数据分发系统的事件声明方式采用主题、内容和类型相结合的方式，主题是核心，内容是扩展，类型是辅助。匹配模式采用集中注册匹配模式，在系统中设置注册服务器，实现发布条件和订阅条件的注册和匹配。采用非中心直连通信架构，传感器数据发布者以主动的推模式将数据直接分发给订阅者。

图 3-3 以 UML 顺序图的方式说明基于发布/订阅框架的传感器数据分发系统诸要素之间的交互顺序。

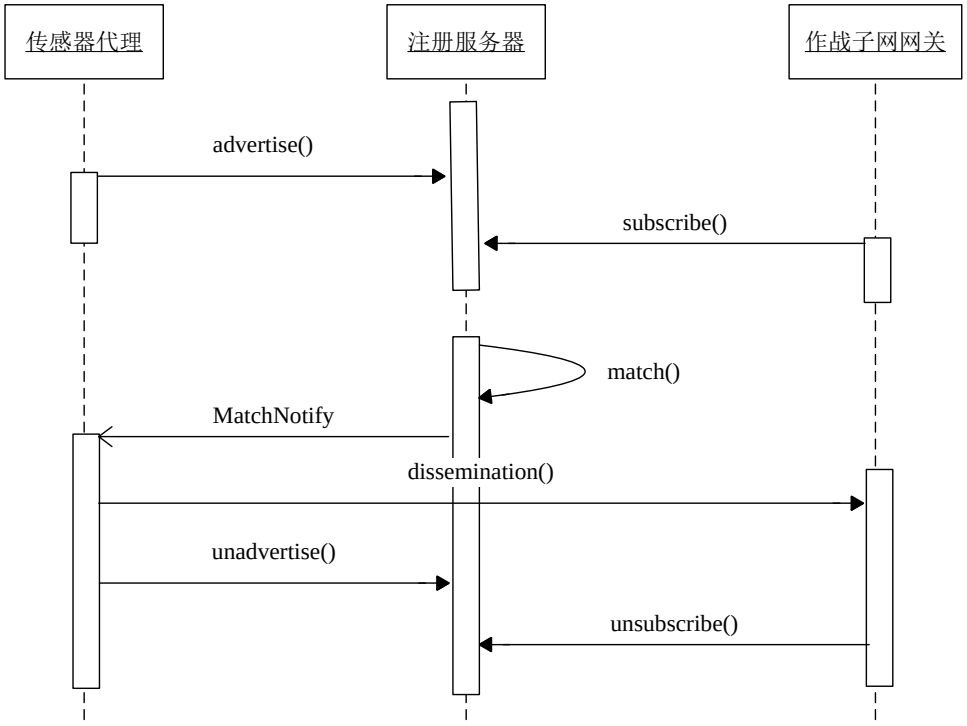

图 3-3 发布/订阅实体交互顺序图

3.4 应用业务和功能

本课题的目标是将传感器数据实时地分发给战术互联网和战术数据链的用户终端，本节通过图 3-4，阐明系统具体的应用业务，设计注册服务器、传感器代理和作战子网网关的主要功能。

3.4.1 应用业务

图 3-4 是传感器数据分发系统网络设备部署图，IP 骨干网将多个传感器和作战子网互连起来。传感器代理将所代理的传感器生成的数据以发布/订阅的方式分发给战术互联网网关节点。作战子网网关在接收到订阅的传感器数据之后，再将其分发给网内的用户终端。

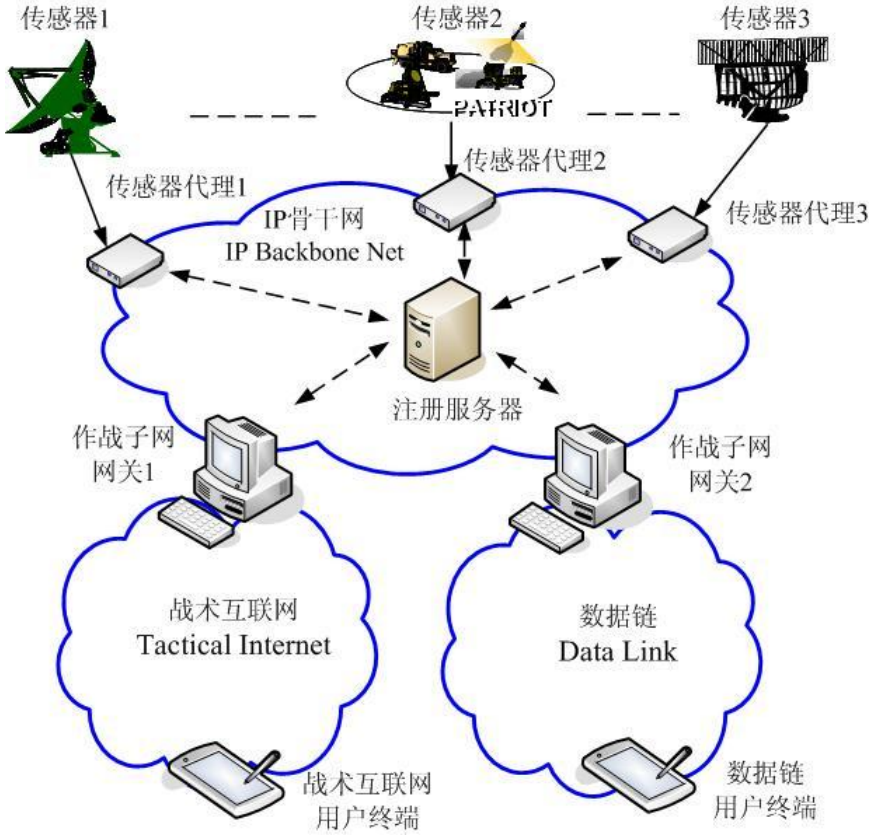

图 3-4 传感器数据分发系统网络设备部署图

在 2.1.4 节中，提出了单源选择分发、多源选择分发和多向分发三种应用需求，结合上面的网络设备部署图，将三种需求具体化为以下三种应用业务：

一、单源选择分发：作战子网网关 1 向网络订阅传感器数据，经注册服务器匹配之后发现传感器 1、2、3 均符合订阅条件，但战术互联网网关 1 仅需要一个传感器提供的数据就可满足需求，此时只需在传感器 1、2、3 中选择一个数据源向战术互联网网关 1 发布数据。这样的分发模式可用于远程预警应用。

二、多源选择分发：与上述模式相似，战术互联网网关需要传感器 1、2、3 同时向其发布数据。这样的分发模式应用于传感器数据融合应用，如 CEC，将多源的传感器数据融合成质量更高的数据。

三、多向分发：这是从传感器代理的角度出发制定的应用业务，只要传感器代理能同时向多个订阅者发布数据即可。

在上述三种场景中，作战子网网关不需要自己掌握传感器的位置，只需向注册服务器注册，发送发布条件，只要存在符合订阅条件的传感器，注册服务器就能为作战子网网关定位合适的数据源。数据源自动发现能力是发布/订阅机制的重要特性，是系统动态扩展能力的基础，它使得发布/订阅机制在多源数据应用场景

中比 Client/Server 等机制更具优势。

3.4.2 注册服务器

注册服务器采用集中注册匹配模式，部署于 IP 骨干网中，接受传感器代理/作战子网网关的注册，向其提供注册匹配服务。注册服务器对应于 SOA 结构中的服务注册中心，提供服务注册和查询。注册服务器需要具备如下功能：

一、接受登记传感器代理和作战子网的注册。

二、为数据发布者（传感器代理）在已注册的订阅者中查找符合发布条件的订阅者（作战子网网关）。这种匹配方式是实现数据推模式的前提条件。

三、将查找结果（匹配结果）返回给数据发布者。

3.4.3 传感器代理功能

传感器通过传感器代理接入 IP 骨干网络，将传感器数据发布到 IP 骨干网中。传感器代理是传感器数据的发布者 Publisher，对应于 SOA 结构中的服务提供者。传感器代理的功能规划如下：

一、与传感器节点建立通信链接，接收传感器生成的传感器数据。

二、向注册服务器注册，提交发布条件。

三、接收注册服务器的匹配结果，解析返回结果中的订阅条件，形成传感器数据过滤器，对传感器数据进行过滤。

四、将传感器数据转换为统一的消息格式。

五、将传感器数据实时的发布给合适的订阅者。传感器数据分发需要满足实时按需分发的要求，即传感器向外分发的数据必须符合用户的数据请求，并不是自身产生的所有数据。使用返回订阅条件形成的数据过滤器对传感器数据过滤之后，再向外发布，这样就可减小网络中的冗余数据。

3.4.4 子网网关功能

战术互联网和数据链通过作战子网网关接入 IP 骨干网络中，子网网关是数据订阅者，对应于 SOA 结构中的服务请求者，向网络请求数据服务。子网网关需要具备如下功能：

一、汇聚子网内内各用户终端的数据请求，生成一到多个订阅条件（一个订阅条件对应一个 Topic），订阅条件周期更新以体现用户终端的需求变化。数据链网络的数据请求通过自由文电的方式递交给数据链网关。

二、向注册服务器注册，提交订阅条件。如果作战子网能够向网外作战平台提供数据时，也可以作为发布者向注册服务器注册，并向外发布数据。

三、将接收到的订阅数据映射为 VMF（战术互联网）和 J 消息（数据链）。

四、根据子网用户终端的数据请求，将封装了传感器数据 VMF/J 消息向子网内的用户终端分发。

五、子网网关在 IP 骨干网络侧，以发布/订阅机制获得合适的传感器数据，将这些数据映射为 VMF/J 消息，然后按照子网通信协议将传感器数据分发给网内用户终端。图 3-5 是战术互联网子网网关的协议栈转换图。

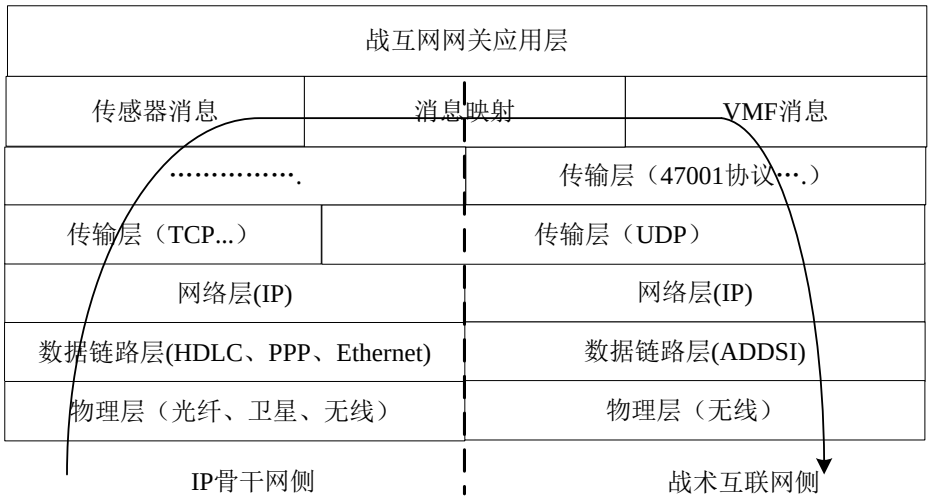

图 3-5 战术互联网子网网关协议转换图

3.5 服务定义和分解

SOA 方法围绕着服务进行系统的设计，本节按照 SOA 方法分析传感器数据分发业务流程，在系统中发现并定义了注册服务、注册匹配服务、发布服务和订阅服务四个顶层服务，然后将这些顶层服务分解成更基本的独立子服务或组件。

3.5.1 数据分发流程

在传感器数据分发系统中，共有五个要素参与了传感器数据的分发过程：注

册服务器、传感器数据生成者－传感器、数据发布者－传感器代理、传感器数据订阅者－作战子网网关、数据消费者－子网用户终端。下面是传感器数据分发顺序图，描述了上述五个要素之间的交互顺序。

按照 SOA 方法，对图 3-6 所示的数据分发流程进行分析，分解出四个主要服务：注册服务、注册匹配服务、发布服务和订阅服务。在确定了服务之后，接着采用 SOA 自顶向下的服务发现方法，分析上述四个顶层服务的工作流程，发现支持更多的子服务或支持组件。

要实现 3.4.1 节中的应用业务，关键在于注册匹配算法 match()，其次是数据实时分发过程，这将分别由注册匹配服务、发布服务和订阅服务实现。

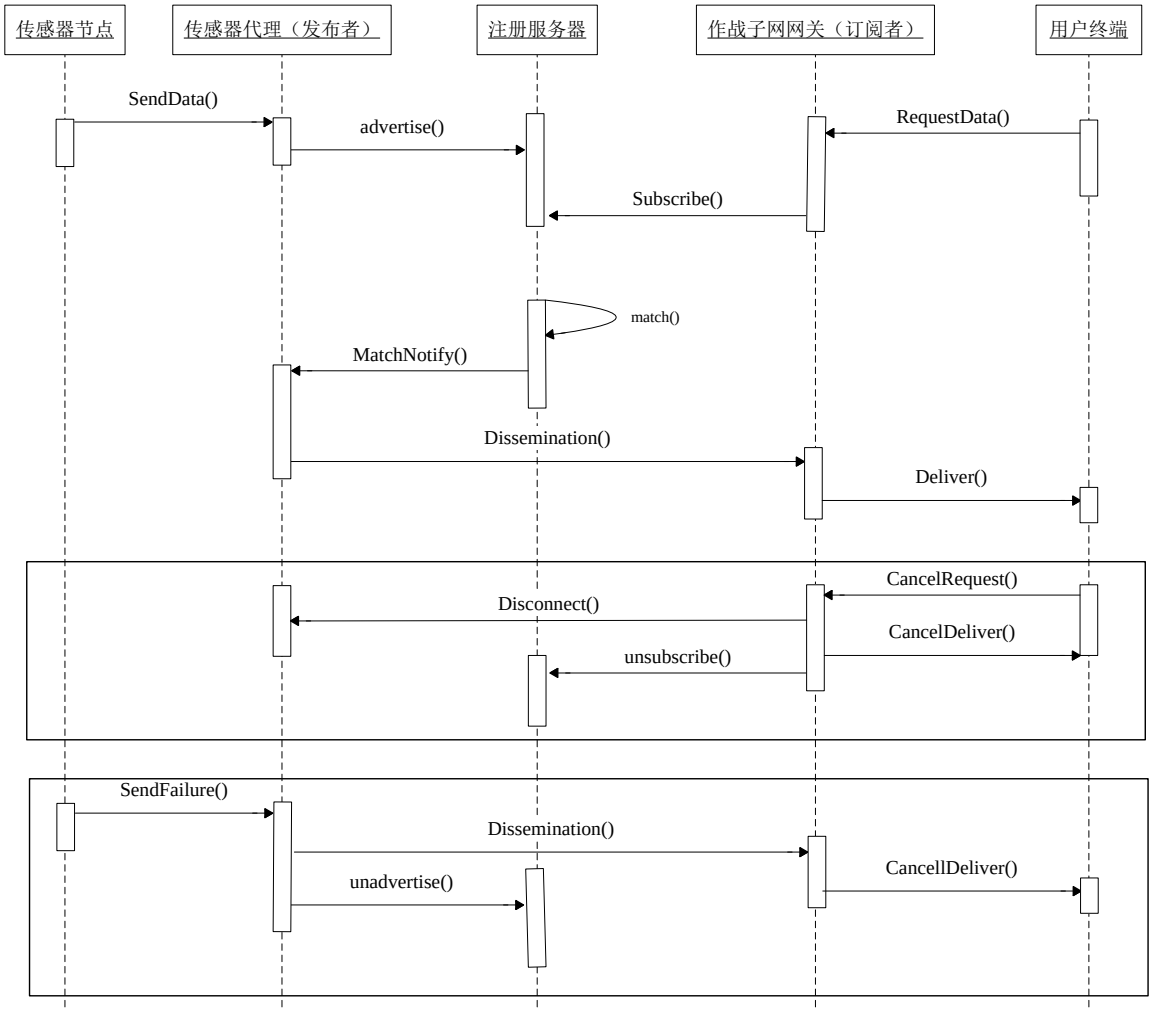

图 3-6 传感器数据分发顺序图

3.5.2 注册服务

注册包括传感器代理的注册和作战子网网关的注册。传感器代理的注册对应于图 3-6 中的 advertise()，其注册过程描述如下：传感器代理是传感器的网络接入点，代理传感器向网络中发布器数据。在本课题中，一个传感器代理代理一个传感器。当传感器代理接收到传感器发送的数据后，就认定传感器代理具备了向网络中发布数据的能力。传感器代理根据配置信息（所代理的传感器及其数据的属性的信息）生成发布条件，调用 advertise()操作向注册服务器注册。表 3-2 描述了传感器代理的注册流程。

表 3-2 传感器代理注册流程

Initial Actor	传感器代理	Other Actor	传感器、注册服务器
Business Event	传感器代理向注册服务器注册		
Business Rules	注册的过程必须是可靠的		
Event List	1、传感器代理开机，读取配置文件初始化传感器代理		
	2、将配置文件中的传感器属性转化成发布条件。发布条件包括 Topic（传感器类型）、能力区域、数据刷新率、目标精度。		
	3、与注册服务器建立通信链接连接，将发布条件发送给注册服务器，进行注册		
	4、如果注册成功，进入 6		
	5、如果注册失败，回到 3		
	6、等待注册服务器的匹配结果，至此注册过程结束		
	7、当传感器数据接收失败，向注册服务器注销发布条件		

作战子网网关的注册操作对应于图 3-6 中的 subscribe()，它的注册过程如下：作战子网网关代理网内的用户终端通过 RequestData()操作向子网网关提出数据请求，数据请求的具体格式在 4.3.2 节中进行了定义，子网网关汇聚网内的数据请求并生成订阅条件。子网网关通过 Subscribe()操作向注册服务器进行注册。表 3-3 描述了作战子网网关注册流程。

表 3-3 作战子网网关注册流程

Initial Actor	作战子网网关	Other Actor	子网用户终端、注册服务器
Business Event	作战子网网关向注册服务器注册		
Business Rules	注册的过程必须是可靠的		
Event List	1、作战子网网关开机，读取配置文件初始化网关节点		
	2、接受并登记子网用户终端的数据请求		
	3、解析汇聚用户终端数据请求，生成订阅条件。订阅条件项目包括 Topic（传感器类型）、目标区域、数据刷新率、目标精度、目标类型、数据时延		
	4、与注册服务器建立通信链接，将订阅条件发送给注册服务器，进行注册		
	5、如果注册成功，进入 7		
	6、如果注册失败，回到 4		
	7、等待注册服务器的匹配结果，至此注册过程结束		
	8、子网用户终端数据请求终止，向注册服务器注销相关订阅条件		

