动态自适应DDS实时中间件的研究与实现

谷青范[1,2] 康介祥[1] 冯国良[2] 付宇卓[3]
(航空电子系统综合技术重点实验室 上海 200233)[1]
(南京航空航天大学计算机科学与技术学院 南京 210016)[2] (上海交通大学微电子学院 上海 200240)[3]

摘 要 从动态服务发现和容错两个角度对现有DCPS DDS中间件进行了改进。将目前的发布/订阅模型的信息集中存储结构修改为分布式P2P分布式结构,并利用Chord协议实现主题信息的发布和匹配查询,从而实现了自适应能力。在传统心跳检测模型的基础上,提出了加速推拉模型,以提高失效节点的检测效率。最后通过对实现的中间件进行测试,验证了其在相对动态变化不很频繁的情况下,具有实时、容错、自适应的特点。

关键词 数据分发服务,动态自适应,容错,中间件
中图法分类号 TP393 **文献标识码** A

Research on Implementation of Dynamic Adaptive Real-time Middleware Based on DDS

GU Qing-fan[1,2] KANG Jie-xiang[1] FENG Guo-liang[2] FU Yu-zhuo[3]
(Science and Technology on Avionics Integration Laboratory, Shanghai 200233, China)[1]
(College of Computer Sicence and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)[2]
(School of Microelectronics, Shanghai Jiaotong University, Shanghai 200240, China)[3]

Abstract An adaptive DDS middleware was implemented by improving the DCPS DDS middleware on data service finding and fault-tolerance, which achieves adaptability by modifying the centralized information store model with peer-to-peer model and applying chord protocol for data publication and subscription. Based on the traditional heart-beating model, a heart-beating accelerate push-pull model was proposed for enhancing the node failure detecting efficiency. Finally, a middleware was implemented. The test result shows that the implemented middleware is characterized with real-time ability, fault-tolerance and adaptalility.

Keywords Data distributed service, Dynamic adaptive, Fault-tolerance, Middleware

1 引言

协同作战是未来战争发展的必然趋势,要实现协同作战,就必须有提供战场态势信息共享能力的系统。全球信息网格GIG(Global Information Grid)[1]就是美军网络中心站和夺取信息优势的基础设施,它充分利用计算机、通信网络和性能优越的处理器,实现目标跟踪与识别、雷达信息捕获提示以及所有作战单位协同作战的功能。经过近20年的发展,协同作战系统已经比较成熟,但还存在带宽要求高、数据共享盲目、网络规模扩大困难以及开放性差等不足。为了构建我军协同作战能力网络,很有必要研究协同作战系统中数据通信基础设施,以解决目前存在的通信瓶颈问题。用于协同作战的"数字化战场中枢系统"的数据链网络是一个典型的分布式实时网络,需要存在一个实时通信中间件,实现各个分布节点之间的实时信息传输,同时还能够将异构的节点以松耦合的方式连接起来,从而确保网格节点的动态加入和离开不影响现有的系统架构配置。数据分发服务(Data Distribution Service, DDS)[2]是OMG继推出CORBA[3]规范后,专门针对分布式实时系统中数据发布/订阅模型颁布的一个最新规范。其定义了以数据为中心的发布/订阅(Data-Centric Publish-Subscribe)机制,提供了一个与平台无关的数据模型;此外,DDS规范还定义了大量的QoS策略,使得DDS可以很好地配置和利用系统资源,协调可预测性与执行效率之间的平衡,以支持复杂多变的数据流需求等。本文基于DDS DCPS模型,对应用于协同网络作战环境的实时通信中间件自适应关键实现技术(包括服务动态发现和节点容错两个方面)进行了研究和实现。

本文第2节介绍了DDS DCPS模型及相关研究;第3节针对协同作战环境、现有DDS中间件实现存在的问题,提出了动态自适应DDS中间件应满足的需求;第4节讨论了动态服务发现和容错的实现机制并给出实验结果;最后给出了结论和进一步的研究方向。

