企业服务总线研究综述*)

谢继晖　白晓颖　陈　斌　肖思南
(清华大学计算机科学与技术系　北京 100084)

摘　要　企业服务总线(Enterprise Service Bus, ESB)结合事件驱动技术,提出了一种新的服务连接架构,以实现服务间基于标准、松散耦合的通信方式,为基于服务的系统集成和扩展提供了更有效的支持。本文首先介绍了 ESB 的基本概念,然后从消息机制、消息转换、消息路由、服务容器四个方面分析了 ESB 核心功能组件,讨论和比较了当前主要的 ESB 软件产品和开源项目。ESB 技术尚处在发展初期,本文分析了 ESB 研究目前存在的主要问题以及今后研究方向。

关键词　企业服务总线,SOA,消息机制,消息路由

A Survey on Enterprise Service Bus

XIE Ji-Hui　BAI Xiao-Ying　CHEN Bin　XIAO Si-Nan
(Department of Computer Science and Technology, Tsinghua University, Beijing 100084)

Abstract　ESB (Enterprise Service Bus) is a key infrastructure for SOA (Service-Oriented Architecture) with event-driven technology. It is a new architecture which connects standard-based and loosely-coupled services. ESB enables transparent system integration and enhances system extensibility. This paper first introduces the general concepts of ESB. It then analyzes the state-of-the-art research on the four key components in ESB, including message mechanism, message transformation, message routing and service container. The paper also introduces major ESB products and compares three open-source ESB projects. It finally discusses the main problems of ESB and future research directions.

Keywords　Enterprise service bus, SOA, Message mechanism, Content-based routing

1　概述

2002 年,Sonic 软件公司提出企业服务总线(ESB, Enterprise Service Bus)的概念,将其定义为"集成了消息机制、Web Services、消息转换和智能路由的基于标准的集成主干"[1]。2003 年,Gartner 在面向服务体系架构(SOA, Service-Oriented Architecture)技术的基础上,提出了事件驱动的体系架构(EDA, Event-Driven Architecture)[2]。EDA 采用发布/订阅机制,具有异步处理能力,又被称为事件驱动的 SOA。结合 EDA,ESB 技术发展到新的阶段。IDC 提出了更为宽泛的概念,将 ESB 定义为开放的基于标准的连接主干[3]。IBM 认为 ESB 是 SOA 的核心和基础[4]。

ESB 实质上是服务间的连接框架,其核心功能包括消息转换、消息机制、基于内容的路由和服务容器四部分。ESB 采用基于 XML 规范的消息格式,可支持多种标准,如 Web Services、JMS、JCA 等标准。可支持发布/订阅及请求/回复等同步/异步消息机制。通过基于内容路由技术,支持服务之间消息通信。服务集成支持 JBI(Java Business Integration)标准。服务容器将软件构件和应用封装成标准的服务,以实现对软件的访问以及对服务总线访问透明。在上述技术的支持下,ESB 使服务实现与服务通信相分离,服务之间松散耦合,基于标准协议建立通信联系,使应用系统具有更好的开放性和可扩展性。

作为新兴的软件框架和 SOA 的核心组件,很多公司和研究机构进行了 ESB 软件的研发,如 Iona, Sonic, BEA, IBM 等。本文简要介绍了 IBM Service Integration Bus 和 BEA Aqualogic 产品的主要特点,并分析比较了 Mule, Celtix, ServiceMix 三个主要的 ESB 开源软件项目。

ESB 技术目前尚处于发展初期,相关研究还在探索之中。针对 ESB 的四个核心功能组件,本文分析了当前 ESB 存在的主要问题以及今后的发展方向。

2　ESB 体系架构概述

ESB 主要包括消息机制、消息转换、消息路由和服务容器四个主要构件。

2.1　消息机制

消息机制提供管理计算资源和网络通信的机制,屏蔽分布环境复杂性和异构性,为应用程序提供透明的通信服务。ESB 的消息机制采用通信通道(channel)抽象服务之间的消息通信,服务之间建立通道联系。ESB 可支持两种通信模式:发布/订阅和点到点的请求/回复的消息模式。

发布/订阅是异步消息传递模式,发布者发布的消息可传递给多个订阅者。在发布/订阅模式中,首先订阅者将订阅信息发布到 ESB,发布者在发布消息后,ESB 将消息转发给相关订阅者。发布/订阅方式一般通过主题树(Topic Tree)实现。主题树以消息发布者为根接点,父、子结点之间表示了一种发布/订阅关系。每个父结点将消息发布给其子结点;每个子结点可以选择接受或拒绝来自父结点的所有消息。