对传感器代理和作战子网网关注册过程进行分析，发现完成注册服务需要以下子服务和组件的支持：

一、子网请求管理分发服务：作战子网的用户终端是通过作战子网网关请求外部网络的传感器数据，作战子网网关服务负责接受登记子网内用户终端的数据请求，汇聚优化数据请求，形成数据请求列表（一个或者多个请求项生成一个订阅条件，一个订阅条件对应一个 Datareader ），当 Datareader 接收到订阅数据时，Datareader 查找与其相关的请求项表，将数据过滤并通过 Deliver()操作分发给相应的子网终端。

二、发布条件生成组件：应用于传感器代理，将传感器属性及其数据属性信息以 Topic 和 Content 相结合的方式映射为发布条件，发布条件的定义如图 3-7 所示。发布条件中的能力区域由传感器当前的位置和探测能力共同决定。

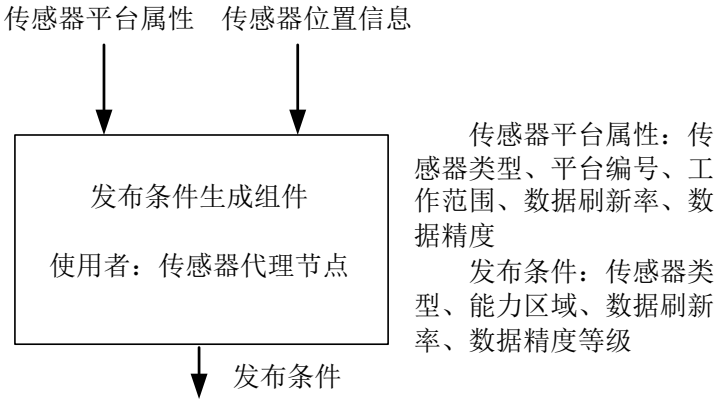

图 3-7 发布条件的定义

三、订阅条件生成组件：应用于作战子网网关，将作战子网用户管理组件汇聚之后的数据请求，以 Topic 和 Content 相结合的方式映射为订阅条件，订阅条件的定义如图 3-8 所示。

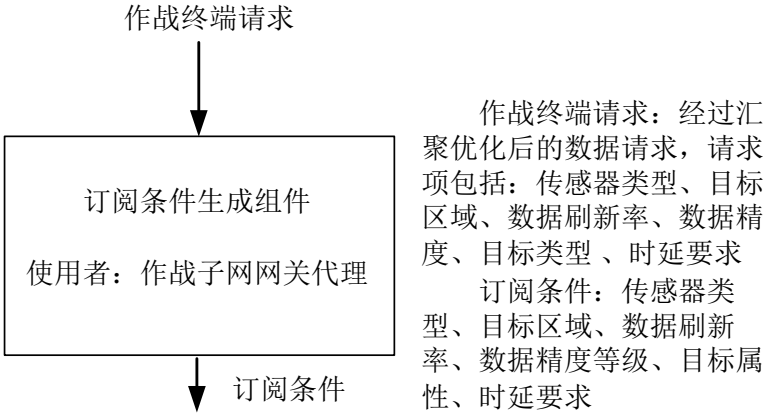

图 3-8 订阅条件的定义

四、可靠传输服务：为注册过程提供可靠的传输服务。可靠传输服务将在后面和实时传输服务等一起详细说明。

注册服务也涵盖注销过程，发布者通过 unadvertise()注销发布条件，订阅者通过 unsubscribe()注销订阅条件。注销会由多种原因引起：

（一）传感器关机或其它原因导致传感器向传感器代理发送数据失败。

（二）子网用户终端终止数据请求。

（三）其它异常情况导致注销过程的发生。

3.5.3 注册匹配服务

注册服务器使用注册匹配服务实现图 3-6 中的 match()操作，负责为注册的发布者找到符合发布条件的订阅者，然后将匹配的结果通过 MatchNotify()通知给发布者，这是为数据订阅服务寻找数据发布服务，是实现发布/订阅框架服务自动发现的关键。将订阅条件返回给发布者是为了发布者能够按照订阅条件中的目标区域、数据刷新率和精度等级和目标属性要求对发布者发布的数据进行过滤，实现更小粒度的按需发布。匹配的过程需要周期的执行，因为在系统运行过程中，可能还会有新的发布者和订阅者动态的加入和退出。因此注册匹配服务是实现数据源自动发现的关键。

注册匹配服务最为核心的问题就是匹配算法，匹配算法取决于发布条件和订阅条件的定义以及匹配策略。匹配是通过 match()操作完成。传感器数据分发系统的 match()是在主题、内容和类型方式相结合的基础之上实现的。当通过 match()操作为某一发布者查找到订阅者集合之后，注册服务器调用 MatchNotify()操作将匹配结果正确可靠的通知给发布者。

匹配策略与应用需求密切相关。对单源选择分发应用，注册匹配服务只需为订阅者找到一个符合订阅条件的发布者即可，但对多源选择分发应用，注册匹配服务却需要为订阅者找到多个发布者。针对 3.4.1 节的三种应用业务，设定三种匹配策略：ONE_MATCH_POLICY、MULTI_MATCH_POLICY、ONE_MANY_POLICY。匹配策略由订阅者根据应用需要指定，封装在订阅 Topic 的 Ownership Qos 中。

3.5.4 发布服务

在基于发布/订阅的 SOA 中，发布服务扮演数据服务提供者的角色。发布服务对应于图 3-6 中的 Dissemination()。发布服务将应用数据按照订阅条件按需的分发给数据订阅者，具体由 Datawriter 完成。Datawriter 在发布数据之前需要依照订阅条件进行数据过滤，数据发布过程受 Datawriter Qos 的控制。数据分发过滤在应用层调整 Datawriter 发布的数据，Datawriter 的 Qos 在中间件层通过控制网络资源和内存资源的分配来控制 Datawriter 的数据发布过程。

表 3-4 数据发布服务流程

Initial Actor	传感器代理	Other Actor	传感器、作战子网代理网关
Business Event	将传感器的数据分发给符合提出数据请求的用户终端		
Business Rules	传感器数据分发的实时性优先于可靠性		
Event List	1、传感器代理从传感器接收数据		
	2、对传感器数据进行标准化处理		
	3、将标准化之后的数据存放在本地缓冲区，新的数据不断覆盖老数据，以保证数据总是最新的		
	3、监测注册服务器返回的订阅条件和订阅者信息		
	4、根据返回订阅条件的数量创建多个 Datawriter，分别与订阅 Datareader 建立实时通信链接，发送队列和通信链接的创建受 Qos 控制		
	5、用订阅条件对每个 Datawriter 的待发布数据进行过滤，Datawriter 从本地缓冲区取数据，依照订阅条件对待发布的数据进行过滤		
	6、将过滤之后的数据实时地发布给 Datareader		

对传感器代理的分发服务流程进行分析后发现，完成发布服务需要以下子服务和组件的支持：

一、传感器数据标准化组件：将各类传感器生成的传感器数据映射为一种标准化的格式，如表 2-1 所示。

二、分发过滤组件：发布端使用注册服务器返回的订阅条件构成数据分发过滤器，对传感器发送的数据进行过滤，滤掉订阅者不需要的冗余数据，这也是实现按需分发的关键步骤。本文的按需分发体现在两方面，一是数据源的选择，通过发布/订阅框架的注册匹配服务实现；另一方面是传感器数据的过滤，在发布端根据订阅条件过滤掉对订阅者来说是冗余的数据。

三、实时传输服务：传感器数据实时地分发给符合发布条件的订阅者集合，它们的地址包含在注册服务器的返回结果中。数据分发的实时性和可靠性是相互制约的，在第一章中已经指出，传感器数据分发优先考虑实时性，其次再考虑可

靠性，因此数据发布服务应该在首先满足实时性的基础上兼顾可靠性。

3.5.5 订阅服务

在基于发布/订阅的 SOA 中，订阅服务扮演数据服务消费者的角色。订阅服务定义订阅数据的接收，具体由 Datareader 实现。Datareader 的订阅过程主要受 Datareader Qos 的控制。

表 3-5 数据订阅服务流程

Initial Actor	作战子网网关	Other Actor	传感器代理、作战子网终端
Business Event	接收传感器代理发布的数据		
Business Rules	数据订阅的实时性优先于可靠性		
Event List	1、作战子网网关等待传感器代理自动发起的的通信链接		
	2、订阅者将接收到的订阅数据存放在相应的 Datareader 的缓存区，接收和缓存过程受 Datareader Qos 的控制		
	3、Datareader 查找与其关联的子网数据请求，创建对应的数据过滤器		
	4、数据过滤器过滤 Datareader 缓存区中的数据		
	5、将过滤后的数据转换成 VMF/J 序列消息		
	6、把 VMF/J 序列消息分发给子网用户终端		

对作战子网网关的订阅服务流程进行分析后发现，完成发布服务需要以下子服务和组件的支持：

一、实时传输服务：与发布者共同实现数据的实时分发。

二、子网传输服务：用于作战子网网关将订阅到的数据分发给子网内的用户终端，实现由 TCP/IP 协议栈与战术互联网协议栈/数据链协议栈间的转换。

三、消息映射组件：将标准的传感器数据格式映射为战术互联网的 VMF 消息格式或战术数据链系统的 J 序列消息。

四、子网请求管理分发服务：在 3.5.2 中已经说明了功能，接受子网用户终端数据请求的登记，并将订阅到的数据分发给子网用户终端。

3.5.6 其它服务和组件

通过分析传感器数据分发流程，定义了注册服务、注册匹配服务、发布服务和订阅服务四个主要服务，以及一些子服务和组件。除此之外，还需要使用节点配置组件对系统中的节点进行初始化配置；使用时间同步服务实现系统节点间的时间同步。

一、节点配置组件：通过文件的方式对传感器代理/作战子网网关进行节点配置，配置项目包括节点内侧端口、网络侧端口、注册服务器地址、传感器属性（传感器类型、平台编号、工作范围、数据刷新率、数据精度等）以及节点正常运行的其它相关参数，其中传感器属于传感器代理节点的配置项。

二、时间同步服务：时间同步服务是为了使系统中所有的传感器、传感器代理、作战子网网关具有相同的时钟。这对传感器数据的实时分发是必须的：

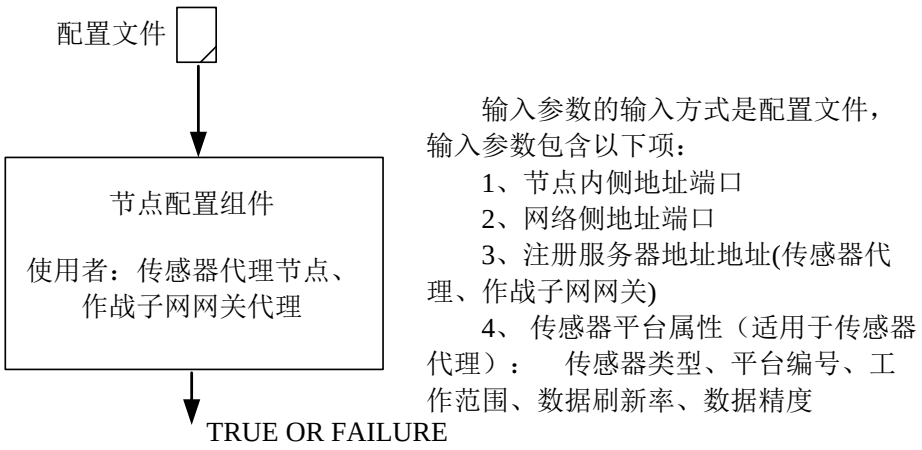

图 3-9 时间同步服务

1、在作战子网网关，需要根据订阅需求中的 Deadline、Latency_budget Qos 对已订阅到的数据进行实时验证，这需要实现传感器代理和作战子网网关实现时间同步。

2、对于多源选择分发应用，作战子网网关的 Destination_order Qos 需要使用传感器数据的生成时间对来自不同数据源的数据进行重新排序，这要求各个传感器具有相同的时钟。

对传感器数据分发系统，因为系统具有注册服务器这个中心节点，因此可以采用 Client/Server 模式的时间同步方案[40]，作战子网网关、传感器、传感器代理跟随注册服务器的时钟。数据链和战术互联网内的用户终端，本应跟随作战子网网关的时钟，但由于两者具备自有的时间同步机制，在本文中暂不考虑子网用户

终端的同步问题，这不会对数据分发系统的功能造成大的影响。图 3-10 是传感器数据分发系统时间同步关系图。

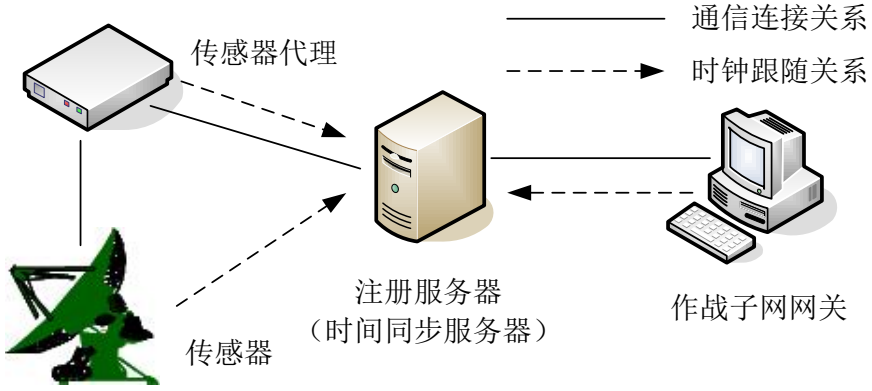

图 3-10 数据分发系统时间同步关系图

3.5.7 小结

本节通过分析传感器数据分发流程，定义了注册服务、注册匹配服务、发布服务和订阅服务四个主要服务，以及其它子服务和组件。现将上述服务和组件根据使用关系组织在一起，如图 3-11 所示。

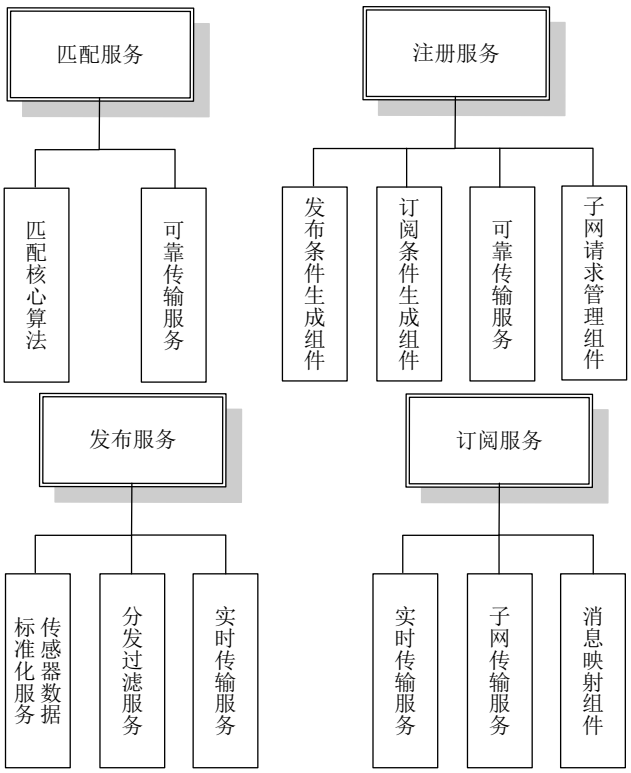

图 3-11 系统服务组织结构图

结合 3.2 节图 3-1，将上面的服务和组件填充在系统软件结构各层，使得系统的软件设计和实现结构更加清晰，填充之后的软件层次结构如图 3-12 所示。

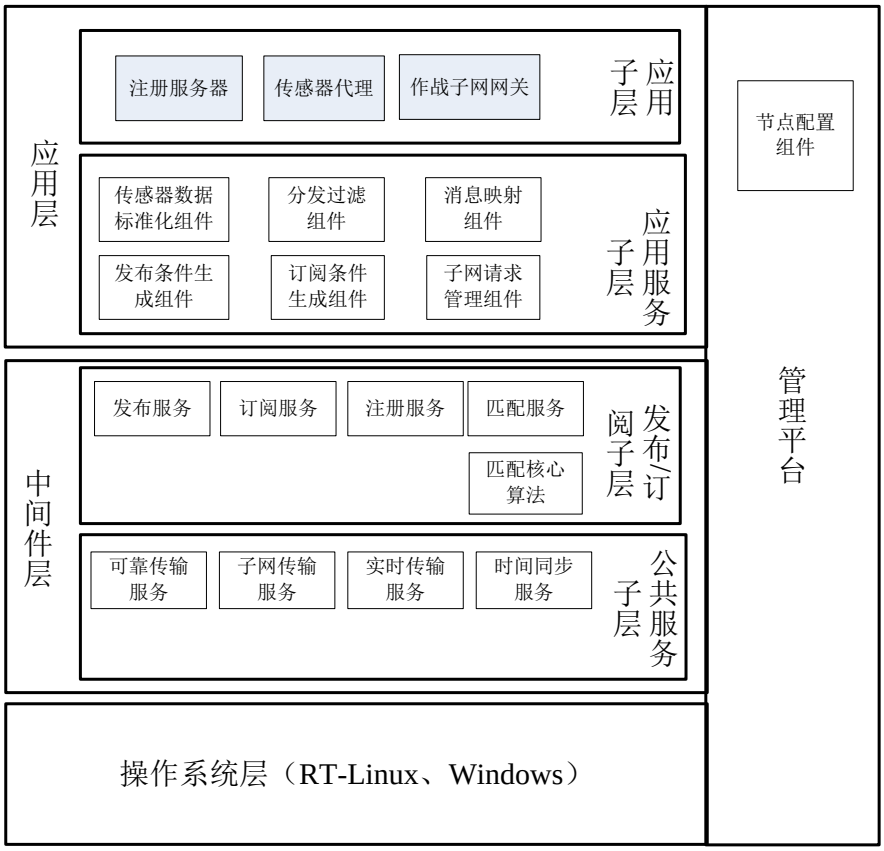

图 3-12 软件服务层次组织结构图

第四章　传感器数据分发系统实现

发布/订阅子层使用中间件技术封装了发布/订阅机制的实现细节，仅提供功能接口供开发者调用。中间件技术还屏蔽了底层硬件和操作系统的差异，能够实现软件异构平台移植。本章在 X86/WindowsXP 平台上移植了 ACE、TAO 和 OpenDDS 中间件，构建了中间件层。针对传感器数据分发应用，利用 DDS 规范的资源管理、通信控制、实时控制和数据附加属性等 Qos 在中间件层上实现了发布服务、订阅服务、注册服务和注册匹配服务，注册服务器、传感器代理和作战子网网关使用上述服务实现了满足实时按需分发要求的传感器数据分发系统。