2 DDS的发布/订阅模型及相关研究

DDS规范的核心就是标准化了以数据为中心的分布式应用"订阅/发布"通信模型——DCPS模型和应用程序编程

到稿日期:2011-09-22 返修日期:2011-12-01 本文受南京航空航天大学基本科研业务费专项科研项目(NS2010092)资助。
谷青范(1974—),博士后,副教授,主要研究方向为分布式计算、软件工程、嵌入式操作系统,E-mail:qingfangu@nuaa.edu.cn;康介祥(1969—),研究员,主要研究方向为航空电子、嵌入式操作系统;冯国良(1986—),男,硕士生,主要研究方向为软件工程、中间件技术。

接口，DCPS模型如图1所示。

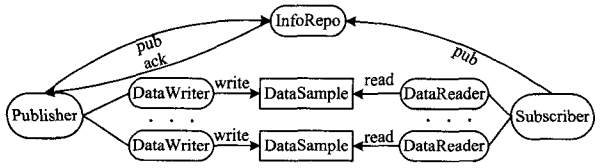

图1 DDS的发布/订阅模型(DCPS)

该模型包括信息库(InfoRepo)、发布者(Publisher)、订阅者(Subscriber)、数据写者(DataWriter)、数据读者(DataReader) 5个参与者(其中前3者是系统的主参与者，而数据写者是由发布者创建的，一个发布者可以创建多个数据写者。类似地，数据读者是由订阅者创建的，一个订阅者可以创建多个数据读者)和与之相对应的确认信息(acknowledgement)、发布主题信息(publication)、订阅主题信息(subscription)以及数据实例(DataSample)。发布者向信息库发送它的发布主题信息，订阅者向信息库发送它所感兴趣的订阅主题信息，信息库匹配发布主题信息和订阅主题信息及QoS(如数据持久度、更新周期等)，若匹配，则告之对它的主题感兴趣的订阅者，之后发布者将相应的数据实例发送给对应的订阅者，而订阅者通过它的数据读者读取数据实例。这里QoS策略可以定义在不同的层次(如主题、发布者、数据写者、数据读者、订阅者)。文献[4]提出了一种应用于移动Ad hoc网络中的发布/订阅协议，其着重考虑了发布/订阅协议与多播协议的结合，而多播在实际使用中往往会受到路由器的限制；文献[5]在DCPS模型基础上提出了分层的以指挥节点为核心的信息分发模型，其仍然属于集中式的数据存储模式。

3 协同作战环境下DDS存在的问题

从实现角度，目前遵循DDS标准、采用DCPS模型的中间件实现的有RTI(Real-Time Innovation)公司的RTI DDS[8]、PrismTech的OpenSplice DDS[7]以及OCI开源代码OpenDDS[9]，其都利用了多播机制来提高多个“订阅者”情况下数据传输的效率和可靠性。但是这些中间件需要在系统运行之前对多播地址进行预先配置，这主要适用于相对静态的局域网环境，无法满足动态变化的广域协同作战环境。协同作战环境中节点不仅是移动的，而且不断有节点退出和新的节点加入，这就要求中间件具有动态自适应的能力。因此，需要存在一套动态服务发现框架来实现组播地址的动态配置，具体地讲，需要解决以下两个问题：

(1)DDS中间件的一个主要优点是可扩展性比较好。新节点的加入只需要向信息库注册即可，不需要事先知道其他节点的情况或整个网络拓扑，但在协同作战环境下，组播网络是动态变化的，如果不了解现有组播网络情况，新节点将无法加入到现有组中。

(2)信息传递具备确定性和容错性。例如发布者发布数据后，如何确保在一定的时间内，接受者必须接收到并给予响应；假设某个参与信息交换的节点遭受攻击或硬件失效后，新的备份节点能够自动加入数据链网络中并正常执行信息的交互。