*)本文得到国家"863"计划项目(No. 2006AA01Z157)和国家自然基金研究项目(No. 60603035)资助。**谢继晖**　硕士研究生,研究方向:软件工程;**白晓颖**　硕士生导师,研究方向:软件工程;**陈　斌**　本科生;**肖思南**　本科生。

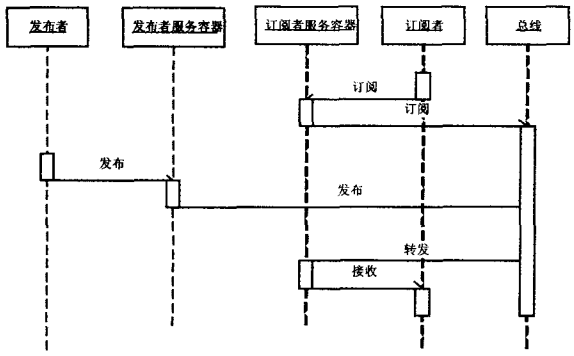

图 1　发布/订阅模式消息传递图

请求/回复模式是服务提出请求，其它服务响应回复的模式，每个消息仅传递给一个消费者。它可以是同步也可以是异步的。同步方式中，请求方等待回复以进行后续操作；异步方式中，请求方无需等待回复消息。请求/回复模式可以有单向/双向两种消息通道。单向通道只传递请求或回复消息；双向通道中一个通道可同时传递请求和回复消息。请求/回复一般通过队列实现。每个服务都可建立其请求和回复队列。

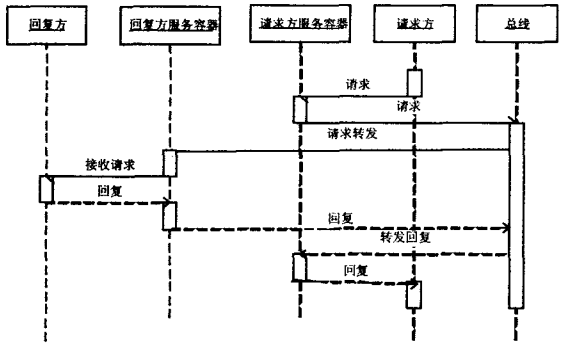

图 2　请求/回复模式消息传递图

2.2　消息转换

消息是自包含、自治的实体。一个消息通常由消息头、消息属性以及消息体三个部分构成。自包含消息定义是解耦合（避免两个服务之间直接通信）、实现异步消息机制的关键。通常 ESB 选择已有的企业数据模型或工业标准消息作为内部通用的基于 XML 的标准通信协议和消息格式，例如目前电子商务中的 xCBL 格式[5]、cbXML[6]等。

连接在总线上的服务种类很多，可能采用不同的消息协议，其对于信息的需求也不同，例如两个通讯的服务 AA 采用时间格式是“年年年年-月月-日日”，而 B 采用“月月-日日-年年年年”。因此，需要对消息进行转换。消息转换包括消息通信协议的桥接和消息内容转换[7]。协议桥接实现不能直接通讯的协议之间消息的传递，内容转换支持对不同消息内容的转换。IBM[7]还提出，ESB 还可以补充、完善消息内容，例如在消息中添加消息的来源等信息。

图 3 显示了消息转换的主要过程。

消息桥接有三种实现方式：1）通过一个共享文件缓冲区暂存交互的消息，消息交互方可采用不同的协议，往缓冲区中存、取消息；2）通过第三方标准协议，如 JMS、JBI、SOAP 等；3）ESB 直接提供对不同消息协议的处理能力，如将 HTTP 或 SOAP 请求映射到 JMS 消息。此外 ESB 可采用例如 XSLT（eXtensible Stylesheet Language Transformation）技术，通过 Schema 映射的方式，实现对不同的 XML 消息格式的的转换。

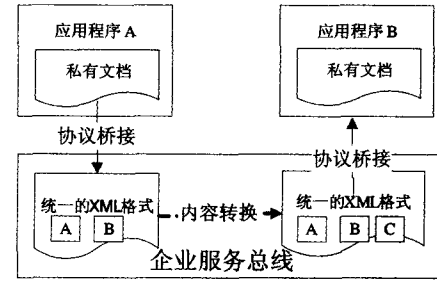

图 3　消息转换

2.3　消息路由

消息路由是分析服务传递的步骤，建立传递线路和规则，并逐步传递消息的过程。ESB 可根据消息内容将其由提供者传递到接受者。消息路由主要包括路由线路和路由规则两个部分。