4.1 系统的软硬件环境

本课题选择 X86 体系的 PC 机平台作为系统开发和演示验证的硬件平台，WindowsXP with SP2 作为软件环境，主要是出于以下三点考虑：

X86 体系结构应用最为广泛，后面将提到的 ACE、TAO、OpenDDS 中间件在 X86 平台上经过了严格的测试，具有可靠的稳定性。

X86 平台硬件平台可以使用广泛使用的 PC 机，演示系统环境搭建方便，成本最低。

Windows XP 是使用最广泛的操作系统，具有丰富易用的软件开发工具。本课题采用 Microsoft Visual Studio2005 作为软件开发环境，使用面向对象语言 C++作为编程语言。

表 4-1 系统开发环境软硬件配置表

硬件配置	CPU	Pentium1.5GHz	不低于此配置的 PC 机至少两台
	内存	1G	
	硬盘	80G	
	显示器	1 台	
	鼠标	1 只	
	键盘	1 副	
	网卡	10/100M 以太网卡	
	以太网交换机	1 个	
软件环境	直连网线	若干	
	操作系统	WindowsXP with SP2	
	开发工具	Microsof Visual Studio 2005、Ethereal 网络协议分析工具等	

4.2 中间件层实现

发布/订阅中间件是中间件层的核心，是实现发布/订阅机制的基本功能层。发布/订阅中间件采用遵循 DDS 规范[31]的开源软件 OpenDDS[41]，针对其存在的缺陷和不足做了自己的改进。本节通过移植 ACE、TAO 和 OpenDDS 中间件，构建了传感器数据分发软件的中间件层。

4.2.1 中间件层次结构

发布/订阅中间件 OpenDDS 是整个软件系统的核心，应用层的传感器代理应用程序、中心注册服务应用程序和作战子网网关应用程序均是在发布/订阅中间件之上开发的。发布/订阅中间件也是层叠在其它中间件产品之上的。下面是软件的实现分层结构图。

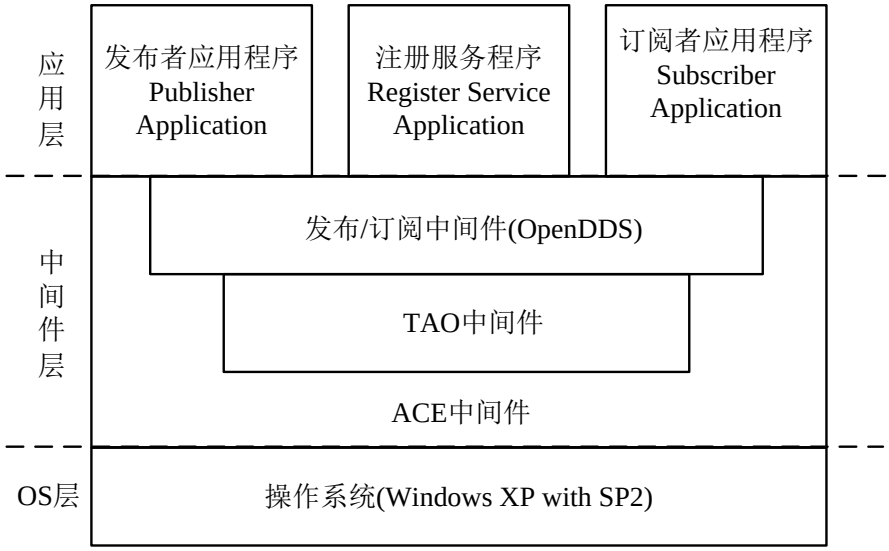

图 4-1 数据分发系统软件实现分层结构图

在操作系统层之上是中间件层，在本课题中设置中间件层的目的有两个：

1、通过设置中间件层，屏蔽底层操作系统的差异，增强应用软件的移植能力。

2、使用 ACE、TAO 和发布/订阅中间应用程序提供简单易用、统一的功能接口，降低应用软件开发难度，特别是 ACE 中间件的网络通信功能封装，简化了系统的编程任务。

中间件层的从下向上依次是 ACE（Adaptive Communication Environment）自适配通信环境、TAO（the ACE ORB）和 OpenDDS。这三个中间件软件之间的关系不是简单的上下关系，从软件结构图来看是一种“凹”字形关系。最底层的 ACE 向 TAO、OpenDDS 和应用层程序提供功能接口。

一、平台移植中间件－ACE

ACE 是一个功能丰富、面向对象的软件开发包，由美国华盛顿大学 Douglas C.Schmit 博士领导开发。它将并发同步、进程间通信、内存管理等与操作系统平台相关的 API 封装成统一的功能类，使得采用 ACE 编写的程序代码可以无差别地运行在各种系统平台上。ACE 提供的功能类简化了封装了复杂的实现细节，使得程序开发人员专注于软件功能的实现，不必过多的关注实现细节。ACE 共分为四层：ACE 操作系统适配层、ACE 面向对象包装层、ACE 框架层和 ACE 网络服务组件层[42]。

ACE 中间件屏蔽操作系统平台的差异性，简化了发布/订阅中间件和传感器数据分发应用软件的开发和移植工作。另外 ACE 面向对象的开发包为发布/订阅中间件和应用软件的开发提供大量简单易用的功能封装和实现类。在传感器分发应用

软件的实现中用到了 ACE_Task、ACE_Thread_Manager 等功能类实现线程派生和管理，使用 ACE_Message_Block 用于消息队列实现线程间通信，使用 ACE_SOCK_Dgram、ACE_SOCK_Dgram_Mcast 实现网络数据报通信，另外使用 ACE Connector 和 Acceptor 实现通信的链接和管理。上述功能包和类的具体使用可参看 ACE 源码和文献[42][43]。本课题采用 ACE version 5.5.9 版本[44]。

二、CORBA 中间件－TAO

TAO（the ACE ORB）是通用请求对象请求代理体系结构 CORBA（Common Object Request Broker Architecture）的开源实现，由 OCI 组织开发，经过了严格的测试，目前应用于许多商用软件系统中，例如 RTI 公司的 NDDS。使用 ACE 提供的软件概念和框架构建了对象请求代理 ORB（Object Request Broker）。它遵循对象管理组织(OMG)标准的 CORBA3.1 参考模型规范[46]，封装了底层的网络编程实现工作，专注于上层应用的开发。TAO 具有很高的性能和可靠性，特别适合实时的嵌入式应用。本次移植所采用的 TAO 的版本是 Ver1.5.9[45]。

本课题采用了 CORBA 中间件 TAO 来实现信息发布者和信息订阅者的注册服务。因为 CORBA 基于 Client/Server 机制，适合于服务器（Server）向一到多个客户端（Client）提供服务的应用。传感器数据分发系统的注册匹配服务是由注册服务器向多个传感器代理和作战子网网关提供注册匹配服务，与 CORBA 的应用场景相符。

在本课题中，CORBA 中间件 TAO 的主要功能有两个：

1、TAO 的 ORB 为注册服务器和传感器代理/作战子网网关之间建立了分布式的软件总线，用于实现传感器数据分发系统的注册服务。ORB 软件总线基于 Client/Server 交互模式和可靠的传输层协议，适用于注册服务的可靠通信需求。传感器代理和作战子网网关通过 CORBA 对象调用简化了注册匹配服务请求。

2、CORBA 的接口定义语言 IDL（Interface Definition Language）用于应用数据结构的定义，所定义的数据结构具备平台通用性。IDL 数据定义经过编译器编译之后，为每种应用数据（对应于各个 Topic）生成相应的代码框架：TopicTypeSupportC.h TopicTypeSupportC.cpp 、TopicTypeSupportS.h TopicTypeSupportS.cpp、TopicType SupportImpl.h、TopicTypeSupportImpl.cpp。这些代码文件包含了与 Topic 相关的 Datareader、Datawriter 的操作实现代码，如 rigister_instance()、write()、read()，其它相关操作和功能参看文献[31]。这些数据类型相关的代码既用于 TAO 所实现的注册服务，也用于 DDS 所实现的数据分发过程，便于开发编程。

三、发布/订阅中间件－OpenDDS

OpenDDS 是由 OCI 组织开发的面向数据的发布/订阅中间件，源代码完全公开。它遵循 OMG 的 DDS Ver1.0 规范，使用了 ACE 和 TAO 中间件，实现了 DDS 规范 DCPS 层的主要功能以及部分 Qos，是发布/订阅框架的核心。使用 OpenDDS 中间件，可以开发面向数据、基于发布/订阅机制的应用系统，在系统节点间实时地分发数据。OpenDDS 目前只实现的是 DDS 规范 DCPS 层的功能子集。本课题采用 OpenDDS Version 1.0 版本。

目前遵循 DDS 规范主要的中间件产品有 RTI 公司的 NDDS[9]、Thales Netherlands 公司的 Splice-DDS[47]和 OCI 组织的 OpenDDS。前两种是商用软件产品。文献[48]比较了三种产品的结构和传输层支持能力，并测试了三种软件的数据分发时延以及时延抖动。从测试结果来看，虽然 OpenDDS 的性能与两款商用产品有差距，但相差不大。另外 OpenDDS 源码完全公开，使用 C++语言，便于根据设想和需求改造。基于上述原因，选择 OpenDDS 作为发布/订阅中间件。

OpenDDS 发布/订阅中间件，使用 ACE 和 TAO 中间件的提供和接口，实现了面向实时数据分发、Topic 方式的发布/订阅功能层。OpenDDS 构建在 ACE 和 TAO 中间件之上， ACE 中间件提供平台的移植性。OpenDDS 通过 TAO 的 ORB 软件总线为注册服务提供注册通信通道。OpenDDS 为数据发送和接收创建 I/O 线程，数据传输通过嵌入式传输层实现。嵌入式传输层包含多个传输层协议，可以通过变量指定使用哪个传输协议完成数据传输。

4.2.2 中间件的移植

本节的工作是完成 OpenDDS 在 Windows XP 操作系统平台上的移植。OpenDDS 是在 ACE 和 TAO 之上开发的中间件产品，而 TAO 也是基于 ACE 所开发的 ORB 中间件，所以要完成 OpenDDS 在 Windows XP 上的移植，首先需要完成 ACE 在 Windows XP 上的移植，在此基础上完成 TAO 在 Windows XP 上的移植，最后完成 OpenDDS 的移植工作。

一、移植软件准备和安装

1、ACE：Ver5.5.9；

2、TAO：Ver1.5.9；

3、OpenDDS：Ver1.0。

将上述三个压缩文件解压到“../OPENDDS”目录下。

要完成移植，还需要安装 Active Perl Version 5.8.8.822 脚本语言软件和 Microsoft Visual Studio2005 开发环境。Active Perl 用于在编译和测试过程中执行 Perl 脚本文件，VS2005 是中间件的编译环境和应用程序的开发环境。

二、设置环境变量

ACE_ROOT=..\ACE_wrappers；

MPC_ROOT=..\MPC;

TAO_ROOT=..\ACE_wrappers\TAO;

DDS_ROOT=..\DDS;

PATH=%DDS_ROOT%\bin;%DDS_ROOT%\lib;%ACE_ROOT%\lib; ACE_ROOT%\bin；

三、源码编译

在设置好环境变量之后，就可以进行软件在 X86 平台、Windows 操作系统上的移植工作。首先在%ACE_ROOT%\ace 文件夹下新建文件 config.h，输入指令行以指定移植平台和环境：

#include "ace/config-win32.h"

至此就可以使用 VS2005 开发环境编译中间件源码，编译文件和顺序如下：

加载工程%ACE_ROOT %\ace\ace_vc8.sln，生成 ACE 的解决方案；

加载工程%TAO_ROOT %\TAO_vc8.sln，生成 TAO 的解决方案；

加载工程%DDS_ROOT %\DDS_vc8.sln, 生成 OpenDDS 的解决方案。

源码编译完成之后的生成的库文件存放在%DDS_ROOT%\bin; %DDS_ROOT%\lib; %ACE_ROOT%\lib 文件夹。

在完成上述编译过程并成功生成库文件之后，可以运行源码中的测试程序验证移植工作的正确性。

Perl %ACE_ROOT%\tests\run_tests.pl

Perl %DDS_ROOT%\tests\run_tests.pl

至此中间件的移植工作完成，可以在中间件层之上开发发布/订阅应用程序。

4.3 服务实现

在系统设计中，通过分析业务流程发现并定义了多个服务和组件，本节将具体阐述其中部分服务的具体实现，这些服务包括：传输服务、注册服务、注册匹配服务、发布服务、订阅服务、子网请求管理分发服务和时间同步服务。

4.3.1 传输服务

在此将上章定义的实时传输服务、可靠传输服务和子网传输服务归结为传输服务。实时传输服务和可靠传输服务基于 TCP/IP 协议栈实现，子网传输服务基于战术互联网和战术数据链的通信协议栈实现。传感器代理使用实时传输服务把传感器数据实时地分发给作战子网网关，作战子网网关使用子网传输服务将订阅到的数据分发给子网内的数据请求节点。作战子网网关和传感器代理使用可靠传输服务实现注册服务。

一、实时传输服务

实时传输服务是传感器数据分发系统对传输服务提出的要求，应用于传感器代理和作战子网网关间实现数据实时通信。实时传输服务的实现从两方面入手，一是在通信端点使用 Qos，为实时传输任务指定实时要求，预分配通信资源；另一方面选择适用于实时传输的传输层协议。

1、Qos 控制

在发布者端和订阅者端优化设置 DDS Qos 提高数据分发的实时性。实现实时传输服务用到了以下 Qos：Deadline、Latency_budget、Reliability 和 Transport_priority。

（1）Deadline：数据发布者使用 Deadline Qos 表征传感器数据的更新周期，数据订阅者使用该 Qos 表示期望的数据更新周期。Dealine Qos 与发布条件和订阅条件中的数据刷新率对应。Dealine 需要赋值给 Topic、Datawriter 和 Datareader 的 Deadline Qos。

```
//发布者端 Deadline 设置
topic_qos.deadline = 1/Update_rate;
dw_qos.deadline = 1/Update_rate;
//订阅者端 Deadline 设置
topic_qos.deadline = 1/Update_rate;
dr_qos.deadline = 1/Update_rate;
```

在注册服务器中进行注册匹配时，如果发布条件的 topic_qos.deadline 小于订阅条件的 topic_qos.deadline，则表明发布者的 Deadline 满足订阅者的 Deadline 需求。发布者按照 dw_qos.deadline 发布数据，订阅者依据 Datareader 的 dr_qos.deadline 对接收到的数据进行有效性验证，如果发布者发布的数据的到达时间间隔大于 Deadline 时，判定数据失效，丢弃该数据包。

本课题的传感器的数据更新周期是 6 秒，战术互联网和战术数据链的用户终端请求的数据刷新周期是 12 秒，通过上述代码为 Deadline Qos 赋值。

（2）Latency_budget：这是实时性的具体落脚点。在订阅端，将订阅条件中的 Latency 赋值给 Topic 和 Datareader 的 Latency_budget Qos，设定订阅者对数据的实时性要求。由于发布端不清楚订阅者的实时性要求，因此将此 Qos 先设置为一个较大的值，以便先与 Datareader 达成匹配。当 Datareader 的订阅条件和 Datawriter 的发布条件匹配时，注册服务器将订阅条件返回给 Datawriter，此时再将订阅条件中 Datareader 的 Latency_budget Qos 赋值给 Datawriter 的 Latency_budget Qos，之所以这样使用是因为 Latency_budget 具有 Changeable 特性（Qos 能够在与之相关的实体创建之后继续更新）。当数据到达订阅端时，Datareader 将数据时延与其 Latency_budget Qos 对比，如果时延小于其 Latency_budget，符合实时要求，数据上传给应用层，反之丢弃。

Latency_budget 与 Transport_priority 配合使用，低的 Latency_budget，将映射为高的 Transport_priority。下面以战术数据链为例，列举相关的程序代码：

```
//订阅者端 Deadline 设置
topic_qos.latency_budget = Deadline_Link16；
dr_qos.latency_budget = Deadline_ Link16；
//发布者端 Latency_budget 设置,首先设置为一个较大的值
topic_qos.latency_budget = 1000；
dw_qos.latency_budget = 1000；
…….
//将订阅条件的 latency_budget 赋值给 Datawriter
dw_qos.latency_budget = latency_budget_sub；
```

（3）Transport_priority：设置 Datawriter 发送数据的优先级，通过优先获取通信资源的方式提高实时性。为 dw_qos.latency_budget 小的 Datawriter 设置高的 transport_priority 以优先获取通信资源，降低数据在本地的等待时间，缩短整个分发过程的时间。

当前 Transport_priority 设为两个等级，对应的 Latency_budge 分别为 0-6、6-12（根据战术互联网和战术数据链的实时要求）。

（4）Reliability：可靠性和实时性是相互矛盾的，要获得高实时性，就需要牺牲可靠性。因此选择可靠性低的传输协议以获得更高的实时性，将 Reliability Qos 的 kind 参数设置为 Best Effort。

topic_qos.reliability=BEST_EFFORT_RELIABILITY_QOS;

dw_qos.reliability=BEST_EFFORT_RELIABILITY_QOS;

dr_qos.reliability=BEST_EFFORT_RELIABILITY_QOS;

