一种基于主题树的 DDS 发现机制的研究与实现

朱珂珂[1] 李 华[1] 唐新怀[2]

（1. 上海航天精密机械研究所，上海 201600；2. 上海交通大学电子信息与电气工程学院，上海 200240）

摘要：研究了 DDS 规范中的节点发现机制以及现有 DDS 中间件产品中不同实现机制，分析了现有的发现机制在大型分布式场景中所面临的问题，设计并实现了一种基于主题树的发布/订阅关系信息配置系统（Topic Tree Info Repo，TTIR）。通过 TTIR 对主题的实时过滤，有效降低了节点发现过程所需要发送的消息数量。同时，在 TTIR 上实现了一种分布式的主题链路锁机制，用于管理 DDS 系统中并行实施的发布/订阅关系变更，有效保证了系统在升级维护过程中的高可用性。

关键词：实时中间件；数据分发服务；全局数据空间；服务发现机制；实时发布/订阅机制

Design and Implementation of A Topic Tree based DDS Discovery Mechanism

Zhu Keke[1] Li Hua[1] Tang Xinhuai[2]

(1. Shanghai Institute of Aerospace Precision Machinery, Shanghai 201600;
2. School of Electronic, Information and Electrical Engineering, Shanghai Jiaotong University, Shanghai 200240)

Abstract: Research the participant discovery mechanism in Data Distribution Service (DDS) specification and the different implements in current DDS products, and analyze current implementations' problems in large distributed system scenario, then design and implement a topic tree based publish/subscribe information configuration repository (TTIR). By real-time filtering the topic, TTIR reduces the total number of message in discovery process. In addition, TTIR provides a topic link distributed lock, which can be used to manage the parallel publish/subscribe information changes in a DDS system. It ensures the high availability of the system in upgrade or maintenance stage.

Key words: Real-Time Middleware; Data Distribution Service; Global Data Space; Service Discovery Mechanism; Real-Time Publish/subscribe Mechanism

1 引言

随着计算机和网络通讯技术在实时应用场景中的广泛推广使用，实时系统对数据的交互性能和体验提出了越来越高的要求。OMG 针对这样以数据为中心的实时通信场景，专门制定了一套针对实时数据发布/订阅模型的规范：数据分发服务（Data Distribution Service，DDS）[1]。在 DDS 定义中，一个系统中发布出来消息是通过主题关系投递到远端的系统中的。为了保证相互通信的双方能够相互发现，DDS 规范中引入了一种简单的发现协议（Simple Discovery Protocol，SDP）。这种协议满足了中小型实时系统中的节点发现功能需求，但不能很好地支持拥有成百上千节点的大型分布式实时系统[2, 3]。本文在研究了现有规范和实现的基础上，分析了 SDP 在大型应用场景中面临的问题，设计实现了一种基于主题树的发布/订阅关系配置

作者简介：朱珂珂（1984–），工程师，计算机科学与技术专业；研究方向：信息安全、信息化技术。
收稿日期：2014-04-21

系统，降低了节点发现过程中所需要发送的消息量，并且提升了系统的可管理性。

2 DDS 发布/订阅模型及发现机制

2.1 DDS 发布/订阅模型

与传统的消息中间件（Message Oriented Middleware，MOM）有所不同，DDS 不再仅关注于对消息本身的投递，它更多的关注于用户数据。DDS 使用全局命名空间（Global Data Space，GDS）来维护数据发布者与订阅者之间的订阅关系[1, 5]。如图 1 所示，数据发布者和数据订阅者通过全局数据空间中的主题（Topic）关联在一起。为了保证实时发布/订阅参与方能够发现其他相关的参与者以及这些参与者的接入点，DDS 在节点发现模块中引入了实时发布/订阅发现协议。