路由线路描述了服务将要发送的地址和路由规则，包含在消息的元数据中。它分为不带分支判定的简单消息线路和包括分支、判定、条件等执行过程的复杂消息线路。简单消息线路上的各个服务结点顺序执行；复杂处理过程在复杂的路由中，需要支持流程分离（Split 图 4-A）、流程聚合（Aggregation 图 4-B）以及复杂分支判定（图 4-C）等多种处理机制，通常用服务工作流语言如 BPEL4WS 来描述。

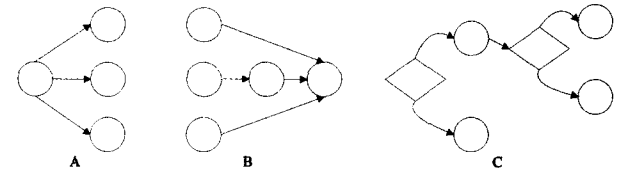

图 4　复杂线路

ESB 路由规则多采用分布式的管理方式，即规则分散在各个服务节点，由该节点的服务容器管理。路由规则定义了消息传递和路由线路的选择策略，例如需要满足的前提条件、路由时间延迟要求、可允许的失败连接的次数等。在复杂路由线路选择中，路由规则为其提供决策依据。

2.4　服务容器

服务容器是将各种类型的软件组件或应用，封装成可支持标准通讯协议（如 JMS、JBI、JCA、Soap 等）的服务，并抽象成一个端点（endpoint），连接到总线上的组件。服务容器既可以封装用户应用软件，也可以封装 ESB 的基础服务。

为了实现分布式处理，服务容器需要支持服务的注册、发现和选择。有两种方式实现服务发现：一种方式是服务容器本身不管理注册信息，通过查询总线全局注册表，获取服务信息；另一种方式是服务容器维护一个本地注册表，通过本地查询即可获得服务信息[8]。

通过服务容器，可以实现对软件的局部管理和全局管理相结合的方式。局部管理是服务容器对其所封装的软件的管理，包括生命周期管理、连接管理等。ESB 可以通过标准协议如 JMX（Java Message eXtension）框架、SNMP（Simple Network Management Protocol）等集成其它管理工具（如 IBM Tivoli，HP Openview 等），实现对服务容器的全局管理，如远程配置管理、路由规则管理等。通过服务容器管理，可实现容

器选择性功能布置。

服务容器屏蔽了软件的异构性，使得总线的基础服务对每个特定的服务软件透明。对应用软件而言，容器是总线的代理，服务容器是基于ESB的系统可扩展性的重要保证。

2.5 其它

ESB还提供一些高级服务，以和现有的应用集成技术兼容，并有效集成遗产系统。例如，ESB可集成基于PD4J(Process Definition for Java)、BPEL(Business Process Execution Language)等标准的服务引擎，提供可支持JCA(Java Connector Architecture)标准的适配器，以及集成XQuery或者SQL的数据查询引擎。

3 研究现状

ESB出现时间不长，其解耦合、异步通信等方面还存在很多问题。目前对ESB技术的研究还较少，大多研究是从SOA与ESB的关系[9]方面考虑，例如将BPEL引擎引入ESB[10]，提出ESB服务分层模型等[2]。增强ESB核心部件功能，实现松散耦合的集成体系架构是将来ESB研究的重点。

3.1 消息机制

消息机制的主要研究问题是如何提高发布/订阅、请求/回复两种消息传递机制的处理能力、适应性和安全性。

发布/订阅一般通过消息代理(broker/proxy)实现[11]，这样会存在单节点瓶颈。为了解决这个问题，M. Ivan等提出CAMX(Computer Aided Manufacturing using XML)框架，利用集群的概念，将本地消息代理集群化，以提高处理能力的发布/订阅系统[12]。

ESB统一的消息机制难以适应实际应用中不同软件和数据库的需求。针对适应性问题，Ludger Fiege提出了Rebeca消息机制[13]，引入了"域"(Scoping)的概念，将消息通知(notification)的接收者按照其特点划分为域，同一个域的服务具有相同的一组属性，采用同样的方式来进行消息通信。根据域的应用需求，可对不同的域提供不同的模型化和定制化的系统工程工具。

现有的消息机制依赖于静态的访问控制策略，即当策略发生变化时，系统不能动态加载以实现新的控制机制。但在电子商务等应用中，动态的访问控制非常重要。IBM学者Zhao Yuanyuan提出动态访问控制的确定性模型[14]，确定性指的是不管事件的发起端的位置、网络延迟或失败等都对事件保密性有确定；模型是一系列定义清晰的关于访问控制的描述，它包含在动态访问控制中涉及的实体及其角色，通过基于内容的表来确切描述访问控制规则与访问控制变化的起点。动态访问控制的确定性模型，通过对访问控制进行预归类来实现基于内容的可靠传递的动态访问控制。