2、传输层协议

选择合适的传输层协议来提升数据通信过程的实时性。目前 OpenDDS 支持的传输协议有 TCP、UDP、ReliableMcast 和 SimpleMcast（）。这里的 ReliableMcast 和 SimpleMcast 是多播协议的简单实现，它们都是基于 UDP 传输层协议，SimpleMcast 是尽力而为的数据多播传输，ReliableMcast 是在其基础上增加了确认机制以实现可靠多播。根据经验，在有线信道等理想的通信环境下，TCP 和 UDP 的实时性相差不大，在通信环境较差的情况下，UDP 协议由于取消了重发验证机制，能够达到较高的通信速率，实时性优于 TCP 协议,ReliableMcast 和 SimpleMcast 的实时性比较相似。

传输层协议的选择与 Reliability Qos 相关，目前 Reliability Qos 的 BEST_EFFORT_RELIABILITY_QOS 对应 UDP 协议，RELIABLE_QOS 对应于 TCP 协议。

实时传输服务选择 UDP 或 SimpleMcast 协议作为传输层协议。

二、可靠传输服务

可靠传输与实时传输服务相反，可靠性是第一位的。发布者、订阅者和注册服务器使用可靠传输服务实现注册和匹配结果返回，也用于发布者和订阅者之间的数据分发。可靠传输服务实现同实时传输服务相似，也是通过端 Qos 控制和选择传输层协议的方式实现。传输协议选择可靠性的传输协议，如 TCP 协议。在此仅说明如何通过 Qos 控制如何实现可靠传输服务。

1、Reliability：将 Reliability Qos 的 kind 参数设置为 RELIABLE。Duration 和 RELIABLE 设置配合使用，设定数据包在发送队列中的驻留时间。当数据包发送失败后从发送队列中重新取出数据包再次发送。Duration 又受到 Resource_limits 的限制，Duration 越长，数据在发送端等待的时间越长，就需要更大的队列空间缓存发送数据。

2、Resource_limits： Datawriter 和 Datareader 通过此 Qos 申请可用的缓存资源，与 Reliability Qos 配合使用。 Resource_limits 设置越高，数据发送队列缓存空间越大，可以缓存更多的历史数据等待确认，实现更高的可靠性。

三、多播传输服务

适用于一对多的数据通信，相比于多个点对点的通信方式，多播可以节省网络的带宽资源。目前 OpenDDS1.0 支持简单的可靠多播传输（ReliableMcast）和非

可靠多播传输（SimpleMcast）。

四、子网传输服务

用于作战子网网关将订阅到的数据分发给作战子网内的用户终端，使用战术互联网和战术数据链的通信协议栈实现。

4.3.2 注册服务

一、注册条件描述

在 DDS 规范中，发布条件和订阅条件的描述是基于 Topic 的方式，通过对传感器数据分发系统的发布条件和订阅条件组成分析之后，认为将传感器类型作为 topic 的划分标准是最为适合的。在传感器数据分发演示系统中，topic 的命名方案如表 4-2：

表 4-2 Topic 规划表

传感器类型	Topic
火控雷达 1	FC_Radar_1
火控雷达 2	FC_Radar_2
警戒雷达 1	Guard_Radar_1
警戒雷达 2	Guard_Radar_2
红外雷达 1	Infrared_Radar_1
红外雷达 2	Infrared_Radar_2

Topic 实现的只是最粗劣的注册匹配，匹配粒度太大，并不能有效地实现按需分发，在 3.3.1 节提出了对 Topic 进行基于内容（Content_based）扩展的设想，将能力区域/目标区域、数据刷新率、目标类型、数据精度等级作为内容扩展项。

1、将发布条件和订阅条件的数据刷新率映射为 Deadline Qos。

2、将订阅条件的数据时延 Latency 映射为 Latency_budget Qos。

3、将发布条件除传感器类型之外的数据项定义为 PUB_CONTENT 结构体，在发布端将这个结构体映射为 topic_data Qos；将订阅条件除传感器类型之外的数据项定义为 SUB_CONTENT 结构体，在订阅端将这个结构体映射为 topic_data Qos，相关代码如下：

```
struct Sub_Content                    {
```

```
    SENSOR_TYPE    Sensor_type;
    AREA           Intent_area;
    FLOAT          Update_rate;
        INT            Latency;
    u_short        Precision_level;
    u_short        Target_type;
}SUB_CONTENT;
......
SUB_CONTENT
*topic_content_data;
......
topic_qos.topic_data = topic_content_data;
topic_qos.deadline = 1/Update_rate;
topic_qos.latency_budget = 1/Update_rate;
......
topic_FC_M = participant->create_topic(
                "FC_Radar_1",
FC_MEASUREMENT_TYPE,
                topic_qos,
DDS::TopicListener::_nil());
```

```
struct Pub_Content
{
    SENSOR_TYPE    Sensor_type;
    AREA           Detect_area;
    FLOAT          Update_rate;
    u_short        Precision_level;
}PUB_CONTENT;
......
PUB_CONTENT
*topic_content_data;
......
topic_qos.topic_data = topic_content_data;
topic_qos.deadline = 1/Update_rate;
......
topic_FC_M = participant->create_topic (
                "FC_Radar_1",
FC_MEASUREMENT_TYPE,
                topic_qos,
DDS::TopicListener::_nil());
```

除此之外，通过 IDL 语言，将主题方式和内容方式结合在一起。IDL 定义了每种 Topic 的数据结构，由 TAO 的 IDL 编译器生成与 Topic 数据结构对应的代码框架。下面代码是部分 Topic 的 IDL 定义。

```
typedef struct Air_Track
{
    unsigned short    platform_no;          //航迹数据产生平台编号
    unsigned short    track_no;             //此航迹点的航迹号
    unsigned short    Target_Identifier;    //目标属性标识数组
```

```
    TrackTimestamp      current_time;              //此航迹点的时间戳
    TargetPosition      current_position;       //目标的当前位置
    TargetVelocity      current_velocity;       //目标的三位速度
}AirTargetTrack;

#pragma DCPS_DATA_TYPE "RadarData::FC_Radar_Measurement"
#pragma DCPS_DATA_KEY "RadarData::AirTargetTrack.platform_no"
struct FC_Radar_Measurement
{
    AirTargetTrack     Track_point;
};
#pragma DCPS_DATA_TYPE "RadarData::FC_Radar_Track"
#pragma DCPS_DATA_KEY "RadarData::AirTargetTrack.platform_no"
struct FC_Radar_Track
{
    AirTargetTrack     Track_point;
};
……
```

二、注册通信方式

发布者和订阅者向注册服务器注册，然后等待注册服务器生成的匹配结果，这是一种典型的请求/应答应用模式，注册过程和匹配结果返回过程需要的是一个可靠的通信过程。Client/Server 通信模式是一个很好的解决方案，本课题使用了 TAO 中间件的 CORBA 功能实现注册传输服务。

在 CORBA 应用的体系结构中，服务器组件，能够向外提供服务或对象实现接口[49]，它把数据和与数据相关的操作捆绑在一起，封装在对象中，向外提供对象接口，这与面向对象编程中的类相似。客户组件，它通过对象引用的方式使用服务器上的对象实现或服务。在本课题的发布/订阅框架中，发布者和订阅者是客户端，注册服务器是服务器端，客户端通过对象引用使用服务器端的对象实现，即注册匹配服务。关于注册匹配服务的实现将在后面小节中详细说明。

作为客户端的传感器代理和作战子网网关，首先要获取注册匹配服务的对象引用，然后通过 ORB 核心向注册服务器调用注册匹配服务，服务器将注册匹配服务的结果－订阅者信息/发布者信息返回给发布者/订阅者。图 4-2 是注册服务

CORBA 方式的实现过程。

发布者和订阅者的对象引用是通过发布者程序和订阅者程序的配置文件添加相关配置命令行的方式指定：

DCPSInfoRepo=corbaloc::192.168.1.10:12345/DCPSInfoRepo

“corbaloc”表示是 CORBA 可解析的通信地址类型，“192.168.1.10:12345”代表服务器服务请求的监听地址端口，“DCPSInfoRepo”代表的是服务实现的名称。

上面命令行是告诉传感器代理和作战子网网关，在地址为 192.168.1.10 的主机提供注册匹配服务，名称是 DCPSInfoRepo，接入地址端口是 192.168.1.10:12345。

对上述的对象调用过程，对象调用参数如下所示：

传感器代理的对象调用参数：

Publisher:

Socket_Adress

Publisher_ID

Publisher_qos

Topic(发布条件):

Topic_name

Topic_qos

Datawriter:

Datawriter_ID

Datawriter_qos

作战子网网关的对象调用参数：

Subscriber:

Socket_Adress

Subscriber_ID

Subscriber_qos

Topic(订阅条件):

Topic_name

Topic_qos

Datareader:

Datareader_ID

Datareader_qos

注册服务器将作战子网网关的对象调用参数返回给传感器代理，这样传感器代理就知道了符合其发布条件的作战子网网关的数据请求和通信地址等信息，可以主动与之建立通信连接，向其发布数据。

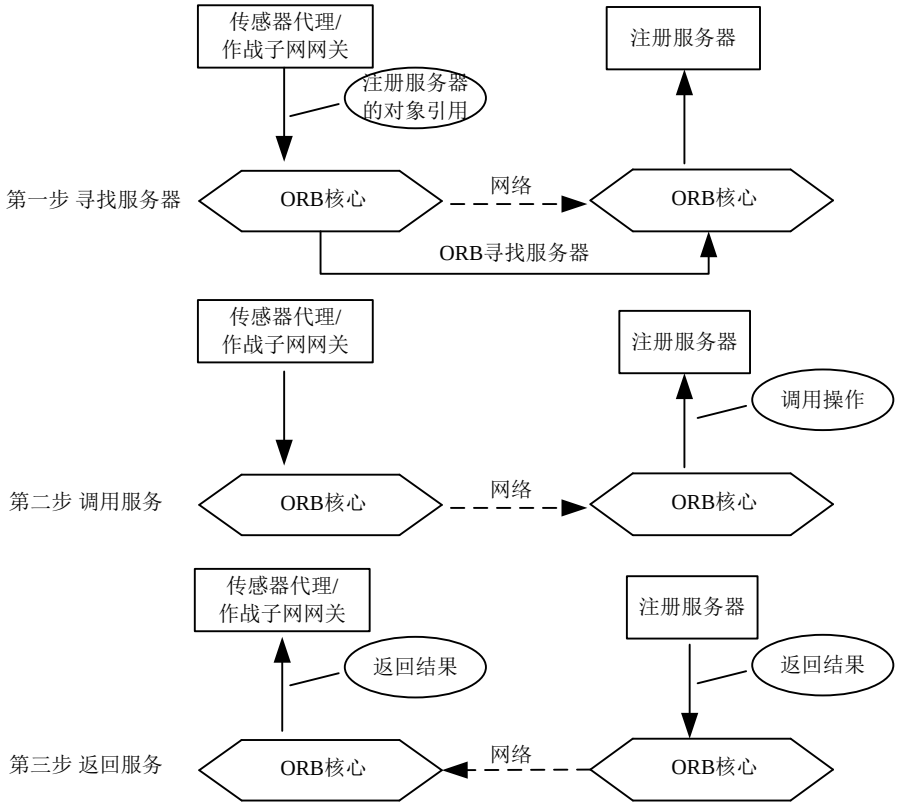

图 4-2 注册服务 CORBA 方式的实现过程

三、注册更新

由于发布者和订阅者动态地加入或退出，需要注册服务器对注册信息定期更新。更新分为主动更新和被动更新。主动更新是指发布者的发布条件和订阅者的订阅条件发生变化，主动的向注册服务器更新注册信息。被动更新是指发布者和订阅者由于通信链路中断等异常情况致使丧失主动更新的能力，由注册服务器按照预先设定的规则自行更新发布者和订阅者的注册信息。

被动更新通过监测心跳信号的方式实现。发布者和订阅者周期向注册服务器发送心跳信号，如果注册服务器在规定的时限内一直未能收到心跳，则认为发布者或订阅者出现异常，需要注销，删除相关的注册信息，直到再次接收到新的注册请求。心跳信号通过 DDS 规范中的 Liveness Qos 实现，在创建 Datawriter 和 Datareader 时设置，相关代码如下：

dw_qos.liveliness.kind = AUTOMATIC; //自动发送心跳信号

dw_qos.liveliness.lease_duration.sec = LEASE_DURATION_SEC ;

dw_qos.liveliness.lease_duration.nanosec = 0 ;

Datawriter 和 Datareader 的心跳信号采用周期自动发送的方式，心跳信号发送周期的设置与网络通信环境有关，通信环境好的情况下，设置较长的发送周期，反之则要设置较短的发送周期。

4.3.3 注册匹配服务

注册匹配服务在注册服务器上实现，是发布者和订阅者注册服务的对象实现，通过 ORB 软件总线接收订阅者和发布者的对象引用，然后根据对象引用调用注册匹配服务的对象实现，最后通过 ORB 软件总线将注册匹配服务的结果－订阅者信息返回给发布者。匹配算法是注册匹配服务的核心。

一、匹配算法

在注册服务的实现中，将发布条件和订阅条件映射到 Deadline、Latency_budget 和 topic_data Qos 中。注册匹配服务的匹配算法就是匹配 Datawriter 和 Datareader 的 Topic 及其 Topic Qos，匹配算法流程如图 4-3 所示。Topic 对应于发布条件中的传感器类型，topic_qos.deadline 对应于数据刷新率。

在 Topic 匹配成功之后，其它项目匹配之前，需要先匹配 Datareader 和 Datawriter 的通信可靠性策略，它们附加在 Topic 的 Reliable Qos 中，因为如果 Reliable 不匹配，Datareader 和 Datawriter 就不能建立通信链接，否则其它项目的匹配都是徒劳的。

在上述的匹配过程中，订阅条件中的 Latency 和 Target_type 没有参与匹配过程，因为 Datawriter 无法给二者确定值。

在匹配算法中，区域的匹配采用了包含或部分包含策略，即当订阅者 Datareader 的目标区域包含或者部分包含在发布者 Datawriter 的能力区域范围内时，就认为区域匹配成功。

二、算法实现

本课题的数据发布采用的是主动的推模式，由发布者主动发起通信过程。在 3.4.1 节中，提出了单源选择分发、多源选择分发和多向分发三种应用业务，这三种业务的实现依赖于注册服务器的匹配策略，因此针对上述三种业务设定两种匹

配策略：ONE_MATCH_POLICY、MULTI_MATCH_POLICY和ONE_MANY_POLICY。匹配策略由订阅者在其订阅条件的 Topic Qos 中指定，使用 topic 的 Ownership Qos。

topic_qos.ownership = SHARED_OWNERSHIP_QOS;　　//多源选择分发

topic_qos.ownership = EXCLUSIVE_OWNERSHIP_QOS; //单源选择分发

在注册匹配线程中创建三个队列：Datawriter 队列、Datareader 队列 1 和 Datareader 队列 2。

1、ONE_MATCH_POLICY 策略

此匹配策略针对于单源选择分发，即 Datareader 只需一个 Datawriter 发布数据，需要同时使用上述三个队列。，Datareader 队列 1 由新注册的或者未匹配到 Datawriter 的 Datareader 组成。Datareader 队列 2 由已匹配到 Datawriter 的 Datareader 组成。

（1）新注册的 Datawriter 和 Datareader 分别插入 Datawriter 队列和 Datareader 队列 1。

（2）从 Datawriter 队列的头节点开始，按照图 4-3 的匹配算法对 Datareader 队列 1 遍历，当某个 Datareader 与之匹配时，将此 Datareader 的接口信息、订阅条件返回给 Datawriter。

(3) 检查该 Datareader 订阅条件中的 Ownership Qos，如果是 EXCLUSIVE_OWNERSHIP_QOS，将其插入 Datareader 队列 2，Datawriter 继续向后遍历直至队尾。

(4)取 Datawriter 队列中的下一成员继续上述过程。

(5)当 Datareader 队列 1 中的成员与原 Datawriter 间的通信链接失效，重新插入 Datareader 队列 1 的队首等待匹配。上述过程这样就可保证 Datareader 只有一个 Datawriter 向其分发数据。

2、MULTI_MATCH_POLICY 策略

该策略只使用 Datawriter 队列和 Datareader 队列 1。上述过程的步骤（3），如果检查到 Datareader 订阅条件中的 Ownership Qos 是 SHARED_OWNERSHIP_QOS 时，注册匹配服务就采用 MULTI_MATCH_POLICY 策略。当 Datareader 队列 1 中的成员匹配到 Datawriter 的成员时，无须出队，可以继续等待 Datawriter 队列其它节点的匹配。只要有 Datawriter 符合 Datareader 的订阅条件，就可以向其发布数据，最后由 Datareader 在本地对多源的数据按照 Destination_order Qos 进行组合排序，上传给应用程序。

上述两种匹配策略均可满足多向分发，为一个 Datawriter 匹配到多个 Datareader。

三、匹配结果返回

发布者和订阅者与注册服务器之间是通过 CORBA 方式进行交互，发布者调用注册服务器上的 DCPSInfoRepo 对象实现，DCPSInfoRepo 对象实现采用匹配算法为发布者找到合适的订阅者，将 Datareader_ID、Datareader Qos 订阅者的通信地址等信息通过 ORB 软件总线返回给 Datawriter。