4 基于DDS的动态服务发现及容错实现

4.1 自适应服务发现实现机制

为了满足动态自适应的要求，需要改进现有DDS在信息库中集中存放域、主题、发布者、订阅者的实现机制，需要将这些信息分布存放在动态网络中的各个节点中，形成P2P网络[6]，利用P2P路由机制来实现域的查找和主题的匹配。

4.1.1 P2P构建及路由查找实现机制

发布者、订阅者以节点的形式加入到P2P网络中，节点以IP地址和关键字的哈希值为序分布在一维环上。任意一个节点P仅维护有限个(n个)其他节点中的主题索引信息，这些主题索引信息通过一个m位的二进制字符串(key)来标识。节点和key构成节点的路由表(finger table)[11,12]，查询关键值key时，每个节点通过查询Finger表，将搜索请求发送到离key最近的节点上，直至定位到key所在的节点。称发起订阅的节点为查询源节点，称存有所需主题的信息的节点为查询目的节点。查询源节点首先对主题名称做哈希变换并询问其路由表中最接近此哈希值的节点(完成第1跳)，再由此最近的节点继续其路由表中最接近目的的节点(完成第2跳)，如此反复，直到找到查询目的节点。具体算法描述如下：

```
/* this 表示当前节点，predecessor 为当前节点的前驱节点。
  closestPrecedingFinger(key)是在 Finger 表中查找距离 key 最近节点的函数。
  SendData()是向下一跳(消息状态为 REQUEST)或目的节点(消息状态为 REFRESH)发送查询数据消息的函数。
*/
if (predecessor ! = null && key∈(predecessor,this)
{
  对 key 进行本地处理;
  return;
}
  if (finger[0] ! = null && key∈(this,finger[0])
  {
    sendData(message,finger[0],REFRESH);//找到目的节点
  }
else
{ //取 finger 表中最近节点
  closeFinger= closestPrecedingFinger(key);
  //下一跳最近节点在 Finger 表中
  sendData(message,closeFinger,REQUEST);
}
```

注：这里路由节点中的资源指的是发布/订阅的主题索引信息，主题内容的传输直接在相应的两个节点之间进行。当发布者和订阅者加入或离开时，某些资源标识符会重新进行分配，从而达到网络的自适应。

4.1.2 域及主题发现过程

为了获取特定域的特定主题，订阅者首先查询本地系统，如果没有相应域的存在，需要启动组播会话，进入消息路由，直到找到目标主题，算法主要步骤如下：

Step1 在时刻t_q，节点i发起查询请求$R_i\langle domain_id, topic_id\rangle$，经过一致性哈希函数转换为key值，检查该key能否在本地机上得到满足，如果满足，则查询成功，返回结果$\langle domain_id, topic_id, publisher_id\rangle$，否则转向Step2。

Step2 申请一个组播地址IP_m，随机产生一个64位长的组会话标志grp_id，根据目标主题键值$topic_id$和预设的搜索范围TIL创建查询消息的四元组$msg\langle grp_id, topic_id, TIL, IP_m\rangle$，启动组播会话，根据Chord路由协议将消息发

送到其后继节点。

Step3　转发查询信息到中间节点 j，收到查询消息后首先判断本地系统有无目标主题，若存在，则结束消息路由，转到 Step4；否则，采用基于键值相似度的回溯算法查询本地路由信息表，确定下一转发节点。转发前，节点将根据组成员筛选算法判断是否将当前节点加入组播组 grp_id。路由终止的条件是找到目标主题或超出搜索范围。