除此之外，还有学者研究消息容错性、安全验证、性能改进和QoS等问题。WSMQ[15](Web Services Message Queue)就是一个利用数据库存储消息，以提高Web服务消息中间件通讯可靠性的项目。

3.2 消息转换

消息转换的主要研究如何提高消息转换的智能、转换准确率和处理速度。

消息转换一般采用XSLT技术分析Schema，实现不同消息格式的转换。但由于没有考虑客户端环境及其转换规则的特点，无法与客户端能力、用户或网络状态等环境信息结合起来。Kinno提出基于XML的环境和规则描述方法，并将基于XSLT的消息格式转换与基于DOM的消息内容扩充结合，使消息转换能够感知环境和规则，具有更高智能[16]。

ESB中多采用基于XML的消息格式，但随着应用的增长，存在着大量的异构的XML数据定义。现在多是人工方式，定义不同XML元素之间的对应关系，难以满足大规模的消息转换的需求。如何有效建立起XML定义的消息格式之间的映射关系，实现消息的自动转换，是现在研究的一个关注点。A. Bouko- ttaya提出了一种基于语义Schema分层化模型的消息转换方法LIMXS(Layered Interopera- bility Model for XML Schemas)。LIMXS在传统的基于Schema的语法分析的基础上，定义了Sche- ma的语义模型，定义了数据的构成和解释方式，通过描述Schema语义的概念、关系和限制，实现基于Schema语义匹配的消息转换[17]。

传统的内容转换中，针对XML的转换处理是顺序进行的，效率低、时间延迟长。一些学者从算法角度研究如何提高基于XSLT的XML文档的转换效率。Steffen Schott将XSLT的优化分为静态分析和动态分析两种方法[18]，并提出了一种"懒惰"的处理方法，即通过简单预处理，可以初步建立其文档的结构如DOM树，但不需要对文档进行完整的分析，而是在用户查询或是使用特定数据的时候再对该数据结点信息展开分析。目前大部分都是采用文本分析方法分析XML信息，文档处理过程中会有很多额外的开销。而BiM[19](a Binary Format for Metadata)在二进制域下进行XML序列化和XML树操作，以降低XML处理开销，提高对XML的处理能力。当前存在众多的XML转换方法，不同的转换方法有不同的转换的XML文档定义，D Fotsch和A Speck，提出了XTC[20](XML Transformation Coordinator)的概念，以集成不同的文档定义和转化方法；而Eric Wohlstadter等则引入了AOP(Aspect Oriented Programming)风格的概念称为content-based pointcuts[21]，以集成不的XML技术并支持多文档横向分析。

3.3 消息路由

消息的基于内容路由目前主要有两种实现方式：组播方式和基于过滤器方式的(Filter-based)路由。

组播方式是基于事件空间Event Space Partitioning(ESP)分类的方法，将所有的服务按照事件分成组，消息在组内广播。这种方式的优点是消息只在定义的组内传播，路由表简单。缺点很依靠分组的策略，组内流量还是比较大。

过滤器方式是在所有的路由的节点中匹配和响应订阅信息，然后将消息传递给下一个处理节点。过滤器的方式实现了分布式的消息处理。但缺点是路由计算量大，速度慢，增大了中间节点的负担，而且每个服务器都需要维护消息路由表用于保留分布式消息网络的状态信息以及其消息传递的路径。在ESB中，大量的消息传递会导致规模巨大的路由表，大规模的路由表同样会导致路由表查找困难。

Babavar等提出可以自学习的中间路由优化技术，在消息传递的过程中，消息接受节点能够向其传递路径上的各个节点反馈订阅确认信息，以便在后续消息传递中，避免对非订阅信息的转发[22]。这种方法可以降低中间节点的负担，提高处理速度，同时减少冗余消息。针对路由URL表规模大，查找速度慢的缺点，Zornitza等提出采用URL签名来取代URL在路由表中的内容，以降低路由表规模、同时将签名作为哈希路由表的键值以哈希方法提高路由表查找速度[23]。

Cao Fengyun等将组播与过滤器中间路由处理相结合，提出Kyra发布/订阅方法[24]，将订阅信息和事件在预选择的

道路中进行传递以实现更细粒度分组方式，做到路由表和流量的平衡。

组播与过滤器都是采用 XPath 来判定和匹配服务。为了提高对 XPath 的处理速度，Ashish 等提出在消息产生过程中对 XML 消息进行解析时，先对一部分 XPath 值作出计算，放在消息包中，以后对值进行匹配。如果成功，则不用再次解析，否则再进行解析来改善服务器对 XML 消息的验证处理能力[25]。