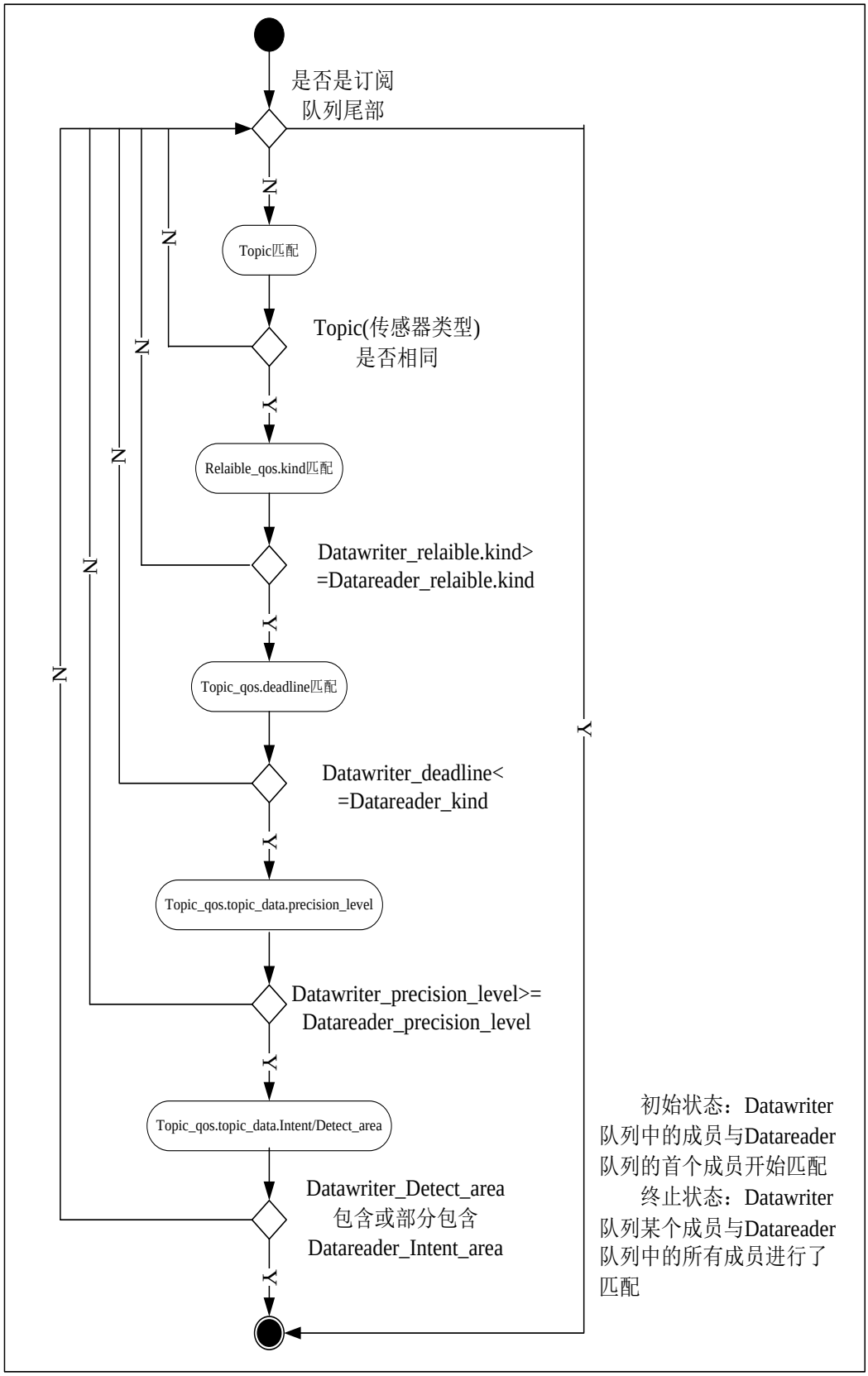

图 4-3 注册服务匹配算法

4.3.4 发布服务

发布服务和订阅服务是发布/订阅中间件层的核心功能。发布服务将应用数据按照订阅条件按需的分发给相应的订阅者，具体由 Datawriter 完成。Datawriter 的分发不仅与 Datawriter Qos、Publisher Qos 紧密相关，还与订阅条件形成的数据分发过滤服务紧密相连。数据分发过滤服务在应用层调整 Datawriter 发布的数据，Datawriter 和 Publisher 的 Qos 在发布/订阅中间件层控制 Datawriter 的数据发布过程。

图 4-4 是发布节点 A、订阅节点 B 和 C 数据分发通信连接图，具体的通信过程在发布节点的 Datawriter 和订阅节点的 Datareader 之间进行。

一、Publisher 和 Datawriter

Publisher 是 Datawriter 对象的容器，可包含多个 Datawriter 对象。Datawriter 是完成应用数据分发任务的真正实体，Publisher 根据数据的 Topic 及数据结构创建相应的 Datawriter 对象，Datawriter 在创建时根据应用需求配置 Qos，数据的分发仅需通过一个写操作调用即可完成：

retcode = writer->write(data, instance_handle);

“data”是发布对象，“instance_handle”是数据的处理句柄，关联了目的地址、Qos 控制参数等信息。

二、Qos 控制

表 2-2 详细地列举了 Topic、Subscriber 和 Datawriter 和 Datareader 的 Qos。发布服务的 write()操作会受到以下 Datawriter Qos 的控制：Resource_limits、History、Reliability、Transport_priority、Deadline 和 Latency_budget。其中 Resource_ limits 和 History 控制 Datawriter 可使用缓存资源的总量和历史数据保存的数量。Reliability 和 Transport_priority 控制通信资源的使用，为 data 指定传输层协议和发送优先级。Deadline 和 Latency_budget 为数据发布任务指定数据发送周期和时延期限。这些 Qos 参数使用默认值或由用户配置，配置参数通过配置文件读入应用程序。

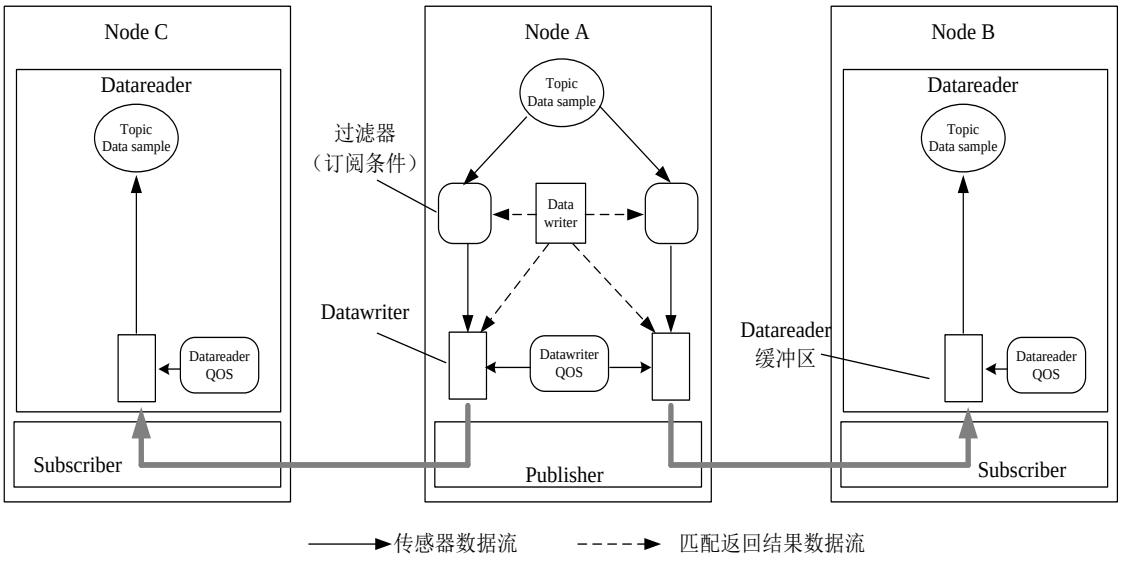

图 4-4 发布/订阅节点数据分发通信连接图

三、数据分发和过滤

基于 Topic 的发布/订阅，只要 Topic 匹配，Datawriter 就会通过上面的 write() 操作将相同的数据发布给所有匹配的 Datareader。对 Topic 进行内容扩展之后，Datareader 的 Topic 相同，但内容项是不同的，如果使用上述 write()操作发布数据，各 Datareader 会接收到相同的数据，这样的数据对每个 Datareader 都会有冗余数据存在，不符合按需分发，因此需要对现有的发布方法做出改进。具体方法如下：

1、Publisher 根据 topic 创建相应的 Datawriter，按照发布条件设置各种 Qos，向注册服务器注册，然后等待匹配结果的返回。

2、当有匹配结果返回时，Publisher 创建一个相同的 Datawriter，这个 Datawriter 不再向注册服务器注册，而是提取前面的返回结果，根据返回结果中的订阅条件构建数据分发过滤组件，对数据进行过滤。

3、新创建的 Datawriter 主动向返回结果中的 Datareader 发起通信链接，这样就可以保证 Datareader 能够接收到完全符合订阅需求的数据。

首先创建的 Datawriter 类似于并发服务器的监听 Socket，后创建的 Datawriter 类似于通信 Socket。如果新返回的匹配结果的订阅条件与之前返回结果的订阅条件相同，而且已经创建了 Datawriter，就无须创建新的 Datawriter，由之前创建的 Datawriter 以一对多的方式分发数据。图 4-4 表示了数据分发的通信连接图。

数据分发过滤组件由匹配结果的中的订阅条件构成，对数据的刷新率、目标区域、目标类型进行过滤。传感器探测到的目标类型是多种的，通过目标类型过

滤滤除掉其它类型的目标信息，这要求传感器生成的数据包含目标类型信息。目标区域的过滤于目标类型过滤相似，滤除其它区域的信息。对数据刷新率，数据链和战术互联网的数据刷新率 Rdl 和 Rti 均为 1/12，本课题使用的雷达数据模拟器的默认刷新率是 Rs 是 1/6,可以对传感器数据以 1/Rdl 或 1/Rti 的时间间隔取样，可降低一半的数据量。对数据过滤之后就可以通过 write()操作将数据发布给 Datareaders。这是实现按需分发重要步骤，从数据源降低了网络中的冗余数据。

四、数据传输

数据传输是由各个 Datawriter 将发布的数据发送给相应的 Datareader，通信链接的创建以及管理是由 Publisher 实现的。应用程序的发布者线程在初始化过程中，节点首先创建 Publisher，然后创建传输层实现，并与 Publisher 关联，这样就使得传输层成为 Publisher 管理的对象。Publisher 根据应用程序的配置参数、reliability 和 transport_priority Qos 为每个 Datawriter 创建相应的通信链接。应用数据需要可靠通信时，Publisher 为 Datawriter 关联 TCP 或 ReliableMcast，调用可靠传输服务，当要求实时通信时，Publisher 为 Datawriter 关联 UDP 或者 SimpleMcast 调用实时传输服务。当 Datawriter 有多个发布对象，且订阅条件相同，此时可采用多播传输服务实现分发，以优化网络资源使用。

4.3.5 订阅服务

订阅服务是发布/订阅中间件层另一核心功能。订阅服务使用注册服务向网络中发出数据请求，然后等待订阅数据的到达，将订阅到的数据交给上层应用。与发布者相似，订阅者的数据订阅是由 Datareader 完成。

一、Subscriber 和 Datareader

与 Publisher 相似，Subscriber 是一个 Datareader 对象的容器，可包含多个 Datareader 对象。Datareader 是完成数据订阅任务的实体对象，Datareader 接收符合其订阅条件的数据。订阅数据的接收是通过一个读操作调用完成的：

status = FC_M_dr-> read (FC_M_data);

“FC_M_dr”是 Topic FC_M 对应的 Datareader，“FC_M_data”是存放 Topic FC_M 类型数据的指针变量。

二、Qos 控制

表 2-2 已经详细地列举了 Topic、Subscriber 和 Datawriter 和 Datareader 的 Qos。数据 read()操作主要受到以下 Datareader Qos 的控制：Resource_limits、History、

Reliability、Destination_order、Reader_data_lifecycle、Time_based_filter 和 Deadline。其中 Resource_limits 和 History 控制 Datareader 可使用缓存资源的总量和历史数据保存的数量；Reliability 控制通信资源的使用；Deadline 和 Time_based_filter 控制订阅数据的接收，剔除过期的数据；Destination_order 用于多源选择分发时，控制多源数据在 Datareader 处的组合方式，是按照到达的时间顺序或者发送的时间顺序（需要网络节点时间同步）。Deadline 和 Time_based_filter 均设置为 1/Update_rate，destination_order 取值 BY_RECEPTION_TIMESTAMP。

三、订阅数据接收

订阅数据接收采用了 DDS 规范的监听订阅模型（Subscription with Listener），为每个 Datareader 创建一个 DatareaderListener，负责监听 Datareader 是否有数据到达。如果有数据到达，触发 DatareaderListener 的 on_data_available()操作通知应用程序有数据到达，然后调用 Datareader 的 read()操作从 Datareader 缓存读取订阅数据。至此 Datareader 获得了感兴趣的数据，IP 骨干网中的按需分发至此结束。最后需要通过子网请求管理分发服务实现传感器数据分发的最后一个步骤－Datareader 将订阅到的数据分发给子网内部的数据请求节点。

4.3.6 子网请求管理分发服务

子网请求管理分发服务汇聚优化子网用户终端的数据请求，生成订阅条件。当订阅者的 Datareader 将订阅到的数据向向子网内的用户终端分发时，与订阅条件相关的数据请求提供分发依据。用户终端汇聚优化能够避免重复订阅，浪费带宽资源，这是实现按需分发、降低带宽资源占用的重要措施。

一、数据请求结构定义

子网请求管理分发服务呈现给订阅者的是一个个相互独立的数据请求条目，订阅者将为每个数据请求条目生成独立的 Datareader，数据请求条目就是该 Datareader 的订阅条件。本文将子网用户终端的数据请求项设置成和订阅条件相同的数据项，数据结构如下所示：

```
struct Node_requestion
{
string          Node_adress;    //节点地址，战术互联网和数据链均转换成字符串
SENSOR_TYPE    Sensor_type;       //传感器类型
```

```
AREA            Intent_area;        //目标区域
FLOAT           Update_rate;        //数据刷新率要求（实时性要求）
u_short         Precision_level; //数据精度等级：态势级、战术级、武器级
u_short         Target_type;        //运输机、战斗机、轰炸机
}NODE_REQUESTION;
```

数据请求条目的数据结构与订阅条件相同，是由多个 NODE_REQUESTION 汇聚优化之后形成的，数据结构如下所示：

```
struct SubNet_requestion
{
u_short         Item_No;
SENSOR_TYPE    Sensor_type;
AREA            Intent_area;
FLOAT           Update_rate;
u_short         Precision_level;
u_short         Target_type;
}SUBNET_REQUESTION;
```

SUBNET_REQUESTION 结构映射为订阅条件，其中 Sensor_type 将映射成 Topic，Update_rate 映射为 topic_qos.deadline，其它封装在 topic_qos.topic_data Qos 中。一个 SUBNET_REQUESTION 与多个 NODE_REQUESTION 关联，SUBNET_REQUESTION 映射成订阅条件订阅数据，Datareader 将依据与之关联的 NODE_REQUESTION 过滤订阅到的数据并分发给子发出数据请求的用户终端。下面举例说明子网用户终端数据请求的汇聚优化过程。

二、数据请求汇聚优化算法

现假设有两个节点提出数据请求 Requestion1 和 Requestion2，需求汇聚依次按照传感器类型、目标区域、数据精度等级、数据刷新率和目标类型的顺序进行。这里使用 C 语言风格的自然语言描述汇聚优化过程：

```
NODE_REQUESTION       Requestion1,Requestion2;
SUBNET_REQUESTION     Item1,Item2,Item_combination;


Item1=Requestion1;
Item2=Requestion2;
//传感器类型汇聚，如果类型不同，生成两个子网请求 Item1 和 Item2
```

```
if (Requestion1.Sensor_type == Requestion2.Sensor_type)
    Item_combination.Sensor_type=Requestion1.Sensor_type;
else
    return FAILED;
//目标区域汇聚，如果部分包含或者不相关，则汇聚失败，生成两个子网请求 Item1 和 Item2
if (Requestion1.Intent_area 包含 Requestion2.Intent_area)
    Item_combination.Intent_area=Requestion1.Intent_area;
else if (Requestion2.Intent_area 包含 Requestion1.Intent_area)
    Item_combination.Intent_area=Requestion2.Intent_area;
else
    return FAILED;
//数据精度等级汇聚，取精度等级高的作为汇聚后的精度等级
if (Requestion1.Precision_level>=Requestion2.Precision_level)
    Item_combination.Precision_level = Requestion1.Precision_level;
else
    Item_combination.Precision_level = Requestion2.Precision_level;
//数据刷新率汇聚，取精刷新率高的作为汇聚后的精度等级
if (Requestion1.Update_rate>=Requestion2.Update_rate)
    Item_combination.Update_rate=Item1.Update_rate;
else
    Item_combination.Update_rate= Item2.Update_rate;
//目标类型汇聚，取并集
Item_combination.Target_type = Requestion1.Target_type | Requestion2.Target_type;
//汇聚成功，Item_combination 作为 Item1 和 Item2 的汇聚结果
return  SUCCEEDED;
```

汇聚之后的数据请求条目继续保留相关联的 NODE_REQUESTION，这是 Datareader 在进行子网数据分发时，过滤订阅数据和分发目的地的依据。

经过上述汇聚优化过程，子网内节点的数据请求将被汇聚成为尽量少的订阅条件，这意味着订阅者订阅的数据的冗余度是最低的。这是本课题实现按需分发，降低网络带宽资源占用的重要方面。对每个数据请求条目，对应着独立的订阅条

件，独立的 Datareader 用于数据订阅。

三、子网数据分发

每个 Datareader 都有一个 SUBNET_REQUESTION 与之对应，而 SUBNET_REQUESTION 又由一到多个 NODE_REQUESTION 汇聚优化生成，NODE_REQUESTION 中包含了子网用户终端的地址和数据请求，因此，Datareader 首先会将订阅到的数据根据请求过滤，然后使用消息映射组件映射为子网的消息格式，最后通过子网通信协议栈分发给用户终端。至此传感器数据的整个按需分发过程结束。

4.3.7 时间同步服务

时间同步服务采用广泛应用的网络时间协议 NTP(the Network Time Protocol)[40]实现，该协议采用 Client/Server 架构，通过 UDP 传输协议消息。结合 3.5.6 节为传感器数据分发系统设计的时间同步结构，注册服务器作为 NTP 协议的服务器端，作战子网网关、传感器代理和传感器作为 NTP 协议的客户端。

本课题使用 Windows XP 系统提供的 NTP 时间服务实现传感器数据分发系统的时间同步服务。将注册服务器运行主机设置为 NTPServer，使用主板上的 CMOS 硬件时钟作为系统时钟源。将系统中其它主机配置为 NTPClient，跟随 NTPServer 的时钟源。具体的配置和启动方法参看文献[50]。NTP 时间同步服务占用 UDP 123 号端口。

4.4 注册服务器实现

注册服务器使用注册匹配服务实现传感器代理和作战子网网关的匹配。将注册服务器、传感器代理和作战子网网关三者放在 SOA 框架中，传感器代理/作战子网网关是注册匹配服务的请求者，注册服务器是注册匹配服务的提供者。注册服务器与其它二者采用 CORBA 技术进行交互，交互的过程如图 4-2 所示。在 CORBA 中，注册匹配服务是注册服务器上的一个对象实现，传感器代理/作战子网网关通过对象引用向注册服务器请求匹配结果。本课题将注册匹配服务对象命名为 DCPSInfoRepo，对象引用名为 DCPSInfoRepo。下面是注册服务器程序编程的主要步骤[49]及运行方法。