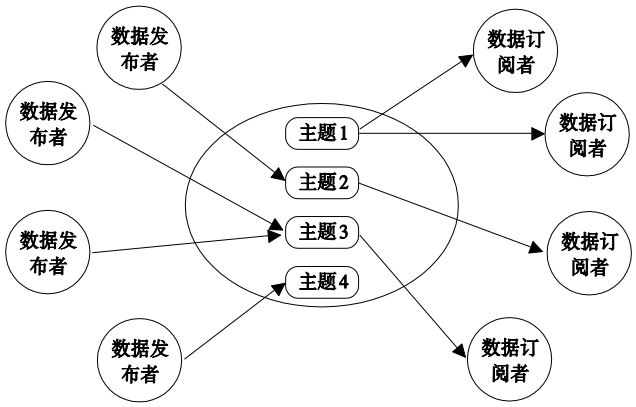

图 1 DDS 全局数据空间

通常随着业务的增长，实时系统的参与者节点会越来越多，这些节点间生成了大量的主题，形成了复杂的网状结构。这时，DDS 系统中每个节点的角色变得越来越复杂，系统需要传输的数据量越来越大。同时，不同类型的数据发布者和订阅者，往往有不同类型的业务需求和 QoS 要求。这给 DDS 系统的实时性、可用性和资源管理能力带来了巨大的挑战。本文设计实现了一种树状的主题组织和管理机制，基于树形拓扑对 DDS 系统进行子系统划分，并基于对各个子系统的 QoS 监测，进行自适应调整主题树，达到减化系统结构和优化消息链路的目标。另外，在复杂的实时场景中，对多个子系统同时实施变更是一项高风险操作。在 DDS 全局数据配置中心上引入了类选举的算法，使 DDS 系统能够自动保证消息链路变更的时序，减少了系统升级过程中的不可控制因素。

2.2 DDS 发现机制

RTPS 发现协议是 DDS 节点发现机制的基础，它将发现协议拆分成了两个独立的协议：参与者发现协议（Participant Discovery Protocol，PDP）和接入点发现协议（Endpoint Discovery Protocol，EDP）。系统中的参与者首先会通过 PDP 来发现彼此，进而再通过 EDP 来交换所需信息。DDS 是支持多种发现协议的，只要参与者之间支持至少一种相互兼容的协议，它们就能够相互发现并交换信息。然后，为了保证可互操作性，DDS 规范要求所有的 RTPS 实现必须支持两种最基本的发现协议：简单参与者发现协议（Simple Participant Discovery Protocol，SPDP）和简单接入点发现协议（Simple Endpoint Discovery Protocol，SEDP），它们统称为简单发现协议（Simple Discovery Protocol，SDP）。

在 SPDP 阶段，DDS 使用了尽力交付的通讯方式。每个参与者都会在本地数据库中维护着其对端参与者的信息，并监视它们的活跃租期期限，因此，为了防止被对端参与者清除，每一个参与者都需要周期性地向对端发送存活状态通知消息来刷新其活跃状态。在 SEDP 阶段，DDS 使用了可靠的通讯方式。SEDP 协议当且仅当参与者相互匹配后才会被触发执行。作为最基本的 RTPS 发现协议，SPDP 和 SEDP 能够很好地满足从小型规模到中型规模的实时系统的应用场景，然而，目前版本的 DDS 规范中尚未针对大型实时系统制定有效的发现协议。

3 大型 DDS 系统发现机制面临的问题

目前，各个供应商的 DDS 软件大部分只支持最基本的发现协议，如 OpenDDS、RTI DDS 和 CoreDX DDS。这些厂商的 DDS 产品中没有支持更多的发现机制，它们使用的节点信息注册仓库存在着性能瓶颈和潜在的单点故障风险，不具备横向扩展能力[7~9]。OpenSplice DDS 实现了一种原生实时网络协议（native Real-time Networking Protocol，nRTNP）。nRTNP 是一种支持网络分区的预配置协议，它定义了一种 0-发现处理机制，保证了即使无法执行节点发现任务，DDS 系统也可以使用预配置的信息进行最基本的数据投递工作[9]。这在一定程度上提升了 DDS 系统的可用性，但它同样没有解决大规模节点同时执行节点发现时所面临的容量问题。有一些研究机构提出了

一些的改进方案。Javier Sanchez-Monedero 等人将 DDS 系统的消息容量简单的量化为：P*E（P 是 DDS 系统中参与者的数量，E 表示接入点的数量），进而提出了一种基于 Bloom-filter 的发现协议。这套协议将 DDS 系统的容量提升到了 P*P，然而这套协议仅在接入点数量远远大于参与者数量的情况下效果非常明显[3]。谷青范等人提出了一种以 P2P 的分布式结构来存储和管理主题的机制[4]，这是一种从根本上解决发现机制容量问题的方案，能够很好地适应于大型实时系统。然而这套机制却引入了另一个问题：当系统中的有大量参与者变化频繁时，P2P 网络所维护的信息更新时延会影响消息的实时性，即该机制只适用于变更不是很频繁的实时系统。