3.4 服务容器

对于容器的研究，主要集中服务发现、服务选择。

服务容器可支持两种服务发现方式：一种方式是服务容器本身不管理注册信息，通过查询总线全局注册表，获取服务信息，这样每次需要服务都要请求全局注册表，受到网络状况的限制比较大，也没有能实现松散的耦合结构。另一种方式是服务容器可维护一个本地注册表，当中心服务目录或者本地服务发生变化时，需要实现与中心目录的通讯，以保证两个目录的一致性。针对这两种发现方式的缺点，Emerson 等从发布/订阅的体系架构上作改进，提出将容器之间建立层次关系，采用树形结构存储，每个容器存储本地服务的注册表，每个服务需要查找服务时首先看本地是否能满足，不能满足则将请求提交到父亲容器，查找父亲目录。依据此降低冗余的方法，能保证目录的简单性，同时实现服务发现机制[26]。

在 ESB 中，一个总线上应该可以存在多个描述相同的服务。如何在这些相同或者功能类似的服务中进行选择，以获得快速响应和高质量的服务是一个关键点。Vijay Dheap 等提出 EDRA[27]（Event- Driven Response Architecture）系统，在 ERRA 系统中，采用一个构件 Context Container 来存储客户端的注册信息、上下文相关服务以及预定义的变化应对策略，通过这个智能的学习机制，实现基于用户选择、系统配置以及可用的服务的服务动态选择。

除此之外，最近有研究提出采用容器实现 BPEL[28] 及实现安全可靠的机制[29] 等非功能需求。服务容器的封装目前主要用 XML 来代替硬编码绑定服务，这与理想的服务容器封装后，动态使用服务还有很大的距离，但是由于 XML 配置文件的不可动态载入性，这方面进展还很缓慢。

4 ESB 软件

作为一个新兴的软件框架，逐渐引起了越来越多公司和研究机构的关注。ESB 软件还处于前期探索开发阶段，本文主要从商业产品和开源项目两个方面简要分析了其软件开发的现状。

4.1 ESB 产品

当前市场上已经有很多 ESB 产品，各大产商为了在 ESB 市场中占有一席之地，除了提供 ESB 基本功能以外，都在不断对其功能进行扩展：Iona4 月发布的 ESB 产品中集成了 BPEL 引擎；Sonic 发布的 ESB 7.0 支持更多标准如 WS-Reliable Messaging，WS-Addresing，WS-Security 和 WS-Policy 等。目前 BEA 和 IBM[30] 的 ESB 产品在市场上占据技术领先地位。

IBM 的 WebSphere Application Server V6 产品中，Service Integration Bus(SIB)是其实现企业服务总线的核心[31]。SIB 可支持请求/回复和发布/订阅两种消息机制，提供可靠消息传递保障；可动态选择服务和连接服务端口；可支持采用 WSDL 标准的消息转换；实现 SOAP/HTTP - SOAP/JMS 协议桥接；可实现聚合与关联等复杂处理过程的基于内容的路由；支持 WS、JCA 和 JDBC 多种标准，并能对 WS 进行性能监控。

BEA 的 ESB 产品是 AquaLogic Service Bus[32]，消息代理作为服务消费者与服务提供者之间的中介，是其 ESB 产品的核心。AquaLogic 支持同步异步的请求/回复及异步的发布/订阅外，还实现了同步-异步桥接（请求者要求同步，服务提供方异步）；支持策略驱动和消息线路的消息路由和消息转换；实现一致的服务注册及消息的证书映射、认证授权等。

4.2 ESB 开源项目

在 ESB 领域还有一些由公司主导的开源 ESB 项目，其中，Mule，Celtix 和 ServiceMix 三个项目影响较大。

Mule 开发比较早，是一个基于 J2EE1.4 轻量级的开源 ESB。它采用消息代理机制，在事件传递的过程中，实现可靠的 SEDA(Staged Event-Driven Architecture)消息传递。

