实时数据分发系统软件的设计与实现

杨传顺，钱幸存

（中国船舶重工集团公司 江苏自动化研究所，江苏 连云港 222006）

摘 要：实时数据分发系统指使用实时数据分发服务中间件的分布式系统。实时数据分发服务中间件采用实时发布-订阅协议，通过在数据发布方和订阅方之间配置的服务质量参数，可以实现不同的数据需求和传输方式。该中间件能够跨平台操作，屏蔽底层操作系统的差异，这有利于系统灵活动态地扩展和升级。文中阐述了数据分发服务的思想和模型，以 RTI 公司的数据分发服务中间件 DDS 为例，具体研究了该中间件的软件设计和实现。

关键词：发布-订阅协议；实时分布式系统；数据分发服务

中图分类号：TP311 **文献标识码**：A **文章编号**：1674-7720(2011)04-0003-03

Software design and implementation of real-time data distribution system

Yang Chuanshun, Qian Xingcun

(Jiangsu Automation Research Institute of CSIC, Lianyungang 222006, China)

Abstract: Real-time data distribution system refers to the distributed system which uses real-time data distribution service middleware. Real-time data distribution service middleware with real-time publish-subscribe protocol, through configuring quality of service parameters between the data publisher and subscriber, can achieve different data requirements and transmission. The middleware can be cross-platform, shielding the underlying operating system differences, which is conducive to flexible and dynamic expansion and upgrade for the system. Paper describes the ideas and model of data distribution service, takes RTI's Data Distribution Service middleware DDS for example, specific studies the software design and implementation of the middleware.

Key words: publish-subscribe protocol; real-time distributed system; data distribution service

实时数据分发系统指采用数据分发服务 DDS(Data Distribution Service)中间件的实时分布式系统。OMG 的数据分发服务正逐渐成为采用实时发布-订阅 RTPS(Real Time Publish-Subscribe)协议的分布式系统的标准，此标准的目的是提供一种通用的应用层接口，能够清楚地定义数据分发的服务，提供一种在实时分布式计算环境中以数据为中心的通信规范，定义一个可扩展、与平台无关、与位置无关的基础服务模型，以连接发布者和订阅者，支持服务质量(QoS)和属性。该通信模型在空间上(节点可以在网络中的任何地方)、时间上(发布后立即或稍微有些延迟后传输)、传输上(可根据带宽流量控制可靠性，每个用户可以管理自己数据传输的状态)、功能上(多源的数据可以很容易地添加和删除)都是松散耦合的。

1 数据分发服务

OMG 的数据分发服务(DDS)的核心是以数据为中心的发布-订阅 DCPS(Data-Centric Publish-Subscribe)，即发布者能高效地将正确的信息传递给适当的订阅者。DCPS 定义了在实时分布式系统中运行在异构平台上的标准接口的应用程序能从系统的全局数据空间读/写数据。具有以下特点：

(1)强制类型的全局数据空间

DDS 创建一个全局性的数据空间，任何参与者使用数据类型接口高效自然地读/写数据，让每个订阅者都可以访问到“最新”的数据。它还建立了一个名称空间，让参与者查找和共享数据的对象。为了提高可扩展性，主题(Topic)可以包含多个独立的数据通道称为“关键字”(key)。这使得节点可以一次性订阅到很多，可能是数以千计的相似的数据流。当数据到达时，中间件根据关键字排序，提供有效的处理。

(2)每个数据流可控的服务质量

DDS 允许控制“每个数据流”的服务质量(QoS)。每对发布者——订阅者之间可以建立独立的 QoS 协议。DDS 也具有独特的允许应用设计支持极其复杂的、灵活的数据流需求。QoS 参数几乎能控制 DDS 模型和基本的

通信机制的每个方面。

(3)自动发现和管理数据流

DDS 能设计成自动发现发布者和订阅者的每一个主题，根据它们之间设置的 QoS 参数自动建立数据流。DDS 模型提供了快速透明的定位。这使得它非常适合用于系统动态配置的变化。它能迅速发现新的节点，新的节点上的新的参与者，新参与者的新的数据主题。能及时地清除掉过时的、失败的节点和数据流。

