面向传感网的服务构建与并发控制

童恩栋，牛温佳，谭红艳，赵志军，唐 晖

(中国科学院声学研究所 北京 海淀区 100190)

【摘要】讨论了无线传感器网络在复杂环境下的服务构建及并发控制。基于传感器网络能量、通信能力、计算存储能力等有限的特点提出了面向传感器网络的服务管理，提出了基于推理的上下文感知工作流来构建传感器网络服务。通过提取传感器数据的语义信息，实现了上层业务逻辑和底层传感器数据的松耦合，实现了传感器网络的资源重用。此外，改进了Web服务请求的并发控制，通过服务请求的重复和冲突检测，避免重复和冲突的服务执行，实现更有效的传感器网络并发控制。

关 键 词 复杂网络； 并发控制； Web服务； 无线传感网； 工作流

中图分类号 TP391 **文献标识码** A doi:10.3969/j.issn.1001-0548.2011.03.001

Service Building and Concurrency Control in Wireless Sensor Network

TONG En-dong, NIU Wen-jia, TAN Hong-yan, ZHAO Zhi-jun, and TANG Hui

(Institute of Acoustics, Chinese Academy of Science Haidian Beijing 100190)

Abstract This paper researches the service building and concurrency control in wireless sensor networks (WSNs). Due to the characteristic of limited resources such as energy, communication, computing and storage in WSNs, a WSN service management approach and a reasoning-based context-aware workflow to build WSN services are proposed. By abstracting the semantic information from sensor data, the loose couple between the upper business logic and lower sensor data can be achieved and the resource reuse can further realized. In addition, the traditional concurrency control methods are improved. With the help of repeat and collision detection, more effective WSN service concurrency control can be received.

Key words complex networks; concurrency control; Web service; wireless sensor networks; workflow

现实世界中，网络形式的系统随处可见，如因特网、产品供应链网络、交通网络及社会关系网络等。近年来随着复杂网络研究热潮的兴起[1-2]，学者们研究了各种复杂网络的特性。如何在复杂网络环境下，保证网络的高效性和可靠性，已经成为复杂网络研究的重要课题。

无线传感器网络(WSN)指的是将红外感应器、全球定位系统等具有传感、数据处理和无线通信能力的传感器节点通过自组织方式形成的网络，用于协作地感知、采集和处理网络覆盖区域内被测对象的信息，并通过无线多跳的通信方式实现与人类社会的交互。由于其传感器节点数量巨大，且传感器节点的移动、失效等特征会导致传感器网络的结构不断发生变化，因此，无线传感器网络是典型的复杂网络，无线传感器网络研究是复杂网络研究的一个重要方面。

由于传感器网络强大的数据收集能力，极大地扩展了人们获取信息的能力。因此被应用在诸多领域，例如智能家居[1]、工业自动化[4]、精准农业[5]、智能监控[6]等。传感器网络存在节点和网络协议等异构性，而Web服务由许多应用程序接口(API)组成，隐藏了服务实现细节，用户只需通过网络提交服务请求调用相应API获得服务，保证了异构应用系统之间的互操作。因此，Web服务得到了广泛应用。而传感器网络的每个应用就是一个Web服务。

Web服务流程包括服务的构建、发布和请求执行。服务由服务提供者提供，服务提供者将自己的服务接口信息在服务注册中心注册，以提供服务使用者发现和访问服务。服务使用者根据自己的需求查询服务注册中心，获得能够满足需求的服务描述，根据服务描述向服务提供者发送服务请求调用相应API。其中，Web服务通过Web服务描述语言(web service description language，WSDL)描述Web服务和说明如何与Web服务进行通信；通过统一描述、发

收稿日期：2011－04－10
基金项目：国家重大专项(2009ZX03004-001)；中国科学院知识创新工程方向性项目(KGCX1-YW-19，KGCX2-YW-149)
作者简介：童恩栋(1986－)，男，博士生，主要从事语义Web服务、智能信息处理方面的研究.

现和集成协议(universal description discovery and integration，UDDI)向Web服务注册中心定义API接口；通过基于简单对象访问协议(simple object access protocol，SOAP)在应用程序间交换信息。

面向服务架构(service-oriented architecture，SOA)将应用程序的业务逻辑(business logic)模块化为不同功能单元(称为原子服务)，而这些服务之间通过定义良好的接口和契约联系起来。由于SOA服务易于构建和共享原子服务的特点，很多研究[7-8]尝试将SOA应用在传感器网络的服务构建中。目前关于SOA的研究可以归为智能规划[9]和工作流[10]两类。其中工作流技术由于其相对成熟被广泛应用[11]。二十世纪九十年代，由于普适计算[12]的提出，上下文感知工作流应运而生。上下文感知工作流是指工作流可以随上下文的变化而动态地改变，以更准确地完成任务。传统的上下文感知工作流尽管实现了工作流的动态改变，但在原子服务的重用方面考虑不够。在服务请求执行方面，目前Web服务请求主要采用SOAP协议。SOAP请求消息一般搭载HTTP协议进行传输，并发控制也是基于传统的HTTP并发控制方法，而没有考虑SOAP请求消息的具体内容。