Step4　通过 IP_m 发送主题。

4.1.3　容错实现机制

容错是一种可信度保障机制。容错技术的基本思想是在系统体系结构上精心设计，利用外加资源的冗余技术来降低屏蔽故障的影响，从而自动地恢复系统正常运行或达到安全停机的目的，达到高可靠性的目标。容错技术的基础是冗余，因此，在该 DDS 中间件中，每个节点都有备份节点。备份节点通过心跳机制来检测主节点的"活性"。在传统的心跳检测机制中，最常用的是推模型(Push)和拉模型(Pull)以及它们的各种变体。在推模型中，信息流和控制流的方向是一致的，被监控对象自发地、周期性地向监控者发送心跳包，以告之它还活着。这种模型传输效率高、实现简单，但容易引起误判。而在拉模型中，监控者周期性地询问被监控对象的健康状态，每次询问的同时设定一个超时时间为 T 的定时器，如果在 T 时间内能接收到被监控对象的回复信息，则认为该对象还"活着"，反之则判断该对象已无效。因此与推模型相比较，拉模型的传输效率较低；并且在请求服务比较密集的场合中应用时，心跳周期往往设置比较短，该模型就有可能导致服务器端信息处理的负荷过重。

本文在传统心跳检测模型的基础上，提出了一种新的检测模型——加速推拉模型。如图 2 所示，在检测过程中同时使用了推模型和拉模型。加速推拉模型的工作过程分为两个阶段。在第一阶段，被监控对象和监控者之间使用推模型，即被监控对象周期性地向监控者发送心跳包。正常情况下，监控者都能在当前心跳周期 T 内收到被监控对象的心跳包。当出现监控者接收心跳包超时时，监控自动转到第二阶段，在该阶段，使用拉模型。由监控者向被监控对象发起询问，被监控对象收到询问后作回复。这里的拉模型和传统的拉模型有所差别，在传统的拉模型中，监控者等待接收被监控对象心跳包的超时时间为 T，并且是固定不变的。当在 T 内未能收到该对象的心跳包则认为该对象已经失效，这就很容易引起误判。而在这个阶段的拉模型中，超时时间不再是固定的 T，而是 $T/2, T/4\cdots$，这就提高了检测的速率；并且不再是像传统的拉模型那样，一次超时则认为被监控对象已无效，而是当出现一次超时，则超时时间减半，监控者再次向被监控对象发起询问，又一次超时则超时时间减半，再次询问……如此反复直到超时时间小于某个临界值 t 时，才判定被监控对象已经无效，监控者不再给被监控对象发起询问。通过这种方式可以排除数据丢包、网络传输延迟等不确定因素，这就大大降低了误判的概率，同时提高了失效节点的检测实时性。由于反复询问给系统增加额外的负担，因此 t 值应该结合网络状况和系统负载平衡来确定，在实际系统实现中，需要反复测定来确定一个经验值。

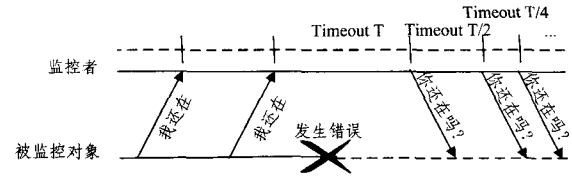

图 2　加速推拉模型中的信息传递

进一步分析该模型，监控者判定被监控对象异常的最大检测延迟为 $\sum_{i=0}^{n}\frac{T}{2^i}$，其中 $\frac{T}{2^n}>t$ 且 $\frac{T}{2^{n+1}}>t$，由于 $\lim_{x\to\infty}\sum_{i=0}^{n}\frac{T}{2^i}=2T$，因此 $\sum_{i=0}^{n}\frac{T}{2^i}<2T$。

由此可知，监控者收不到被监控对象的心跳包时，在 $2T$ 时间间隔内就能判断被监控对象无效，并做相应的处理。

本实时中间件正是通过该模型的心跳机制来检测节点的状态，而心跳检测周期可以根据实际运行的网络状况进行方便的配置。在系统中，可以通过 OWNERSHIP 和 OWNERSHIP_STRENGTH 这两个 QoS 策略来控制，Subscriber 将只接收 STRENGTH 最高的 Publisher 的数据。当某个 Publisher 出现故障之后，其备份节点将成为 STRENGTH 最高的 Publisher，接着，相应的 Subscriber 将从该发布者那里接收数据。备份节点将和主节点一样一直运行并发布数据，但数据读者将丢弃备份节点的数据直到主节点发生故障。当然，也可以方便地配置，使备份节点只有在主节点发生故障时才发布数据，否则不发布任何数据。