(4)不可靠的无连接传输

DDS 以不可靠的传输方式工作，如 UDP 或无线网络。没有任何设备需要中央服务器或特殊节点。高效、直接、点对点的通信、甚至多播，可以实现模型的每一个功能。

2 DDS 的对象模型

图 1 显示了典型的 DDS 分布式应用的关键对象，域(Domain)、域成员(DomainParticipant)、数据发送器(Data writer)和发布者(publisher)、订阅者(Subscriber)和数据接收器(Data reader)、主题(Topic)组成。域成员对象指的是域内参与应用的成员，参与者可以在同一个域上进行通信。

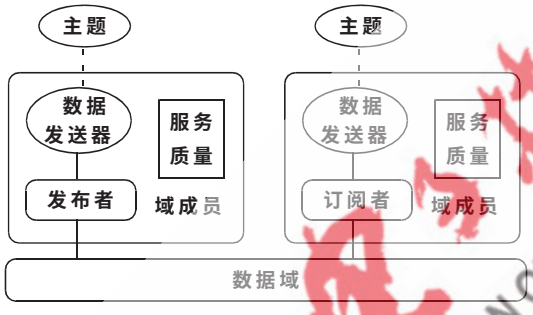

图 1 DDS 的对象模型

发布者负责数据的发布，可以发布不同类型的数据。域成员通过发布者的数据发送器交流数据和改变数据类型。一旦新的数据值传达给发布者，就由发布者决定何时执行数据的发布，实际数据发布的执行取决于发布者的服务质量。

订阅者接收发布者的数据，并提供给域成员。订阅者可接收和转发不同的数据类型。要访问接收到的数据，域成员必须通过和订阅者相关的数据接收器。

数据发送器对象与数据接收器对象通过主题联系。主题和数据类型以及数据本身的服务质量(QoS)关联。DomainParticipantFactory(域成员厂)用于创建和删除域成员，域成员通过 DomainParticipant::get_domain_id()绑定到域上，域成员可以创建和删除主题、发布者和订阅者；发布者用于强制创建和删除数据发送器先前创建的主题的类型；订阅者用于强制创建和删除数据接收器先前创建的主题的类型；DataWriter::write() 用来使应用程序更新数据值；DataReader::take()用来从 DDS 中间件中接收数据样本。DDS 的编程模型如图 2 所示。

抽象的实体接口定义了域成员和域实体(DomainEntities)(即主题、发布者、数据发送器、订阅者和数据接收

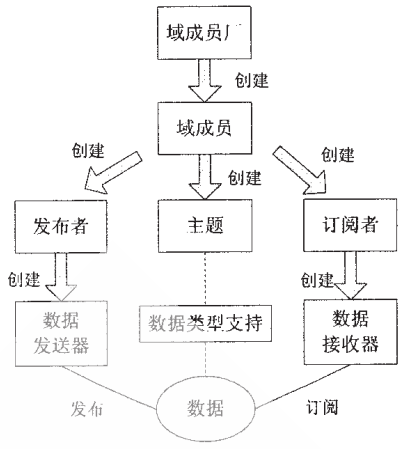

图 2 DDS 的编程模型

器)支持的通用行为。get_qos() 和 set_qos()用来检索和修改服务质量；get_listener()可被 DDS 中间件用来探测和安装用户听众的具体地位的条件。确切的服务质量参数列表是针对每个具体的实体类型。

3 应用软件的设计和实现

RTI 公司的 DDS 中间件是执行数据分发服务标准的一个典型代表。下面采用该公司的 DDS 中间件并结合 Demo 程序(发布-订阅的主题用圆形、三角形和方形的几何图形形象的表示)，研究实时数据分发系统的软件设计和实现。Demo 程序的运行界面如图 3 所示。