当大型实时系统中的参与者数量逐渐增多时，目前 DDS 协议存在以下问题：

a. 节点发现过程会产生大量的无效消息，这导致发现处理过程很慢，并且争用了 DDS 系统中大量的带宽和内存资源。在有大量的参与者频繁变更的场景中，情况会进一步劣化。现有的基于中心信息库的 DDS 系统存在性能瓶颈和潜在的单点故障风险。

b. 可管理性下降，随着 DDS 中参与者数量的增加，发布/订阅主题关系错综复杂。在大量参与者节点发生变更时，会出现不可预期的消息发送链路不可用故障。

4 基于主题树的服务发现机制

4.1 TTIR 服务发现机制

本文设计并实现了一种基于主题树的发布/订阅关系信息配置系统（Topic Tree Info Repo，TTIR）来接管 DDS 系统的参与者发现机制。DDS 是基于无中心节点的前提所定义的规范，但 TTIR 采用了中心式设计，这就要求 TTIR 必须具备良好的可靠性，以支撑分布式实时系统的正常运行。

TTIR 基于 ZooKeeper 进行设计实现，具有多个节点的管理系统，这完全有别于现有的基于单节点中心信息库的系统[6]。ZooKeeper 是一个中心式配置信息管理系统，它实现了成熟的配置信息维护、命名、分布式同步、分组服务等机制，目前已经广泛应用于大型分布式文件系统或数据库系统的[10]。TTIR 具有以下特点：

a. 简化的单阶段的发现机制。

b. 高可用，简单的横向扩展能力，使用一主多从的结构。

c. 多维度发布/订阅关系信息组织。参与者元信息以 key-value 方式组织；发现机制通过主题树组织。

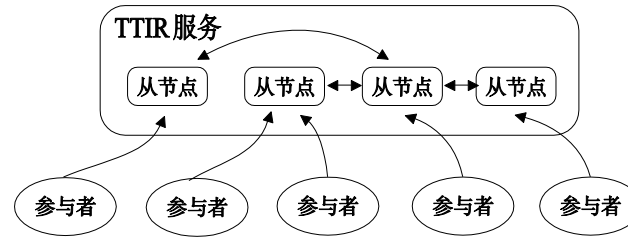

图 2 TTIR 服务

TTIR 中，所有的从节点都是主节点的只读副本，系统参与者与 TTIR 服务建立着长连接，如图 2 所示。在主节点出现不可用的情况下，其中的一个从节点会自动升级为主节点来提供服务，通过 ZooKeeper 的选举机制实现。TTIR 发现协议（TTIR-DP）的节点发现处理流程如下：

a. 一个参与者创建后，首先会向 TTIR 注册 Participant Data 数据，然后它会启动两个线程：接入点发布线程（Endpoint Publish Thread，EPT）和接入点订阅线程（Endpoint Subscribe Thread，EST）。

b. EPT 负责发布接入点的主题信息。它会将 Publication Data 消息发布到 TTIR，然后 TTIR 对接收到的数据进行解析，并将其以主题分支的形式挂载到 TTIR 所维护的主题树上。

c. EST 负责订阅接入点的主题相关信息。它将 Subscription Data 发送给 TTIR，然后 TTIR 对接收到的数据进行解析，并根据订阅需求对主题树上的发布参与者进行过滤。最终 TTIR 将得到一组参与者列表及相关的接入点信息列表，然后将其统一打包返回给参与者。

d. 参与者接收到 TTIR 返回的远端参与者及接入点信息，根据这些信息，与相应的对端建立连接，进行数据通信。

e. 当参与者的 Publication Data 信息发生变更时，EPT 负责重新将其发布到 TTIR。TTIR 检测到信息变更后，会主动将新的发布参与者数据通知到订阅者。订阅者的订阅线程接收到数据后，执行本地的接入点更新。

f. 当参与者的 Subscription Data 信息发生变更时，参与者会重新建立新的 EST，重新从 TTIR 请求数据。这时，旧的 EST 将会被销毁。

以 SDP 协议作为基线比较节点发现过程中产生的消息量。首先构建一个有 P 个参与者、E 个接入点的 DDS 系统，为了减化评估算法，假设：E 个接入点是平均分配在 P 个参与者上面的；只对单播方式的发现机制进行评估，暂不考虑组播方式；忽略网络节点间的存活状态通知消息和 ACK 消息，只关注于发现机制本身产生的消息。