一、InfoRepo 对象定义

```
class    DCPSInfoRepo : public virtual POA_OpenDDS::DCPS::DCPSInfo
{
public:
DCPSInfoRepo(void);
virtual ~ DCPSInfoRepo (void);
bool add_topic();
bool add_publication();      //发布者注册，包含 Datawriter
bool add_subscription();   //订阅者注册，包含 Datareader
bool match_sub_pub();        //匹配操作并给发布者返回匹配结果
……
};
```

二、实现 InfoRepo 对象的操作：实现 InfoRepo 对象中的操作，主要操作如上面的定义。add_topic()、add_publication()和 add_subscription()分别实现 topic、publisher 及其 Datawriter、subscriber 及其 Datareader 的注册，match_sub_pub()实现 publisher 和 subscriber 的匹配，这是匹配服务的算法的具体实现，具体算法参看 4.3.3，该操作通过 ORB 将匹配结果返回给发布者。

三、创建 ORB：ORB 是注册服务器与传感器代理/作战子网网关的软件总线，注册服务器与传感器代理/作战子网网关通过 ORB 实现注册匹配服务的对象引用和匹配结果的返回。

四、为 InfoRepo 创建 POA：POA(Portable Object Adapter)将传感器代理/作战子网的对象请求和注册匹配服务实现 InfoRepo_servant 关联在一起。

五、创建并激活仆从[49]InfoRepo_servant：InfoRepo_servant 是 InfoRepo 对象的实现，执行传感器代理和作战子网对象引用中封装的的操作并返回结果。

六、激活 POA Manager，接收传感器代理/作战子网网关的对象引用请求。

七、程序运行：

DCPSInfoRepo -ORBSvcConf tcp.conf -ORBEndpoint iiop://localhost:12345 -d 100

-ORBSvcConf tcp.conf 配置传输层协议，这里指定 TCP 协议作为传感器代理与作战子网网关/注册服务器间的传输层协议。

-ORBEndpoint iiop:// localhost:12345 指定注册服务器监听注册信息的地址和端口。

-d 100 指定注册服务器工作的域。

至此，注册服务器已经启动，可以接收传感器代理/作战子网网关的注册，并提供注册匹配服务。

4.5 传感器代理实现

传感器代理是传感器数据的发布者，接收传感器发送的数据，将数据发布给数据的订阅者－作战子网网关。发布/订阅中间件使用 OpenDDS，传感器代理是在此中间件软件实现之上构建的，使用了发布服务和诸多与应用相关的服务组件实现传感器数据发布。

传感器代理软件的主要线程有：主线程、注册线程、数据发布线程和雷达数据接收线程。线程间采用消息队列的方式进行通信。传感器代理软件使用了注册服务、发布服务、通信服务、传感器数据标准化组件、分发过滤组件。图 4-5 使用交互活动图的方式说明了传感器代理软件各个线程的工作流程和交互过程，标注了上述服务或组件的调用点，虚线上的注释列出了线程间交互数据的数据结构。

一、主线程：使用节点配置服务读取配置文件，设置节点通信端口，输入传感器属性信息。在应用程序运行过程中，监视程序的运行状态，处理异常，提供人机交互接口。在所有的发布任务终止时，销毁所有的线程，回收内存，结束程序。主线程使用 ACE 的 ACE_Task 类[42]创建发布线程、注册线程和雷达数据接收线程等其它线程。

```
class Publish : public ACE_Task <ACE_MT_SYNCH>;
class Regist : public ACE_Task <ACE_MT_SYNCH>;
class SensorDataReceive: public ACE_Task <ACE_MT_SYNCH>;
……
```

二、发布线程 Publish：使用发布服务实现传感器数据的分发。发布线程首先创建 participant 实体和 Publisher 实体，然后 Publisher 实体根据注册线程生成的发布条件创建 Topic，设置 Topic 的 Qos 属性，所用到的 Qos 包括 resource_limits、history、deadline、topic_data、latency_budget 等。Topic 创建和 Qos 设置的代码如下：

```
// Create the publisher
DDS::Publisher_var pub =participant->create_publisher(
PUBLISHER_QOS_DEFAULT,::DDS::PublisherListener::_nil());
// Set the Qos of Topic
```

```
DDS::TopicQos topic_qos;
participant->get_default_topic_qos(topic_qos);
topic_qos.resource_limits.max_samples_per_instance =
        max_samples_per_instance ;
topic_qos.history.depth = history_depth;
topic_qos.topic_data = topic_content_data;
topic_qos.deadline = 1/Update_rate;
topic_qos.latency_budget = latency;
// Create the topic
topic_FC_M = participant->create_topic ( FC_MEASUREMENT_TOPIC,
                        FC_MEASUREMENT_TYPE,
                        topic_qos,
                        DDS::TopicListener::_nil());
```

在完成 Topic 创建及其 Qos 设置之后，创建与 Topic 对应的 Datawriter，并设置 Datawriter 的 Qos。Datawriter 是完成数据发布功能的具体实体。Datawriter 创建和 Qos 设置的代码如下：

```
// Set the Qos of Datawriter
dw_qos.history.depth = history_depth    ;
dw_qos.resource_limits.max_samples_per_instance =
max_samples_per_instance ;
dw_qos.liveliness.lease_duration.sec = LEASE_DURATION_SEC ;
dw_qos.liveliness.lease_duration.nanosec = 0 ;
// Create the Datawriters
DDS::Datawriter_var FC_M_base_dw;
FC_M_base_dw = pub_->create_Datawriter(topic_.in (),
                dw_qos, DDS::DatawriterListener::_nil());
```

至此数据发布线程生成了所有需要向注册服务器进行注册的实体，这些实体通过注册线程向注册服务器注册。Publisher 包含发布者的端口信息，Topic 和 Qos 组成了发布条件，Datawriter 代表了数据发布实体。发布线程阻塞等待注册服务返回的匹配结果。注册服务器返回的匹配结构包含符合发布条件的 Subscriber、DateReader 实体。Datawriter 主动向符合发布条件的 Datareader 发起通信，将传感器数据接收线程通过消息队列发送过来的标准化传感器数据使用数据分发过滤组

件（返回的订阅条件所形成）进行过滤，最后发布数据。所有的分发任务结束后，发布线程关闭通信线程，注销发布实体，回收内存。

注册线程 Regist：以 CORBA 方式与注册服务器通信，使用注册服务向注册服务器注册本地的 Publisher、Datawriter 和 Topic。注册线程使用发布条件生成组件将配置文件中传感器类型、能力范围、数据刷新率和精度等级等传感器属性信息映射成发布条件，然后通过线程间的消息队列将发布条件传给发布线程，依据发布条件创建 Topic 和 Datawriter。当 Topic 和 Datawriter 在发布线程创建完成之后，注册线程将 Publisher、Topic 和 Datawriter 向注册服务器注册，然后注册线程阻塞，等待注册服务器的匹配结果。注册线程将注册服务器返回的匹配结果解析之后通过消息队列发送给发布线程，由相应的 Datawriter 向返回结果中的 Datareader 发起通信。当发布过程结束之后，注册线程负责向注册服务器注销 Publisher、Topic 和 Datawriter。

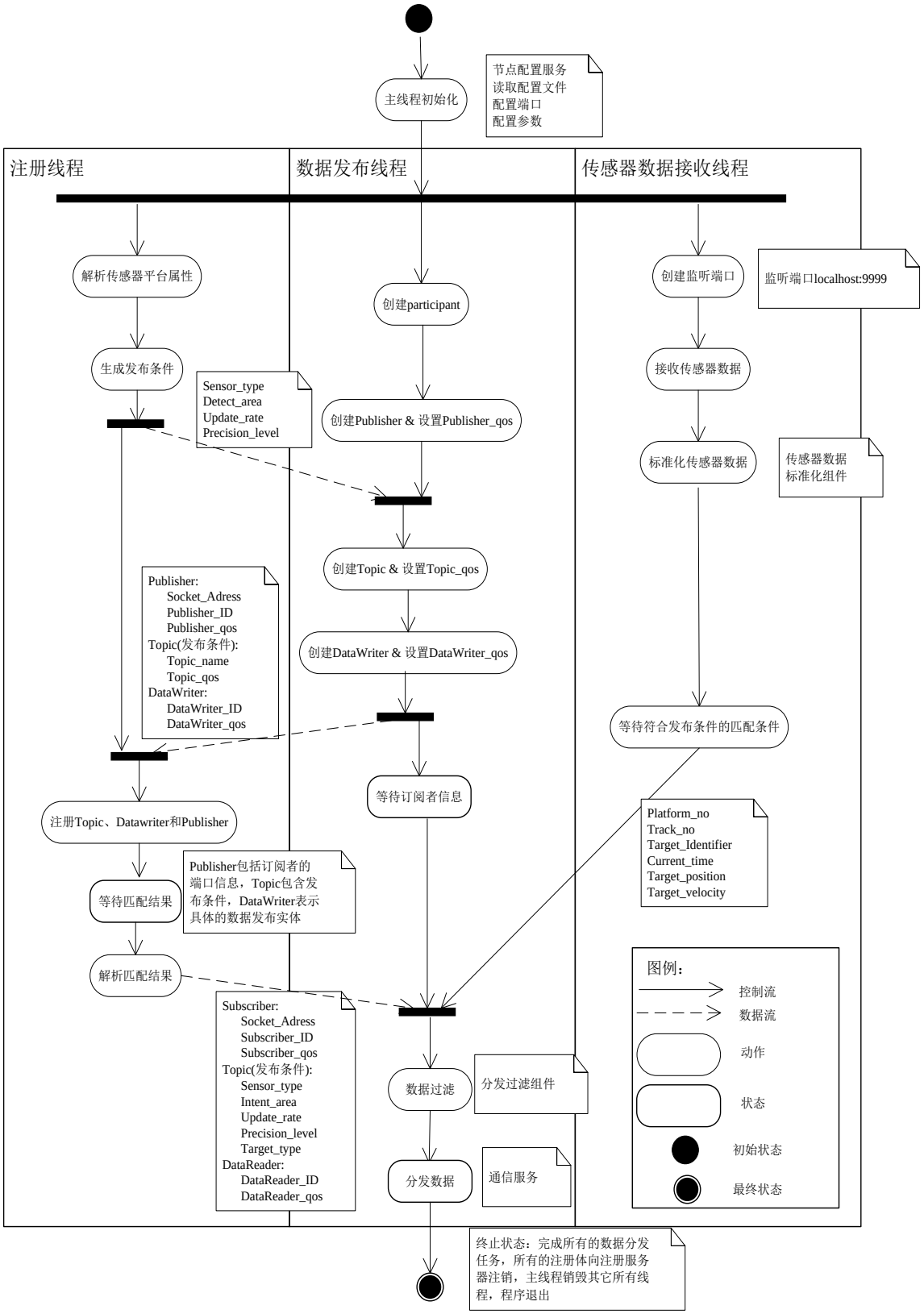

图 4-5 传感器代理线程交互活动图

三、传感器数据接收线程 SensorDataReceive：从所代理传感器接收传感器数据，为发布线程提供发布的数据对象。传感器代理部署在传感器前端，两者之间的通信条件较好，因此采用以太网接口作为通信接口，以 UDP 方式实现数据通信。传感器数据接收线程使用 ACE_SOCK_DGRAM 类提供基于 UDP（用户数据报协议）的无连接数据通信服务。线程通过 socket 端口 localhost:9999 监听传感器发送的数据。使用传感器数据标准化组件将传感器数据转换为标准的消息格式，最后通过消息队列将将数据送给发布线程。

四、传感器代理程序的运行：

传感器代理程序在启动时以配置文件的方式输入各类参数，程序启动的命令行如下所示：

publisher -DCPSConfigFile pub_conf.ini

配置文件 pub_tcp_conf.ini 为应用程序初始化提供配置参数，由以下四部分组成：

1、公共配置参数：包括程序的调试模式、传感器数据监听端口、网络侧通信端口。

2、注册服务器配置参数：指定注册服务器的地址和服务对象引用名称，参数格式如下，参数含义参看 4.3.2 节注册方式部分的相关说明。

3、传输层配置参数：配置传输协议及相关的参数。现以 TCP 为例说明传输层的配置。

4、传感器属性配置：所代理的传感器的基本属性。

```
//公共配置参数
DCPSDebug_level＝0
Internal_address＝192.168.0.1:9999
Network_address＝192.168.1.1
//公共配置参数
DCPSInfoRepo=corbaloc::192.168.1.10:12345/DCPSInfoRepo
//传输层配置参数
transport_type=SimpleTcp              //传输协议类型
local_address＝local_host:4444     //本地地址端口
swap_byte＝0                              //字节序
max_packet_size=2048                  //报文最大长度
……
```

```
//传感器属性配置
Sensor_type＝FC_RADAR          //传感器类型是火控雷达
Platform_no＝1                 //平台编号
Detect_area=10                 //10×10Km 矩形范围
Update_rate=1/6                //数据刷新率
Precision_level=1              //数据精度等级
Latency=6                      //数据时延
```

4.6 作战子网网关实现

作战子网网关是传感器数据的订阅者，代理子网内的用户终端向网络中订阅数据。作战子网网关和传感器代理的软件结构基本相似，只是在功能上不同。作战子网网关使用注册服务向 IP 骨干网中订阅数据，使用订阅服务接收符合订阅条件的数据，然后使用消息映射组件将数据映射为子网专用的消息格式，分发给子网内部的需求节点。作战子网网关软件也构建在 OpenDDS 中之上，使用了订阅服务和诸多与应用相关的服务组件实现传感器数据订阅。

作战子网网关软件的主要线程有：主线程、注册线程、订阅线程和子网数据分发线程。线程间通信同样采用消息队列的的方式。软件用到了注册服务、订阅服务、传输服务、子网请求管理组件、消息映射组件等。图 4-6 使用交互活动图的方式说明了作战子网网关软件各个线程的工作流程和交互过程，标注了上述服务或组件的调用点，虚线上的注释列出了线程间交互数据的数据结构。

一、主线程：使用节点配置组件读取配置文件，设置节点通信端口，输入订阅参数，订阅参数的数据项与订阅条件的数据项相同。主线程的监控和线程管理功能和传感器代理程序中的主线程相同，在此不再赘述。相关的代码可参照传感器代理程序代码。

二、订阅线程：主要是使用订阅服务实现传感器数据的订阅。发布线程首先创建 participant 实体和 Subscriber 实体，然后根据注册线程生成的订阅条件创建设置 Topic，如订阅服务实现中所提到的那样，Topic 的 Qos 属性包括 resource_limits、history、deadline、topic_data 等。

在完成 Topic 创建及其 Qos 设置之后，创建与 Topic 对应的 Datareader，并设置 Datareader 的 Qos。Datareader 是完成数据订阅功能的具体实体。与传感器代理不同，作战子网网关可能需要设置多个 Datareader 对应不同的数据请求。Datareader

创建和 Qos 设置的代码如下：

```
// Create the subscriber
DDS:: Subscriber_var pub =participant->create_subscriber (
SUBSCRIBER_QOS_DEFAULT,DDS:: SubscriberListener::_nil());
// Set the Qos of Topic
DDS::TopicQos topic_qos;
participant->get_default_topic_qos(topic_qos);
topic_qos.resource_limits.max_samples_per_instance =
max_samples_per_instance ;
topic_qos.history.depth = history_depth;
topic_qos.topic_data = topic_content_data;
topic_qos.deadline = 1/Update_rate;
// Create the topic
topic_FC_M = participant->create_topic ( FC_MEASUREMENT_TOPIC,
                                FC_MEASUREMENT_TYPE,
                                topic_qos,
                                DDS::TopicListener::_nil());
```

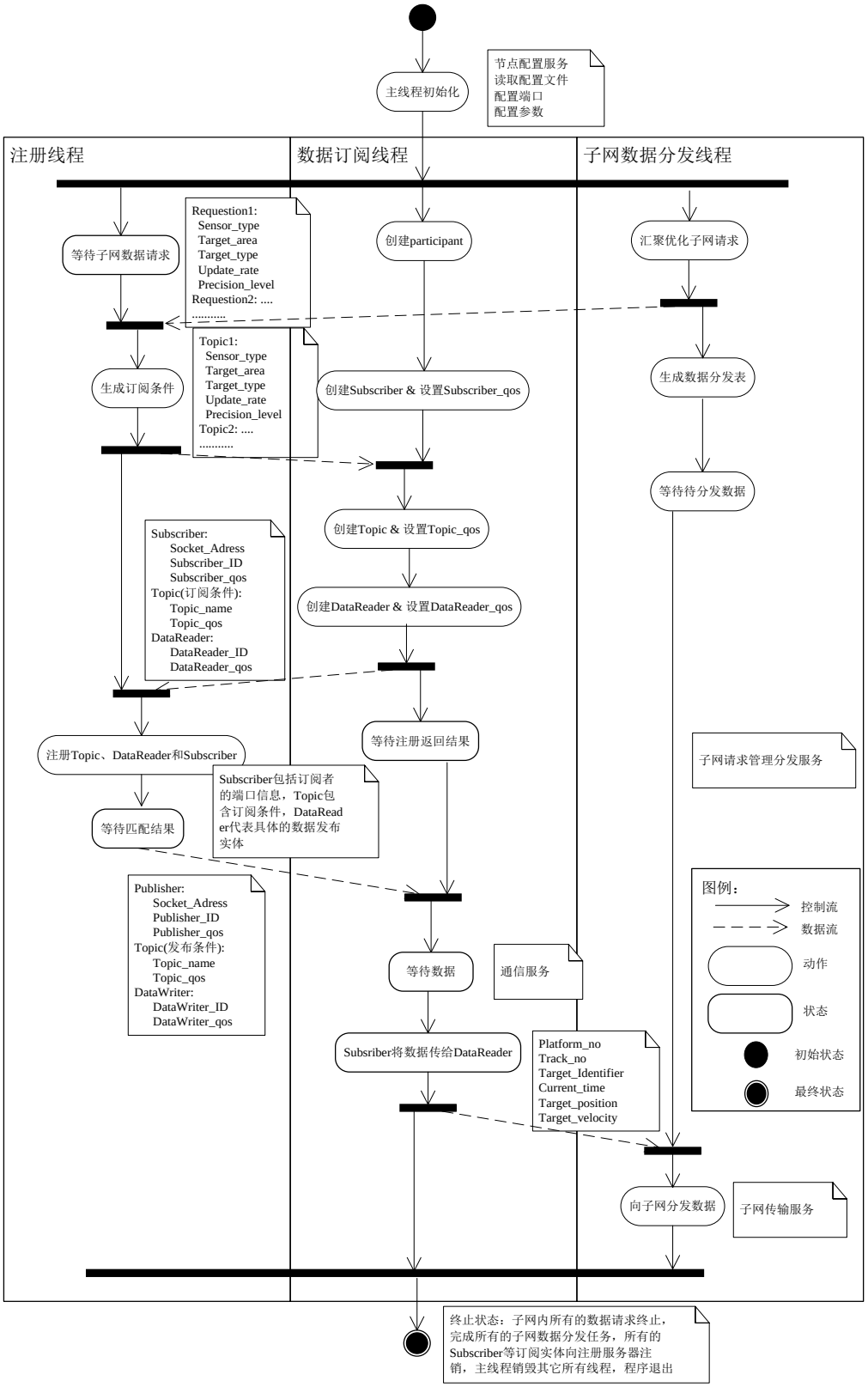

图 4-6 作战子网网关线程交互活动图

在完成 Topic 创建及其 Qos 设置之后，创建与 Topic 对应的 Datareader 并设置 Datareader 的 Qos：liveliness、reliability、destination_order。

```
// Set the Qos of Datawriter
DDS::DatareaderQos dr_qos;
sub->get_default_Datareader_qos(dr_qos);
dr_qos.history.depth = history_depth    ;
dr_qos.resource_limits.max_samples_per_instance =
max_samples_per_instance ;
dr_qos.liveliness.lease_duration.sec = LEASE_DURATION_SEC ;
dr_qos.liveliness.lease_duration.nanosec = 0 ;
dr_qos.destination_order=BY_SOURCE_TIMESTAMP;
dr_qos.reliability=BEST_EFFORT;
dr_qos.time_based_filter=1;
dr.qos.latency_budget=6;
// Create the Datawriters
pub->get_default_Datawriter_qos (dw_qos);
FC_M_dr = sub_->create_Datareader(topic_.in (),dr_qos_,
FC_M_listener.in());
```