5　实现验证

依据前面提出的方法，我们针对动态网络环境实现了一个中间件，并着重对其动态适应性、容错性和实时性能进行了测试，评估其在协同作战环境中的适应能力，验证其在协同作战环境中对实时性的满足；并将其与采用 DCPS 模型的 OpenDDS 中间件进行了比较。下面首先对测试方法进行阐述，然后对试验结果进行分析。

衡量一个软件实时性最重要的指标是时延(Delay 或 Latency)。所谓时延就是指数据(一个报文或分组，甚至比特)从网络(或链路)的一端发送到另一端所需的时间。为了避免时钟同步等问题，本文采用返回确认信息并统计平均的方法：

$$Tavg\_delay=\sum_{i=1}^{n}T_i/n=\sum_{i=1}^{n}(T_i+T_i')/n-\sum_{i=1}^{n}T_i'/n \quad (1)$$

式中，$Tavg\_delay$ 为时延的平均值，T_i 为发送端第 i 次给接收端发送数据所耗的时间，T_i' 为返回确认信息所耗的时间，n 为连续发送的次数。测试时采用一对一的方式，即一个发布者和一个订阅者，发布方数据写者写一个序列的负载数据，范围从 2k 字节到 2048k 字节，成倍增加；订阅方数据读者接收到数据后发送一条 4 字节的应答信息，采用请求/应答协议保证往返时间戳记录在发布节点上，消除时间偏移问题。发布者通过测试数据传输到接收到应答信息的时延，可以评估在不同的负载数据条件下，一个节点到另一个节点数据传输的速度。为保证测试结果的准确性，本文采用两个发布者连续向订阅者发送 10000 次数据并取平均值的方法；由于应答信息是固定的 4 字节，因此可以事先求出返回确认信息所耗的

(下转第 73 页)

[8] Sousa R M, Putnik G D. Formal description technique SDL for manufacturing systems specification and description[C]// APMS 1999. 1999:449-456

[9] Ellsberger J, Hogrefe D, Sarma A. SDL[M]. Formal Object-oriented Language for Communicating Systems, Prentice Hall, 1998

[10] Messer A, Kunjithapatham A, Sheshagiri M. Interplay: A middleware for seamless device integration and task orchestration in a networked home[C]// Proceeding of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications. 2006:296-307

[11] Ranganathan A, Al-Muhtadi J, Chetan S. MiddleWhere: A middleware for location awareness in ubiquitous computing applications[C]// Proceeding of the 5th ACM International Conference on Middleware. 2004:397-416

[12] Ranganathan A, Campbell R H. A middleware for context-aware Agents in ubiquitous computing environments[C]// Proceeding of the ACM International Middleware Conference. 2003:143-161

[13] Kadous M W, Sammut C. MICA: Pervasive middleware for learning, sharing and talking[C]// Proceeding of the Second IEEE Conference on Pervasive Computing and Communications Workshop. 2004:176-180

[14] Lyytinen K, Yoo Y. Issues and challenges in ubiquitous computing[J]. Communications of the ACM, 2002, 45(12):62-65

(上接第 38 页)

平均时间 $Tavg\_ack$。求解方法是让发布者同样发送 4 字节的数据，因而可以采用往返时间的一半作为 $Tavg\_ack$。从而：

$$T_{avg\_delay}=\sum_{i=1}^{n}(T_i+T_i')/n-T_{avg\_ack} \quad (2)$$

为了模拟动态适应性和容错性，为每个节点(包括发布者和订阅者)都配备一个失效备份节点。不失一般性，为了降低测试复杂度，试验中将备份节点采用静态配置的方式，为了模拟节点动态变化情形，采用周期性更新节点路由表的方式，心跳检测的临界值设为 0.5ms，利用失效节点率(即失效的节点数占总节点数的比率)来评估网络的动态变化情况。