在 SDP 协议中，每一个参与者会向所有其他参与者发送它所持有的接入点的信息，同时，它会接收到所有其他参与者的接入点信息。因此，DDS 系统初始化过程中 SDP 单播方式下的消息总量为：P*(P−1)*(E/P)，约为 P*E，SDP 组播方式下消息发送量为：E，但消息总量仍然为 P*E 量级。同时，每个参与者必须要在本地数据库中保存系统中的所有已经发现的接入点的信息。

而 TTIR-DP 协议中，每一个参与者会向 TTIR 提交它所持有的接入点的信息，同时，每个参与者会向 TTIR 请求主题的对端参与者的接入点信息，并且得到 TTIR 回复的信息，相同主题的多个参与者的接入点信息是聚合为一条回复消息的。除了参与者与 TTIR 之间的通信消息，TTIR 内部的主从节点间同样会进行数据同步，这会占用 E*4 的消息量。因此，基于 TTIR-DP 的 DDS 系统初始化过程中的消息总量（以 3 节点的 TTIR 系统为例）为：E+E*2+E*4，约为 7*E，可以看出，通过 TTIR 管理中心的过滤处理，只有真正有效地发现消息才会在网络上进行传输，因此，TTIR-DP 大大降低了消息数量。同时，每个参与者只在本地数据库中维护与自己直接相关的接入点信息，这使得 DDS 系统本身所消耗的内存在整个内存使用中的占比也达到了最小。

4.2 基于主题树的服务发现与管理

TTIR 使用树形结构对主题进行组织。树的根为 DDS 全局数据空间（GDS）的根路径，向下一级每一级子系统将以一个树分支的形式添加到根上。以此类推，GDS 可以无限分支扩展，直到它能够标识一个确定的数据主题，该主题中将会以三元组的形式存放发布了该主题的参与者、参与者上的接入点，以及参与者和接入点的 QoS 信息：

<Publication Data Ref, Endpoint Data Ref, Routing Dependency>，其中，三元组的各项均为 TTIR 中的结点引用。这些引用所指向的节点上存储了真正的数据集。一棵完整的主题树如图 3 所示。在主题树上，可以通过预置的主题链路锁（Topic Link Lock，PTLL）来对一些主题的依赖关系进行加锁控制。

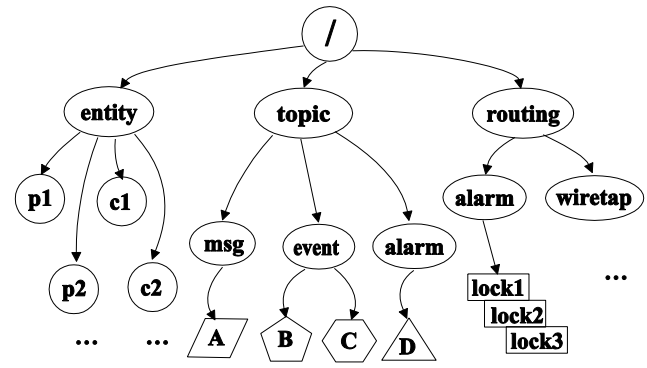

图 3 TTIR 主题树结构

如图 3 所示，/entity 分支中存储了所有 DDS 实体，其中，p1、p2 表示了参与者结点，e1、e2 表示了接入点结点。/topic 分支存储了 DDS GDS 中的发布/订阅主题信息，其中/topic/msg/A 为 DDS 中某个参与者所发布的主题。/routing 中存储了由多个消息链路所组成的消息路径。例如，/routing/alarm 定义了一组保证 alarm 接入点高可用的消息链路，这条链路上有一个链路锁。

假设在 DDS 中有这样一条关键的消息链路：<p1, e1>参与者中接收到各种类型的系统消息，这些消息被通过主题/topic/event/B 或/topic/event/C 传输到事件检测系统进行分析处理。检测系统分析过滤出其中的异常事件，通过主题/topic/alarm/D 投递到报警系统进行报警通知。业务要求所有进入消息系统消息必须能被实时处理，因此，这里要求作为系统关键结点的/topic/event/B 和/topic/event/C 必须是高可用的。可以使用主题链路锁对它们进行保护：

a. 在 DDS 系统中定义一个链路锁，如/routing/alarm，lock1、lock2、lock3 分别表示了该链路中的消息系统、事件检测系统和报警系统。

b. 为要设置链路锁的主题结点添加链路依赖，详细配置信息如表 1 所示。

c. 如果试图对/topic/msg/A 进行删除，TTIR 将响应更新失败。因为该关键链路只有一个系统处于服务状态，如果要对其进行删除，必须首先为该主题添加另一个系统，以保证该链路不中断。