创建完 Publisher 实体、Topic 实体和 Datawriter 实体后，至此数据发布线程生成了所有的需要进行注册的实体，可以通过注册线程向注册服务器进行注册。然后订阅者的 Datareader 阻塞可以等待发布者发布的数据， Datareader 以消息队列的方式将订阅到的数据送给子网数据分发线程，订阅线程的主要任务完成。

三、注册线程：作战子网网关的注册线程和传感器代理注册线程的功能相同，实现上大同小异，稍有差别。差别在订阅条件的生成上，这里订阅条件的生成不是直接配置的方式，而是由子网数据分发线程通过汇聚优化子网用户请求动态间接生成的，除此之外于传感器代理的注册线程相同。

四、子网数据分发线程：该线程使用子网请求管理分发服务接收子网用户终端的数据请求 NODE_REQUESTION,汇聚优化生成 SUBNET_REQUESTION。当 Datareader 订阅到数据之后，子网请求管理分发服务依据 NODE_REQUESTION 为相关联的 Datareader 提供数据分发表和过滤条件。Datareader 的数据经过滤之后再分发给发出数据请求的子网用户终端。

五、作战子网网关程序的运行：作战子网网关程序在启动时以配置文件的方

式输入各类参数，程序启动的命令行如下所示：

Subscriber -DCPSConfigFile sub_conf.ini

配置文件 sub_tcp_conf.ini 为应用程序初始化提供配置参数，组成与传感器代理的配置文件基本相同，除了数据请求部分。

```
//子网用户终端数据请求 1
Node_address=192.168.1.15        //战术互联网用户终端地址
Sensor_type=FC_RADAR               //传感器类型是火控雷达
Intent_area={(108.30,34.15),(109.30,35.15)} //目标区域
Update_rate=1/6                    //数据刷新率
Precision_level=1                  //数据精度等级
Target_type=JET
//子网用户终端数据请求 2
…….
```

第五章　数据分发系统测试

本章将模拟应用场景，利用现有资源，搭建一个测试环境，对系统系统的主要功能和性能进行测试，验证方案的可行性，暴露存在的问题。

5.1 测试环境

5.1.1 软硬件配置

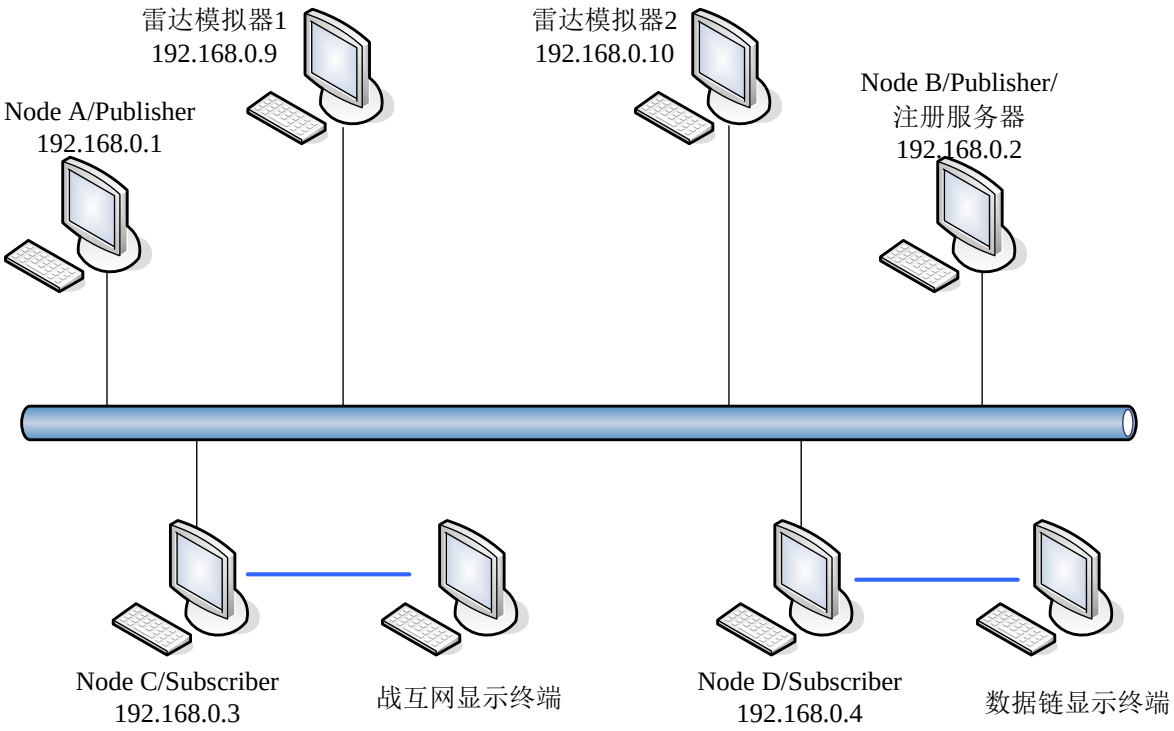

图 5-1 数据分发系统测试环境

传感器数据分发系统的测试环境如图 5-1，通过以太网将雷达模拟器、Publisher（传感器代理）、Subscriber（作战子网网关）、注册服务器（Node B）互连，战术互联网和数据链显示终端直接与两个订阅者相连。

将 Node B（注册服务器）设置为 NTPServer，其它节点均设置为 NTPClient。设置方法参看文献[50]。所有测试节点的网络接口设置为 10Mbps 全双工模式。

由于实验资源有限，Publisher 节点 A 和 B 均接收雷达模拟器 1 和 2 的数据，每个节点运行两个发布程序发布雷达模拟器 1 和 2 的数据，发布程序对应的 Topic 如表 5-1。Subscriber 节点 C 和 D，运行多个订阅程序，然后将订阅到的数据分发给战术互联网显示终端和数据链显示终端。Publisher 节点雷达模拟器数据与规划 Topic 的映射关系如表 5-1 所示：

表 5-1 Publisher 节点数据源配置表

节　点	数据源	Topic	发布程序名
Node A	雷达模拟器 1	FC_Radar_1	Pub_app1
	雷达模拟器 2	FC_Radar_2	Pub_app2
Node B	雷达模拟器 1	FC_Radar_1	Pub_app3
	雷达模拟器 2	FC_Radar_2	Pub_app4

5.1.2　测试工具

一、雷达模拟器：采用实验室自行开发的雷达模拟器，目标数据的更新周期为 5 秒。两部雷达模拟器的平台编识号分别是 50 和 58，探测区域是（108.31,34.8,500）和（105.61,34.5,500）,括弧中的的数据分别代表雷达的位置经度、位置纬度和探测半径（公里）。

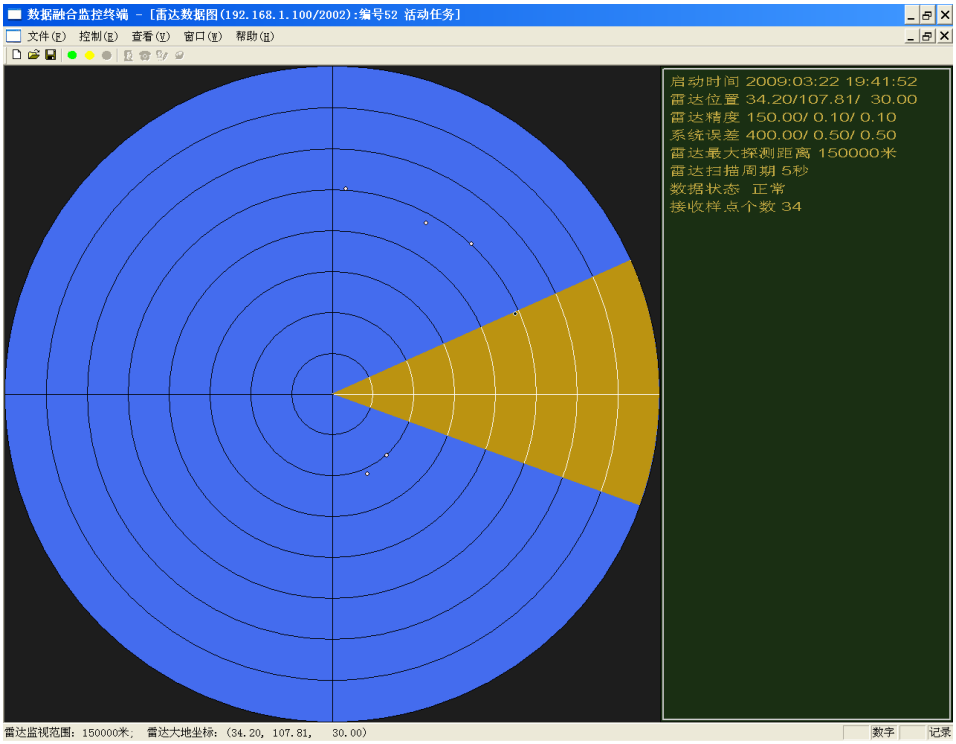

图 5-2 雷达模拟器运行界面

二、战术互联网/数据链用户终端：将封装在 VMF 消息和 J 序列消息里的传感器数据显示在终端界面（模拟终端）上，显示的数据内容有：传感器编号、航迹号、数据时间戳、目标属性和位置。

5.2 测试内容和结果

5.2.1 功能测量

一、单源选择分发

方案：Node A 运行程序 Pub_app1，Node B 运行程序 Pub_app3（Pub_app1 和 Pub_app3 发布相同的 Topic FC_Radar_1，相同内容的数据），Node C 和 D 订阅 Topic FC_Radar_1。NodeC 和 NodeD 将 Topic FC_Radar_1 的 Ownership Qos 设置为 EXCLUSIVE。

内容：Node C 和 D 只收到 Node A 或 B 发布的数据。

结果： 图 5-3 是单源选择分发条件下的数据显示，由 NodeD 将数据分发给数据链终端，由数据链终端显示。

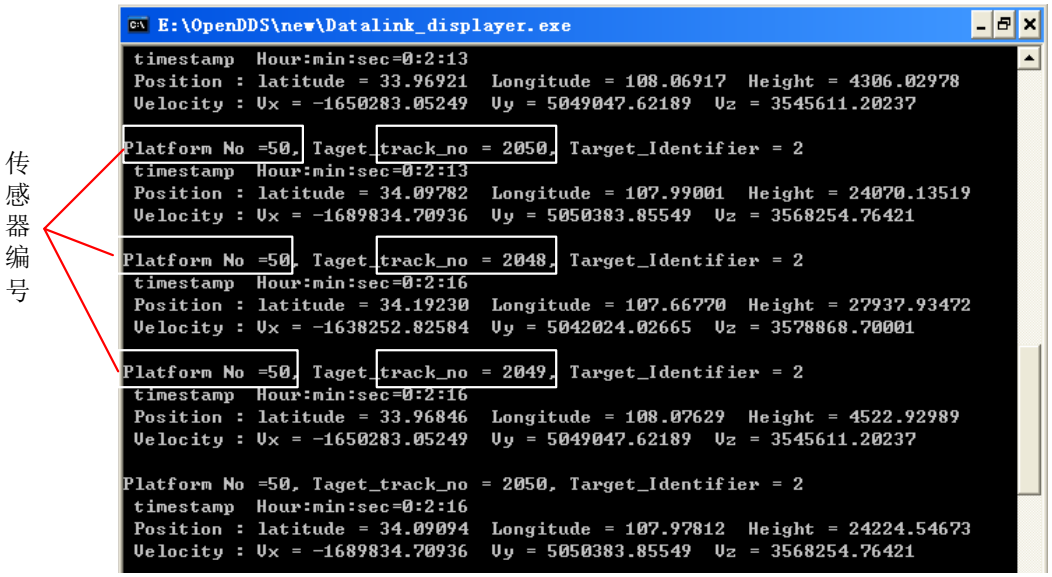

图 5-3 单源选择分发消息显示

二、多源选择分发

方案：Node A 运行程序 Pub_app2，Node B 运行程序 Pub_app4（Pub_app2 和 Pub_app4 发布相同 Topic FC_Radar_2，相同内容的数据），Node C 和 D 订阅 Topic FC_Radar_2。NodeC 和 NodeD 将 Topic FC_Radar_2 的 Ownership Qos 设置为 SHARED。

内容：Node C 和 D 能够同时收到 Node A 或 B 发布的数据。

结果：图 5-4 是多源选择分发条件下的数据显示，由 NodeC 将数据分发给战术互联网终端，由战术互联网终端显示。

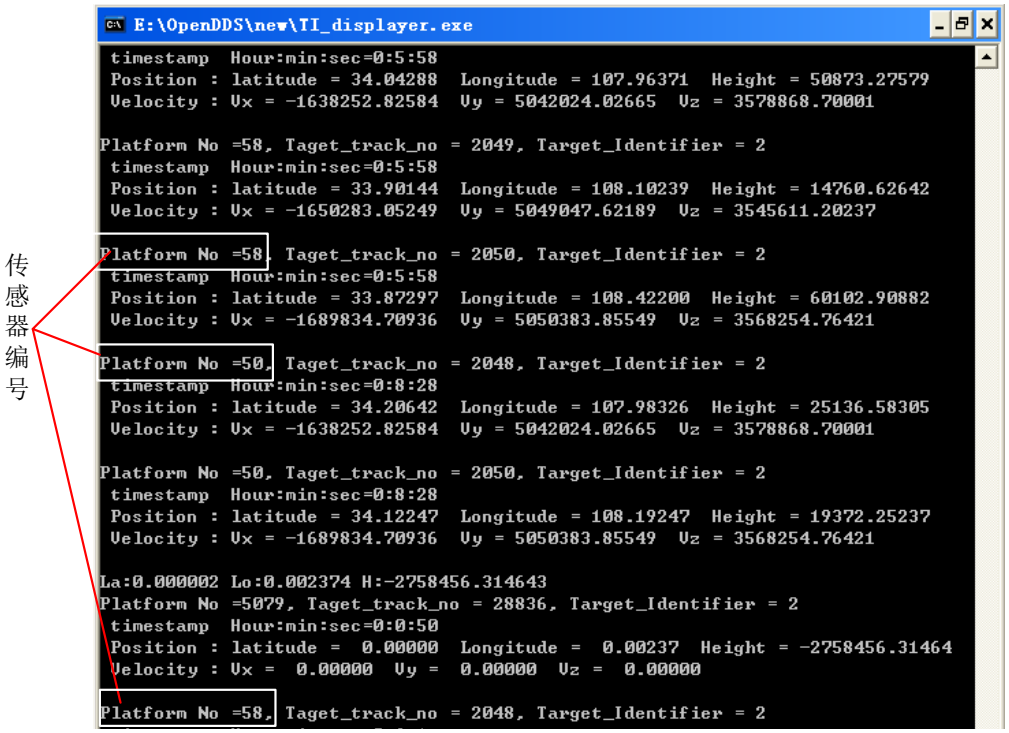

图 5-4 多源选择分发消息显示

三、多向分发

方案：Node A 运行程序 Pub_app1，Node C 和 D 订阅程序订阅 Topic FC_Radar_1。

内容：测试 Node C 和 D 能否收到 Node A 发布的数据。

结果：NodeC 和 Node D 均能收到 NodeA 发布的数据。

四、数据源自动切换

方案：Node A 运行程序 Pub_app1，Node B 运行程序 Pub_app3（Pub_app1 和 Pub_app3 发布相同 Topic，相同内容的数据），Node D 订阅 Topic FC_Radar_1。

内容：在 Node D 正常接收 Node A 数据的过程中，Node A 程序 Pub_app1 退出，Node D 改为接收 Node B 发布的数据。

结果：NodeD 能够自动将数据源从 NodeA 切换到 NodeB。

5.2.2 性能测量

一、点对点分发时延

NodeA 运行 Pub_app1，发布 Topic 为 FC_Radar_1 传感器数据，NodeC 运行程序订阅 FC_Radar_1 Topic。通过增加雷达模拟器上的目标数量逐步增加数据发布速

率，比较 UDP、TCP、ReliableMcast 和 SimpleMcast 四种传输协议在不同流量条件下的传输时延。每个数据包大小为 220 字节（数据链路层），分发时延是 NodeC 上的应用程序从 Datareader 获得数据的时刻与雷达模拟器发送数据时刻的差值。