表 1　3 个开源项目的比较

	Mule	Celtix	ServiceMix
消息机制			
发布/订阅	主题树实现	不支持	主题树实现
请求/回复	同步/异步、单向/双向	同步/异步、单向/双向	同步/异步、单向/双向
消息分发	端到端路由，不经总线分发	总线分发	端到端路由，不经总线分发
消息转换			
总线消息格式	无统一格式	无统一格式	无统一格式
支持的协议	JMS HTTP SOAP EMAIL 等	JMS HTTP SOAP Native XML JCA JBI SCA 等	JMS HTTP SOAP EMAIL FILE JCA WSIF Jabber 等
内容转换技术	String 或 Object 配对	XSLT Xpath	XSLT Xpath Drools
消息路由			
路由规则	不支持	不支持	不支持
动态路由	无	无	无
服务容器			
服务封装	XML 封装	XML 封装	XML 封装
服务注册	手动静态注册	手动静态注册	手动静态注册
服务发现	总线注册表、无本地目录	总线注册表、无本地目录	总线注册表、无本地目录
容器管理	支持 JMX 与 JSW	支持 JMX	支持 JMX

Celtix 是 Iona 基于 Java 的开源 ESB，Celtix 中，服务提供方必须采用 Servant 组件进行封装。Servant 根据 WSDL

生成服务的类框架，开发者在此框架下实现具体服务。同时目前正在开发多种集成功能，如针对 JBI 的功能。

ServiceMix 是一个建立在 JBI 上的开源 ESB。它采用 ActiveMQ 和 Mule 实现路由、消息机制等功能，同时利用 ActiveSoap，Groovy，Drools，PXE，Aglia 等开源项目实现管理、协议转换等功能。

表 1 从 ESB 主要功能的各个方面对三个开源项目进行了比较。

现有的开源 ESB 项目与理想的 ESB 还有很大差距，还主要存在以下几个方面的问题：

* ServiceMix 和 Mule 的路由为任意通讯的服务之间静态配置消息通道(channel)，n 个服务互相通讯需要 $O(n^2)$ 级通道；

* 都不支持动态路由，在总线启动时即将所有的路由线路固定，不支持服务动态添删或注册；

* Celtix 不支持发布/订阅。

5 主要问题

5.1 动态路由

动态绑定是 SOA 的一个主要特点。在 ESB 中，服务流程中每个服务结点采用抽象的服务描述，在实际执行过程中，抽象的服务描述可动态绑定到不同的服务实现上。动态路由就是指服务线路不是固定的，路由线路可以动态改变的机制。

动态路由可以在两个层次实现。第一个是服务动态绑定，即在路由过程中，根据服务的负载情况、消息线路的长度、外界环境等因素来确定选择满足同一服务功能的最佳的服务实现，动态连接到服务结点。第二个是服务线路可以改变。由于业务流程变化等因素，需要重新组合服务，定义服务的执行序列，从而导致服务线路的重新规划。

5.2 路由控制机制

分散路由控制和集中路由控制。分散的路由控制：在通讯的服务之间建立通道，在路由过程中不需要通过总线的分配；路由的路由规则和路由命令都在各服务节点完成，不用通过中心控制器；集中的路由控制是有一个中心路由控制，所有的路由控制命令都在此发出(如分支判定、聚合、下一个)，中心控制器根据中心路由规则来决定通讯消息的下一步该发往哪里。在集中的路由控制中，又包含了两种方式：第一种是所有的通信消息(包括数据块)都提交到总线，总线服务将通信消息传输到下一个服务接受点，第二种是中心路由控制器只做路由控制，并在通讯双方建立连接机制，然后让通信双方通过建立的连接通道进行通讯。

分散路由控制对分布性要求高，需要在分布式的机器上具有很多本地目录备份，而且由于一致性问题，会导致很多开销，但是其分布式特点也使得其路由控制的鲁棒性好，耦合程度低；集中式路由很容易受到网络状况影响，并且容易带来单机性能瓶颈。

5.3 一致性问题

在服务容器的描述中提到了两种服务发现方式：服务容器本身不管理注册信息和服务容器维护一个本地注册表；除此之外，在松散耦合的环境中，发布/订阅目录、路由规则等很多都可能存在本地目录备份和 ESB 中心目录。当中心目录或本地发生变化时，需要实现保证本地目录与中心目录的一致性。

从方法上而言，可以采取两种方法来解决这个问题：中心目录发生变化时，通知所有与之相关的本地目录进行修改，以保证两者的一致性；或者中心目录发生变化并不通知其相关的本地目录，而是在本地目录需要被使用的时候，对中心目录进行查询，如果中心目录相对于该目录的相关服务等发生了变化，那就更新本地目录，以保证一致性，否则本地目录不发生变化(懒惰方法)。

第一种方法下，本地服务不需要在发布消息的时候获取中心目录列表等信息，但是有可能，在其发送消息的时候，其目录还没有更新，造成与中心目录信息的不一致性；而第二种方法能很好的保证中心目录和本地目录的一致性，但是会导致性能下降，同时会增加对中心节点的依赖，增加整个系统的耦合程度。