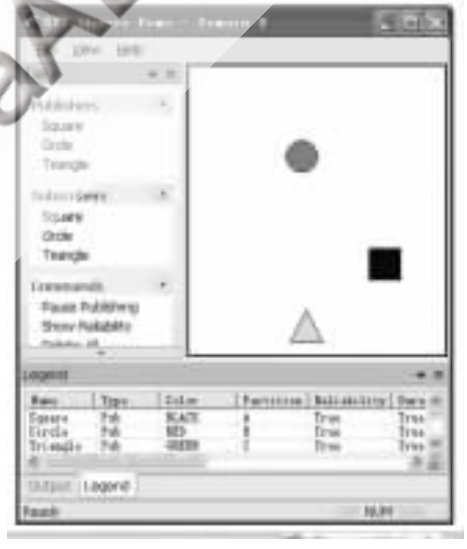

图 3 RTI 的 DDS 中间件的 Demo 程序

3.1 应用软件的设计

一般情况下，一个应用程序可以属于多个域(如图 4 所示)，创建若干域成员，三个应用程序在域 A 交流命令/控制数据，其他三个应用程序在域 B 中交流状态数据，其他两个应用程序是域 C 内通信。创建多个域，可以提供有效的数据隔离。

DDS 的主题是强制的数据类型。DDS 中间件支持多语言的绑定，在 OMG 的接口描述语言(IDL)中，定义类型 TopicInfo。TopicInfo 包含一个 graphics 变量——确定图

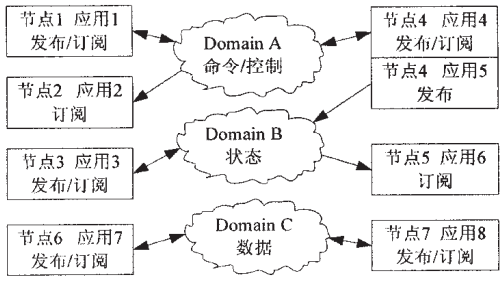

图 4 多个域提供有效的数据隔离

形的形状和 value 变量——图形的颜色。把 graphics 字段表记为关键字，让 DDS 中间件区分不同图形所更新的数据。

```
struct TopicInfo {
int graphics; //key
int value;};
```

鉴于 IDL 类型定义，DDS 的中间件提供的代码生成器用来生成特定语言代码来处理数据类型。生成的代码将在目标语言中包含类型定义（在本文中使用 C++语言）、特定标准类型的 API、中间件执行的特定代码。对于 TopicInfo 数据类型，生成下列标准类：TopicInfoTypeSupport、TopicInfoDataWriter、TopicInfoDataReader。

3.2 应用软件的实现

(1)创建 DomainParticipant 绑定到域上

```
factory = DDSDomainParticipantFactory::get_instance();
factory->get_default_participant_qos(participant_qos);
participant = factory->create_participant (domain_id, participant_qos, NULL /* participant_listener*/);
```

(2)注册用户的数据类型

TopicInfoTypeSupport::register_type()用来注册数据类型，可以在 DomainParticipant 中使用。

(3)创建主题(Topics)

使用标准的 DDS API 创建命名主题绑定到注册用户的数据类型上。主题会自动地在 DataWriters 和 DataReaders 之间建立一个数据流。

```
participant->get_default_topic_qos(topic_qos);
participant ->create_topic ("Square","TopicInfo",topic_qos, NULL/* topic_listener*/);
```

(4)创建 Publishers 和 DataWriters

在给定的 DomainParticipant 上，创建 Publisher。一个 Publishr 可以有不同主题的 DataWriters。DataWriter 提供发送数据样本的方法，管理输入数据通道的生命周期。

```
participant->get_default_publisher_qos(publisher_qos);
publisher =participant ->create_publisher ( publisher_qos, NULL /* publisher_listener */);
publisher->get_default_datawriter_qos(writer_qos);
writer = publisher->create_datawriter ( topic, writer_qos, NULL /* writer_listener */);
```

(5)发送数据更新

对于给定的 writer，使用::write()方法将新的数据值发布给 subscribers。

```
while (1) {
retcode = writer->write (topic_info, topic _instance_handle);}
```