实验环境搭建了 28 个节点，每 14 个节点在一个网段内(备份节点在相同的网段内)，每个节点都是普通的 PC，网络采用 1000Mbps 交换机和 10M 集线器两种连接方式。根据上述测试方法对中间件反复测试，取其平均值，结果如图 3 所示，其中数据大小的单位为 1024 字节(kB)，时延的单位为微秒(μs)。

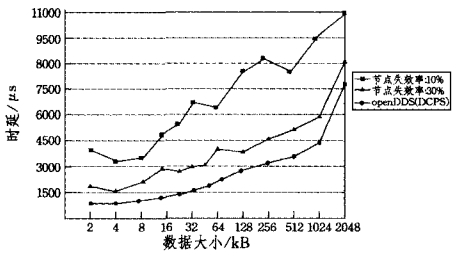

图 3 测试结果

由图 3 可知，时延总体随着数据量的增加而增加，而在数据大小相等的情况下，openDDS 具有最小的时延，这也验证了在网络拓扑固定的情况下，P2P 路由相比集中式路由，花费的时间更多；在存在节点失效率小于 10%的情况下，时延能够控制在 10ms 内，可以满足协同作战环境的实时性要求(毫秒级)；而在失效率为 30%时，可以看到时延明显增大，并且随着传输数据的增多出现跳跃现象，说明路由表的更新严重影响了时延的稳定性。从上面的试验结果可以看出，实现的自适应中间件在动态变化不是很频繁的情况下能够满足实时要求。

结束语 本文通过引入 P2P 架构和 Chord 协议，对现有 DDS 中间件从动态服务发现和容错两个角度进行了改进；并在传统心跳检测模型的基础上，提出了加速推拉模型，以提高系统的实时容错性。实验结果验证了其具有实时、容错、自适应的特点。在一个主题多个订阅的情况下，利用组播技术可以大大提高传输效率，因此下一步的研究方向就是结合组播协议和 Chord 协议来优化现有的路由查找协议。

参 考 文 献

[1] http://www.nsa.gov/ia/industry/gig.cfm

[2] Object Management Group. Data Distribution Service for Real-time Systems[R]. Version1.2, Jan. 2007

[3] Object Management Group. High-performance CORBA Specification[R]. Version1.2, Jan. 2005

[4] 翟立东，刘元安，马晓雷，等. 发布/订阅通信机制在移动 Ad Hoc 网络中的应用[J]. 北京邮电大学学报，2008，31(2)：30-33

[5] 韩松，张晓林，占巍，等. 基于空中指挥节点的信息分发模型及时延分析[J]. 系统工程与电子技术，2009，31(11)：2677-2681

[6] Shirky C. Wat is P2P and what isn't[C]// O'Reilly's Emerging Technology Conference. 2002

[7] An Introduction of openSlice DDS. pdf [EB/OL]. http://www.openslice.org

[8] RTI DDS Qos and Features. pdf. [EB/OL] http://www.rti.com.

[9] Steven Stallion. Using Reliable Multicast for Data Distribution with OpenDDS [EB/OL]. http://mnb.ociweb.com/mnb/MiddlewareNewsBrief-201002.html

[10] Clarke L. A distributed decentralized information storage and retrival system[D]. Master's thesis. University of Edinburgh, 1999

[11] Stoica, Morris R, Karger D, et al. Chord: A scalable peer-to-peer lookup ser vice for internet application[C]// Proceddings ACM SIGCOMM. 2001

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

[13] 贺建立，陈榕，顾伟楠. 一个事件驱动的中间件平台[J]. 计算机科学，2010，37(5)：107-111