5.4 性能问题

性能是 ESB 的一个很重要的方面。发布/订阅机制中，如何实现基于内容的消息传递是提高消息机制性能的关键点；其次，在 ESB 中大量的消息以 XML 格式存在，XML 消息转换是提高消息机制处理速度的瓶颈；此外服务的查找和绑定中，服务列表的查找和服务的动态选择效率也很重要。

对于发布/订阅采取的组播或者广播带来的大量消息导致的性能处理瓶颈，可以采取在总线中发布的消息只存在一份拷贝，它包含有订阅者信息，总线安排它遍历总线的服务消费者，消息在消费方进行复制。这样解决了冗余信息的问题，但消息的传递速度而造成处理速度较慢。而在发布/订阅机制中一般通过消息代理实现，这样存在单节点瓶颈。为了解决这个问题，目前提出的集群化思想方法存在动态集群化不完善、集群节点间协作等问题。

根据处理层次，XML 消息转换性能提高可分为文档优化、编码优化和硬件固化三种。

文档优化一般通过预解析方法实现。XML 以预解析的状态存在时，可以采取多处理器同时处理的方式来加快对文档的处理速度；也可以通过缓存的方式来实现性能的提高。如 XML 日志服务在 XML 消息传递过程中，将解析后的 XML 消息进行复制，保存在该服务中，以后涉及到该消息的获取或者审查和跟踪都由该服务提供解析过的 XML 文档，从而减少解析的次数。如果在处理过程中 XML 消息发生了改变，则可以采用在 XML 消息传递过程中，对 XML 消息中的某些数据进行搜集，将不同数据源的数据存放在同一个 XML 文档中，该文档以解析后的形式存在的方式实现一次解析多次使用，实现不同数据源的数据的整合。

编码优化则针对目前 XML 采用文本分析方法分析 XML 信息多额外开销的特点，在二进制域下进行 XML 序列化和 XML 树操作，以降低 XML 处理开销，提高对 XML 的处理能力。

硬件固化指的是将 XML 的处理在硬件实现，利用硬件处理的速度来优化和提高对 XML 消息的处理能力。

为了提高服务查找和选择的效率，主要可从发布方和路由过程入手。在发布方，一方面可以采取本地注册表备份来提高查找速度，另一方面需要注意控制注册表同步开销；在路由过程中，采取合适的取代措施来降低路由表的规模，如采用 URL 签名取代路由表中的 URL[23]，同时采用更好的算法来提高路由表查找速度和传递速度。

结束语 作为一种新的体系架构概念，结合了 SOA 和 EDA 思想 ESB 已经开始成为当前研究的热点和主流，其松散耦合和标准化思想也将在软件开发和信息系统集成中发挥

重大的作用。采用ESB软件的信息系统能实现服务的动态发现和重用，具有很好的可扩展性。基于目前研究内容及ESB软件的局限性，今后的ESB的研究重点将集中在松散耦合的消息机制实现、基于内容路由的动态修改和服务容器封装功能的完善以支持多种服务。

参 考 文 献

1 Sonic Software Corporation. Sonic ESB: An Architecture and Lifecycle Definition. Sonic White Paper, 2005

2 宋宇宁. 基于SOA的事件驱动企业应用集成技术研:[学位论文]. 浙江大学, 2005

3 Hudson S. The Enterprise Service Bus: Disruptive Technology for Software Infrastructure Solutions. IDC Insight, 2003 March, Volume 1 #29132

4 Keen M, Acharya A, Bishop S, et al. Patterns: Implementing an SOA Using an Enterprise Service Bus. IBM Press, 2004

5 http://www.xcbl.org

6 http://www.cbxml.com

7 Maréchaux J L. Combining Service- oriented Architecture and Event-driven Architecture Using an Enterprise Service Bus. IBM Developworks, Mar 2006

8 YANG Gang, ZHOU Xingshe, WU Xiaojun. MAM: A Novel Mixed Adaptive Messaging Middleware. In: Proceedings of the Second International Conference on Machine Learning and Cybernetics, Xi'an, Nov 2003. Vol. 3. 1913

9 Luo Min, Goldshlager B, Zhang Liang-Jie. Designing and Implementing Enterprise Service Bus (ESB) and SOA Solutions. In: Proceedings of the IEEE International Conference on Services Computing, 2005. 14

10 Zimmermann O, Doubrovski V, Grundler J, et al. Service-oriented Architecture and Business Process Choreography in an Order Management Scenario: Rationale, Concepts, Lessons Learned San Diego, California, USA, Oct 2005. 301～312