(6)创建 Subscribers 和 DataReaders

在给定的 DomainParticipant，创建 Subscriber，一个 Subscriber 可以有不同主题的 DataReaders。通过调用 on_data_available()方法处理接收到的数据样本。

```
participant->get_default_subscriber_qos(subscriber_qos);
subscriber =participant ->create_subscriber (subscriber_qos, NULL /* subscriber_listener*/);
DDSDataReaderListener* reader_listener = new MyReaderListener();
subscriber->get_default_datareader_qos(reader_qos);
reader = subscriber->create_datareader (topic, reader_qos, reader_listener);
```

(7)接收数据样本

当应用程序检索到数据样本时，调用::on_data_available()函数接收数据；接收到 topic_info 数据时，调用::take()函数匹配 DDS 中间件里的数据状态标准。接收完数据样本后，应用程序通过调用::return_loan()函数返回。

```
void MyReaderListener::on_data_available (DDSDataReader* reader) {
TopicInfoDataReader *myReader = (TopicInfoDataReader *) reader;
retcode = myReader->take( data_seq, info_seq,
                          DDS_LENGTH_UNLIMITED,
                          DDS_ANY_SAMPLE_STATE,
                          DDS_ANY_VIEW_STATE,
                          DDS_ANY_INSTANCE_STATE);
for (int i = 0; i < data_seq.length(); ++i) {
/* display_ graphics _measurement(&data_seq[i]); */ }
myReader->return_loan(data_seq, info_seq);}
```

(8)调整服务质量(QoS)

在 DataReader 和 DataWriter 对象之间调整数据流的 QoS 参数设置，也可以通过 get_qos()/set_qos()指定参数设置。

DataReaders 可以要求不同程度的可靠性交付，从快速但不可靠到高可靠性顺序交付。这为每个数据流提供了可靠性控制。

中间件可以通过 QoS 的 OwnershipStrength 参数自动判断多个 DataWriters 的同一主题所代表的信息。根据唯一性，一个通道上的一个主题只能由单一的 DataWriter 更新。DataReaders 从 DataWriter 接收具有最高权值的数据，并提供了自动故障排除功能。如果一个高优先级的 DataWriter 失败，所有 DataReaders 立即接收较弱的备份 DataWriter 的更新数据。DataWriter 也可以指定持续时间，

通过 QoS 参数 Lifespan(预期生存期限)标示样本是否过时。DDS 可以定义许多 QoS 参数。每对相关的实体都可以在此基础上调整各自的行为。

(9)状态改变的响应

每个 DDS 实体的状态可以通过 get_*_status()函数更新。使用侦听者,当状态发生变化后中间件会自动通知或等待条件变量触发特定的状态变化。

应用程序可以对特定状态的变化做出响应。例如,应用程序可以得到通知,当 DataWriter 不再有效或其活力发生了变化;或当另一具有相同名称的主题存在,但具有不同的特征;或当 QoS 参数不符合提供的数据;或当错过了生存期限;或者当发现了一个新的与主题匹配的 DataReader(或 DataWriter);或当数据有效;或当数据样本丢失或被拒绝。应用程序响应状态变化可在每个实例的基础上调整。

OMG 的 DDS 创建了一个简单的数据通信结构,实现了复杂的数据模式。采用 DDS 中间件发送和接收数据,每个实体通过调整 QoS 参数支持复杂的数据通信模式配置,减轻了开发者担心的低层次的网络编程任务。该中间件能够跨平台操作,便于应用程序的移植和维护,易于组建和实现一个扩展的、易维护的实时分布式系统。

参考文献

[1] Middleware solutions for automation applications-case RTPS [R]. Helsinki University of Technology Information and Computer Systems in Automation Espoo 2003.

[2] RTPS wire protocol specification[S]. Version 1.0, November 2001,Real-Time Innovations.

[3] Interface for distributed automation architecture description and specification[S]. Revision 1.1,November 2002.

[4] Data distribution service for real-time systems version 1.2[S]. OMG Available Specification,January 2007.

[5] REVILL G. Designing and debugging real-time distributed systems[J]. Safety Critical Embedded Systems,2008(2):27-31.

(收稿日期:2010-08-18)

作者简介:

杨传顺,男,1978 年生,硕士,工程师,主要研究方向:计算机控制和系统工程。

钱幸存,男,1983 年生,助理工程师,主要研究方向:计算机控制和系统工程。