d. 如果试图对/topic/event/B 或/topic/event/C 中的作一链路进行删除维护，TTIR 将响应成功。但如果试图同时删除这两个主题的参与者，那么 TTIR 将响应更新失败，不予以处理，以保证该链路不中断。

e. 对于/topic/alarm/D，可以参照/topic/msg/A 的情况。

表 1 主题树上的三元组配置信息

主题	三元组	描述
/topic/msg/A	<p1, e1, {/routing/alarm/lock1}>	单链路支持的主题
/topic/event/B	<p2, e2, {/routing/alarm/lock2}>	双链路支持的主题
/topic/event/C	<p3, e3, {/routing/alarm/lock2}>	双链路支持的主题
/topic/alarm/D	<p4, e4, {/routing/alarm/lock3}>	单链路支持的主题

5 系统验证

在实验中对 TTIR-DP 的性能和适用性进行了测试，并与 DDS 规范中的 SDP 进行了比较。建立了一组包含有 100、200、300、600、1200 个参与者的 DDS 系统，对发现机制中发送、接收消息的总量和带宽占用进行了比较。为统计方便，假设每个参与者只有一个接入点，同时忽略消息发送失败时的重试消息。

表 2 SDP 与 TTIR-DP 发送消息数量比较

	SDP 单播	SDP 组播	TTIR-DP（3 节点）
100P、100E	10000	100	400
200P、200E	40000	200	800
300P、300E	90000	300	1200
600P、600E	360000	600	2400
1200P、1200E	1440000	1200	4800

表 3 SDP 与 TTIR-DP 接收消息数量比较

	SDP 单播	SDP 组播	TTIR-DP（3 节点）
100P、100E	10000	10000	300
200P、200E	40000	40000	600
300P、300E	90000	90000	900
600P、600E	360000	360000	1800
1200P、1200E	1440000	1440000	3600

从表 2、3 的消息数量统计数据可以看出，SDP 随着参与者数量增长，通讯消息总量增长迅速。而 TTIR-DP 则将通讯消息的总量控制在非常可控的范围内，并且随着系统中参与者数量的增长，消息数量呈现简单的线性增长趋势。

表 4 是带宽占用量的比较数据。由于 TTIR-DP 的消息体比 SDP 要复杂得多，所以在带宽占用量上，TTIR-DP 并没有呈现出如消息数量统计中那样明显的优势，但总体表现大大优于 SDP。

表 4 SDP 与 TTIR-DP 带宽比较

	SDP/MB·s^{-1}	TTIR-DP/MB·s^{-1}
100P、100E	8.5	3
200P、200E	16.2	7.1
300P、300E	40.1	16.9
600P、600E	110	30.3
1200P、1200E	极不稳定，无法监测	52

6 结束语

本文通过研究 DDS 规范和实现中的节点发现机制在大型实时系统场景中面临的问题，提出了一种基于主题树的节点发现管理系统 TTIR。以 TTIR 中易于扩展的主题存储和匹配计算能力，换得了对 DDS 系统的发送消息量和带宽消耗的节省。同时，TTIR 改善了大型 DDS 系统中发布/订阅关系的可管理性，有效地提升了 DDS 系统的可靠性。TTIR-DP 协议目前只支持单播方式的节点发现机制，这是未来可以继续改进和提升的部分。

参考文献

1 Object management group (OMG). Data Distribution Service for Real-time Systems Specification Version 1.2 [S]. 2007

2 Object management group (OMG). The Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specification Version 2.1 [S]. 2009

3 Sanchez-Monedero J. Bloom filter-based discovery protocol for DDS middleware [J]. Journal of Parallel and Distributed Computing, 2011, 71(10): 1305～1317

4 谷青范. 动态自适应 DDS 实时中间件的研究与实现[J]. 计算机科学，2012，39(7)：36～38，73

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

6 刘浩. 一个面向服务的 DCPS 信息库的设计与实现[J]. 航空计算技术，2013，43(5)：101～103

7 Object computing, Inc. Open DDS Developer's Guide [EB/OL]. 2014. Real-time Innovations, Inc. RTI Connext Core Libraries and Utilities User's Manual [EB/OL]. 2013

8 Prism tech. OpenSplice DDS Version 6.x C Tutorial Guide [EB/OL]. 2011

9 The apache software foundation. ZooKeeper: A Distributed Coordination Service for Distributed Applications. 2014