11 Muthusamy V, Petrovic M, Jacobsen H A. Routing protocols: Effects of routing computations in content-based routing networks with mobile data sources. In: Proceedings of the 11th Annual International Conference on Mobile Computing and networking. Cologne, Germany, August 2005. 103～116

12 Delamer I M, Lastra J L M, Tuokko R. Unified Service-oriented Architecture for Federated and Locally Distributed CAMX Publish/Subscribe Middleware. In: 3rd IEEE International Conference on Industrial Informatics, 2005. 117～122

13 Fiege L, Cilia M, Muhl G, et al. Publish-Subscribe Grows Up: Support for Management, Visibility Control, and Heterogeneity. IEEE Internet Computing, January 2006, 10(1): 48～55

14 Zhao Yuanyuan, Sturman. Dynamic Access Control in a Content-based Publish/Subscribe System with Delivery Guarantees. In: 26th IEEE International Conference on Distributed Computing Systems, July 2006. 60

15 Maheshwari P, Tang H, Liang R. Enhancing Web Services with Message-Oriented Middleware. In: Proceedings of the IEEE International Conference on Web Services, 2004. 524

16 Kinno A, Yonemoto Y, Nakayama T, et al. Environmentally-adaptive XML transformation and its application to content delivery. In: Communications IEEE International Conference, May 2003, 2: 844～848

17 Boukottaya A, Vanoirbeek C, Paganelli F, et al. Automating XML documents transformations: a conceptual modeling based approach. In: Proceedings of the first Asian-Pacific Conference on Conceptual Modeling, January 2004. Volume 31: 81～90

18 Schott S, Noga M L. Document querying and transformation: Lazy XSL transforma tions. In: Proceedings of the ACM Symposium on Document Engineering, Grenoble, France. Nov 2003. 9～18

19 Van Lancker W, De Sutter R, De Schrijver D, et al. A framework for transformations of XML within the binary domain. In: Proceedings of the 24th IASTED International Conference on Internet and Multimedia Systems and applications, Innsbruck, Australia, Feb 2006. 28～34

20 Fotsch D, Speck A. XTC－The XML Transformation Coordinator for XML Document Transformation Technologies. In: Database and Expert Systems Applications 17th International Conference, Sept 2006. 507～511

21 Wohlstadter E, De Volder K. Doxpects: aspects supporting XML transformation interfaces. In: Proceedings of the 5th International Conference on Aspect-oriented Software Development, Mar 2006. 99～108

22 Banavar G, Chandra T D, Mukherjee B, et al. An efficient multicast protocol for content-based publish- subscribe systems. In: International Conference on Distributed Computing Systems, 1999. 262～272

23 Prodanoff Z G, Christensen K J. Managing routing tables for URL routers in content distribution networks. International Journal of Network Management, May 2004, 14(3): 177～192

24 Cao Fengyun, Singh J P. Efficient event routing in cotent－based publish－subscribe service networks. 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, March 2004(2): 929 ～ 940

25 Gupta A K, Dan Suciu, Halevy A Y. The view selection problem for XML content based routing. In: Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, San Diego, California June 2003. 68～77

26 Loureiro E, Bublitz F, Barbosa N, et al. A Flexible Middleware for Service Provision over Heterogeneous Pervasive Networks. In: Proceedings of the 2006 International Symposium on World of Wireless, Mobile and Multimedia Networks, June 2006. 609～614

27 Dheap V, Ward P A S. Event-driven response architecture for event-based computing. In: Proceedings of the 2005 Conference of the Centre for Advanced Studies on Collaborative Research, Oct 2005. 70～82

28 Charfi A, Mezini M. An aspect-based process container for BPEL. In: Proceedings of the 1st Workshop on Aspect oriented Middleware development, Grenoble, France, November 2005. Article No. 4

29 Cotroneo D, Cotroneo D, Graziano A. Security Requirements in Service Oriented Architectures for Ubiquitous Computing. In: 2nd Workshop on Middleware for Pervasive and Ad-Hoc Computing, Toronto, Ontario, Canada, 2004. 172～177

30 Gilpin M, Vollmer K. The Forrester Wave(tm): Enterprise Service Bus, Q4 2005. In: Tech Choices, November 2005

31 Sadtler C, Aggarwal A, Cotignola D, et al. Patterns: Implementing Self-service in an SOA Environment. IBM Press, January 2006

32 BEA. BEA Aqua Logic Service Bus™ A Technical Review of Architecture and Functionality for Service Integration and Management. BEA White Paper, 2006