图 5-5 是上述四种传输协议在不同流量条件下的时延情况，从测试结果来看，UDP 协议的传输时延最小，SimpleMcast 次之，RelaibleMcast 时延最大。虽然 RelaibleMcast 也是使用 UDP，但由于其需要在发送端和订阅端进行确认，增大了传输时延。通过测试结果，可以看出，选择不带确认机制的 UDP 协议作为传感器数据分发系统的传输协议，实现传感器数据的实时分发是合适的，因此在后面的测试中，主要对 UDP 协议进行测试。

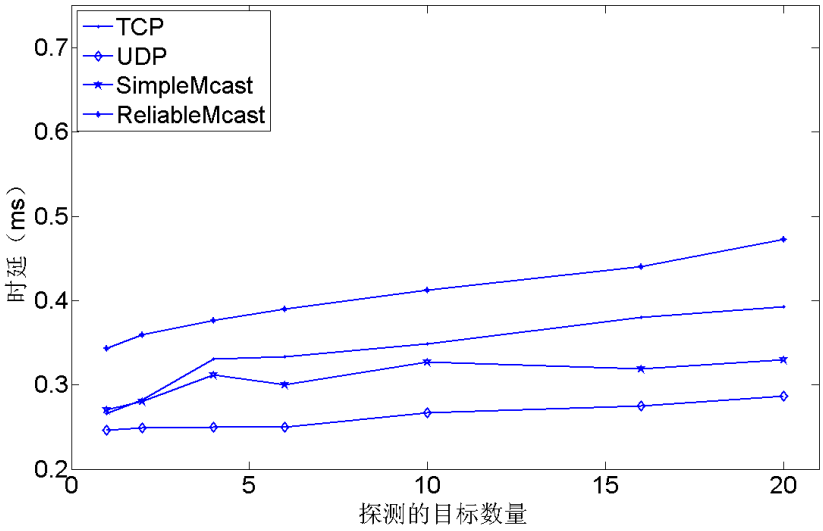

图 5-5 传输协议时延比较

二、多向（一对多）分发时延

NodeA 运行 Pub_app1，发布 Topic 为 FC_Radar_1 传感器数据，NodeC、NodeD 逐步增加订阅者程序的数量，订阅 FC_Radar_1 Topic。雷达模拟器上的目标数量固定为 20 批，测试在不同订阅者条件下的分发时延，传输协议采用 UDP。传感器数据包大小和时延计算方法与前面相同。分发时延是对每个订阅程序的测试结果取平均获得的。

图 5-6 是不同订阅者数量条件下的时延结果，从测试结果分析，在订阅者少的情况下，分发时延变化不大，但当订阅者数量大于等于 8 时，时延增加幅度明显。多向分发的时延主要产生在发布端，因为订阅者增多，发布者的负荷变得越来越大。

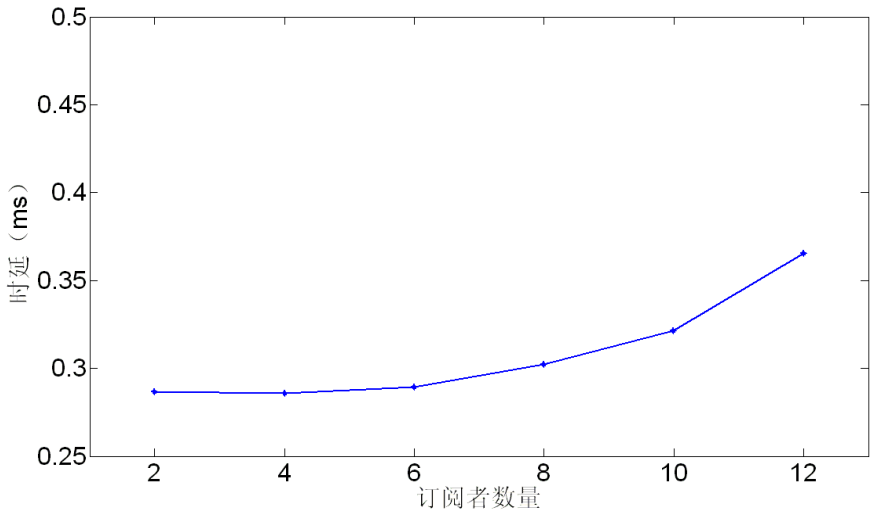

图 5-6 多向（一对多）分发时延

三、多对一分发时延

NodeD 作为订阅节点，运行订阅程序订阅 Topic FC_Radar_1，NodeA 和 NodeB 作为发布者，运行发布程序，发布 Topic 为 FC_Radar_1 传感器数据。雷达模拟器的目标数量固定为 20 批，通过逐步增加发布程序的数量测试 UDP 协议的传输时延。传感器数据包大小和时延计算方法与前面相同。

图 5-7 是不同发布者数量条件下的时延结果，从测试结果分析，当发布者程序数量大于 4 时，分发时延变化明显。多对一分发时延主要产生在订阅者本地，主要受测试平台性能以及接口速率的影响。

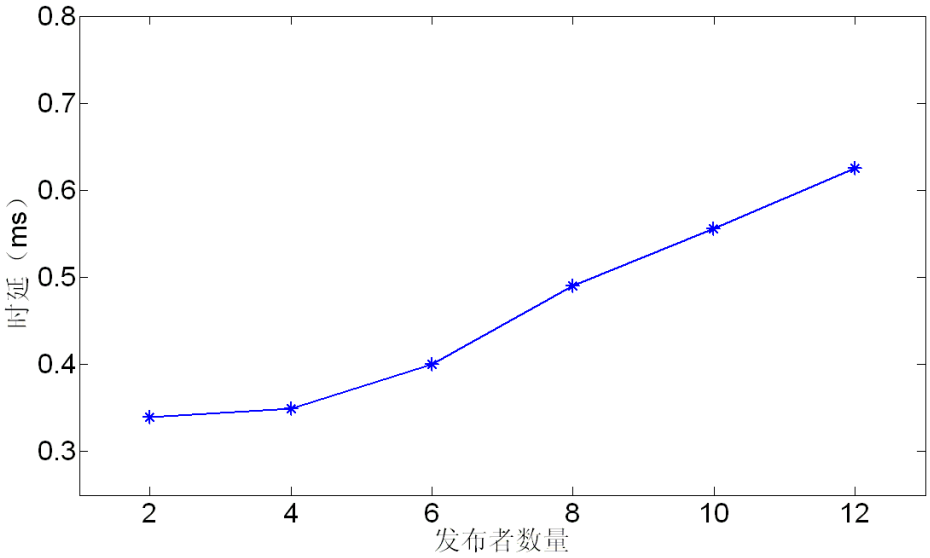

图 5-7 多源选择（多对一）分发时延

四、数据源切换时延

Node A 运行程序 Pub_app1，Node B 运行程序 Pub_app3（Pub_app1 和 Pub_app3 发布相同主题 FC_Radar_1，相同内容的数据），Node C 订阅 Topic FC_Radar_1。Node C 开始选择接收 Node A 发布的数据，然后 Node A 发布程序关闭，测试经过多长时间，Node C 可以收到 Node B 发布的数据。

经过 10 次测试，平均经过 4.45 秒的时延后，NodeC 可从 NodeB 接收到传感器数据。

第六章　结束语

6.1 论文工作总结

论文的主要工作体现在以下方面：

一、规划了符合实际情况的传感器数据分发系统应用场景，针对军事信息网带宽资源紧张的现状，提出“面向业务，按需分发”的设计思路，明确提出传感器数据实时按需分发要求。

二、将面向服务框架 SOA 作为传感器数据分发系统的设计指导思想，研究了发布/订阅机制和实时数据分发服务规范 DDS，选择发布/订阅机制作为 SOA 的实现技术。

三、针对系统按需实时分发要求，充分利用 DDS 规范面向数据的特性，灵活运用其 Qos 策略，对基于主题的发布/订阅机制进行了内容方式地扩展，实现传感器数据匹配算法，实现更小粒度的按需分发；采用非中心直连通信架构，实现数据实时分发。

四、依照“面向业务，按需分发”的思路，围绕 SOA 服务的概念，通过分析传感器数据实时分发业务流程，将传感器数据分发系统分解为注册服务、注册匹配服务、发布服务和订阅服务等服务和功能组件，定义了服务功能和接口。

6.2 下一步工作

由于时间关系，系统的实现还有一些不尽如人意的地方，需要在下一步的工作中进一步完善。

一、改造子网请求管理分发服务，使其能够接收解析来自子网用户终端的数据请求，完全实现面向子网用户终端的按需分发。

二、在注册服务匹配中，目前采用的区域定义方式和区域匹配算法需要结合实际进一步完善。

三、研究基于发布/订阅机制的传感器数据分发系统如何适应发布节点和订阅节点的移动性。

致 谢

论文完成之际，首先向我的导师朱红副教授表示最诚挚的感谢!由于是在职攻读硕士学位，工作比较繁重。在这一年里，朱红副教授对我的论文颇为关心，她知识渊博、治学严谨、思路开阔，对我悉心指导、严格要求，使每每欲陷入困惑之中的我受益匪浅。在我论文完成过程中，从论文选题、研究思路，到论文的写作均得到了朱红副教授耐心的指导和帮助，对我影响至深。

还要特别感谢给我帮助的领导付成群副教授，既是我的校外导师，也是我的领导，对我论文的完成给予了很大帮助，在这里向我的领导表示深深的感谢。

感谢学院的各位领导和同学，给予我的支持、鼓励和帮助，使我顺利完成学业。

最后由于本人研究水平有限，文中可能有不足之处，恳请老师和同学们提出宝贵意见。

参考文献

[1] 李敏. 21 世纪初我国海军海上编队“协同作战系统”的构想[J]. 舰船电子工程, 2001, 21(1):13-17.

[2] 陈康,阳东升,李恒峰,等.TCN 的海军多平台协同作战传感器管理体系结构[J].火力与指挥控制, 2008,33(4):12-15.

[3] Rand Arroyo Centre. Broadening the Army’s Bandwidth [EB/OL].(2004)[2008-06-04]. http://www.rand.org/ared/research/.

[4] RAND Corporation. Future Army Bandwidth Needs and Capabilities [EB/OL].(2004) [2008-06-04]. http://www.rand.org/ared/pubs/monographs/MG156/.

[5] P.C. Cooper. Future Army Bandwidth Needs and Capabilities, Reviewed by Lieutenant Colonel P.C. Cooper[J]. Canadian Army Journal, 2005,8(3): 127-128.

[6] Ketil Lund, Anders Eggen, Dinko Hadzic et al.Using Web Services to Realize Service Oriented Architecture in Military Communication Networks[J]. Communications Magazine, IEEE. 2007,45(10):47-53.

[7] Orton Huang, Stephen McGarry.Performance of some Service-oriented Architecture Based Systems in the Airborne Network Enviroment[C].IEEE Military Communications Conference2007, Oct. 2007:1-7.

[8] Deputy Assistant Secretary of Defense.Department of Defense Discovery Metadata Standard, Review Version 1.2 [EB/OL].(2003-07)[2008-09-15]. June 2003. http://metadata. dod.mil/mdr/ns/DDMS/2.0/.

[9] Real-Time Innovations, Inc.NDDS 4.0 Architectural Overview [EB/OL].(2005-09) [2008-09-12]. http://www.rti.com/docs/NDDS40Arch.pdf/.

[10] The Cooperative Engagement Capability. Johns Hopkins APL Technical Digest, 1995,16(4):377-396.

[11] William D. O’Neil .The Cooperative Engagement Capability (CEC), Case Studies in National Security Transformation [EB/OL].(2007-08-11) [2007-08-12]. http://www.ndu.edu/CTNSP/

[12] Solipsys Corp. Tactical component network: Overview. White paper [EB/OL].(2000) [2007-08-12].http://www.solipsys.com/tcn.php/.

[13] 单奇.应用于雷达组网的数据接入单元设计[J].测控技术, 2004,30(6):6-9

[14] 陈钢,江友谊,夏靖波. 原始雷达情报多连接传输方案[J].火力与指挥控制, 2008,33(1):56-60

[15] 孙义明,杨丽萍.信息化战争中的战术数据链[M].北京:北京邮电大学出版社,2005:39-44, 118,251,137-142.

[16] 赵文栋.宽带高速数字电台通信协议的设计与实现[D].解放军理工大学,2004.

[17] 于金华.国外战场数据链发展综述[J].无线通信技术, 2001,27(5):62-63

[18] 蔺立泳,杨万海.协同作战能力简析.火控雷达技术,2004,33(6):58-63

[19] 毛新生.SOA 原理方法实践[M].北京:电子工业出版社,2007

[20] R.Baldoni, R. Beraldi, S.Tucci Piergiovanni, et al. On the modelling of publish/subscribe communication systems[J]. Concurrency and Computation: Practice and Experience, 2005,17(12):1471 - 1495.

[21] P.Th.Eugster, P.Felber, R.Guerraoui,et al. The many faces of publish/subscribe[J]. ACM Computing Surveys (CSUR), 2005, 35(2): 114~131.

[22] Altherr, M. Erzberger, M.Maffeis. iBus-a software bus middleware for the Java platform[J]. In Proceedings of the International Workshop on Reliable Middleware Systems. 1999:43 - 53.

[23] TIB/Rendezvous. TIBCO.White paper [EB/OL].(1999)[2007-08-04]. www.tibco.com/.

[24] Talarlarian Corporation. Everything you need to know about middleware: Mission-critical interprocess communication. [EB/OL].(1999)[2007-08-04].http://www.talarian. com/.

[25] Sun Microsystems Inc. Java Message Service [EB/OL].(2002-04-12)[2008-04-14].http://www.sun.com/.

[26] P. Eugster and R. Guerraoui. Content-based publish/subscribe with structural reflection. In In 6th USENIX Conference on Object-Oriented Technologies and Systems (COOTS'01) [EB/OL].(2001)[2007-08-04]. http://se.inf.ethz.ch/people/eugster/.

[27] Object Management Group. CORBA event service specification, version 1.1. OMG Document formal/2000-03-01 [EB/OL].(2001)[2007-08-04].http://www.omg.org/.

[28] Object Management Group. CORBA notification service specification, version 1.0.1. OMG Document formal/2002-08-04 [EB/OL].(2002)[2007-08-04].http://www.omg.org/.

[29] Sun Microsystems Inc. Java message service api Rev 1.1 [EB/OL].(

2002)[2007-08-04]. http://cds-esd.sun.com/ESD4/JSCDL/jms/1.1-fr/.

[30] R. Baldoni, A. Milani, L. Querzoni, et al. DDS Report [EB/OL].(2007-02-23) [2007-05-02]. http://www.omg.org/.

[31] Object Management Group. Data Distribution Service for Real - time Systems Specification[R]. Version1.0 [EB/OL].(2004-12) [2007-05-02]. http://www.omg.org/.

[32] Object Management Group. Data Distribution Service for Real - time Systems Specification[R]. Version1.2 [EB/OL].(2007-1) [2007-05-02]. http://www.omg.org/.

[33] Thuan L. Thai 著,陈逸译. DOOM 入门[M].北京:中国电力出版社,2001.

[34] Dave Stearns,Microsoft Corp. The Basics of Programming Model Design [EB/OL].(1998-07) [2007-07-06]. http://msdn.microsoft.com/en-us/library/ms809977.aspx/.

[35] Gerardo Pardo-Castellote,Bert Farabaugh,Rick Warren.An Introduction to DDS and Data-Centric Communications [EB/OL].(2005-12-08)[2007-08-06]http://www.omg.org/news/whitepapers/Intro_To_DDS.pdf/.

[36] 李杰涛,郭敏.雷达级网中的坐标变换问题[J].火控雷达数据,2007, 36(1)::38-42

[37] 杨鹏锐,张延园,牟虹燕.数据容灾系统的原型设计及实现[J].微电子学与计算机,2008,25(5):46-50

[38] Real-Time Innovations, Inc.Meeting Real-Time Requirements in Integrated Defense Systems [EB/OL].(2007) [2007-05-02].http://www.rti.com/.

[39] Angelo CORSARO,Lenonado Querzoni,Sirio Scipion, et al. Quality of Service in Publish/Subscribe Middleware. IOS Press, April,2006.

[40] 宋妍,朱爽.基于 NTP 的网络时间服务系统的研究[J].计算机工程与应用, 2003,39(36):147-149,152.

[41] OpenDDS 1.0[EB/OL].(2007-1) [2007-05-10]. http://download.ociweb.com/OpenDDS/.

[42] Douglas C.Schmidt 著, 马维达译.ACE 自适配通信环境中文技术文档[EB/OL].(2005-1) [2007-07-09]. http://www.flyingdonkey.com/.

[43] Stephen D. Huston. James CE Johnson. Umar Syyid 著,马维达译.ACE 程序员指南-网络与系统编程的实用设计模式[M].北京:中国电力出版社,2004.

[44] ACE [EB/OL]. [2007-05-10].http://www.cs.wustl.edu/~schmidt/ACE-obtain.html.

[45] Real-time CORBA with TAO(TM)(The ACE ORB) [EB/OL]. [2007-05-10].http://www.cs.wustl.edu/~schmidt/TAO.html.

[46] Common Object Request Broker Architecture (CORBA) Specification, Version 3.1 [EB/OL].(2004-1) [2007-06-04]. http://www.omg.org/cgi-bin/doc?formal/04-03-01/.

[47] Prismtech.Inc. Splice-DDS[EB/OL]. [2008-11-04]. http://www.prismtech.com/opensplice-dds/

[48] Ming Xiong, Jeff Parsons, James Edmondson, et al. Evaluating Technology for Tactical Information Management in Net-Centric System.Defense Transformation and Net-Centric Systems 2007.

[49] 朱其亮,郑斌.CORBA 原理及应用[M].北京:邮电大学出版社,2001

[50] 如何在 Windows XP 中配置权威时间服务器[EB/OL]. [2009-03-08].http://support.microsoft.com/kb/314054/.

[51] Stanley B.Lippman, Josee Lajoie 著,潘爱民,张丽译.C++ Primer 中文版（第三版）[M].北京:中国电力出版社,2002.

[52] Andrew S. Tanenbaum 著, 潘爱民 译.计算机网络中文版（第四版）[M].北京:清华大学出版社,2004.