Web服务技术为Internet的服务管理提供了有效的技术支撑。然而，面向具体的传感器网络应用，Web服务呈现出以下两个新特点：

1) 传感器网络的服务主要建立在各种传感设备之上，通过与底层传感设备的数据交互实现特定的功能。例如，一个温度传感器就可以构建入侵检测和空调控制等多个服务。因此，传感器网络传感设备种类和数量的增加将导致传感器网络中Web服务规模呈指数级增长趋势。

2) 由于底层传感设备的高度互联，传感器网络的服务依赖关系也将更加紧密。在这种情况下，同一个服务资源往往可能要被多个服务调用和操作。例如，在提供游客实时视频浏览的公园旅游场景下，摄像头就属于公共的服务资源。服务S_1可以提供查询摄像头是否空闲的服务，而服务S_2也可以提供对该摄像头的移动操作。

由于上述传感器网络Web服务的新特点，需要设计面向传感器网络的服务管理方法，该服务管理方法应该在服务构建时重视传感器网络资源的重用，在服务请求执行时考虑服务间的依赖关系。Web服务管理可以进行改进以实现资源重用的环节包括服务的构建和服务的请求执行。因此，本文提出了一种面向传感器网络的服务构建与并发控制方法。该方法使用基于推理的上下文感知工作流用以构建服务以及改进的Web服务并发控制方法。具体方法如下：通过推理抽取出传感器数据隐含的语义信息，进而使用语义信息刻画原子服务的输入输出，实现上层业务逻辑与底层传感数据的松耦合，从而构建更加动态的工作流。同时，在服务的请求执行方面，改进了已有的Web服务并发控制方法，考虑传感器网络服务间的依赖关系。在收到服务请求信息后，首先解析得到SOAP请求消息的具体内容，通过新增的并发控制模块优化服务请求消息的执行，实现更有效的服务请求并发控制。

本文内容组织如下：第1节介绍了相关工作，第2、3节分别详细介绍了本文提出的面向传感器网络的服务构建与并发控制，第4节通过实验与案例分析该服务构建与并发控制在传感器网络中的优势。最后，第5节对全文做了一个简单总结。

1 相关工作

本文提出的服务管理方法使用基于推理的上下文感知工作流构建服务和改进的并发控制方法优化服务请求。

上下文感知工作流源于上世纪九十年代提出的普适计算[12]，随后上下文感知成为工作流领域的研究热点[13-15]。文献[16]提出了CAWE(context-aware workflow execution)，使用抽象的工作流实现上下文感知。工作流由抽象的原子服务组成，每个抽象的原子服务对应一组实现具体功能的原子服务实例。当前上下文用来决定使用哪一个原子服务实例，以实现上下文感知的工作流。文献[17]把上下文作为条件构建服务子树。当上下文改变时，工作流会通过选择子树实现工作流的动态重组，达到上下文感知的目的。以上工作虽然实现了工作流随上下文改变而动态的改变，但因忽略了资源的重用，并不适用资源受限的传感器网络。

服务请求并发控制方面，基于SOAP的并发研究较少，但由于SOAP多由HTTP承载进行传输，因此HTTP的并发控制对本文有重要参考价值。HTTP并发访问控制多使用线程池技术[18]。文献[19]提出一条SOAP消息携带多条服务请求，在解析出SOAP的服务请求消息后，通过应用层增加的线程池处理SOAP消息携带的多条服务请求。因此，本文提出解析出SOAP的服务请求消息后，经并发控制模块优化后由新增的线程池执行服务请求。

2 基于推理的上下文感知工作流

在本文的服务管理方法中，采用工作流技术来构建服务。二十世纪九十年代，普适计算[12]的提出，研究者们开始尝试将上下文感知引入到工作流的管理中来，上下文感知工作流成为研究的热点。

传感器网络能量、通信能力、计算存储能力有限，且传感器网络服务高度依赖底层传感器数据，传统的上下文感知工作流与传感器数据紧耦合，因此存在大量冗余的原子服务，造成传感器网络的资源浪费。以温度控制服务(获取房间内的温度值并判断是否需要打开/关闭空调)为例。在不同的环境下，温度判断的条件也各不相同(T>58 ℃，T>100 ℃等)，因此需要相应地构建多个原子服务，但它们其实包含同一语义信息，即当前温度高或低。

本文引入基于规则的推理技术，抽取出传感器数据中的语义信息，进而服务提供商设计新的原子服务替代已有原子服务，其中新的原子服务的输入输出由推理出的语义信息进行刻画，实现上层业务逻辑与底层传感数据的松耦合，从而构建工作流时多个工作流可以共用相同的原子服务，最终提高工作流的动态性和资源的重用性。还是以温度控制服务为例，构建新的原子服务(判断当前温度高或低)并使用语义信息(温度高或低)作为输入来替代已有的原子服务(判断温度值是否大于58 ℃或100 ℃等)。这样，尽管所处环境不同，温度控制服务可以共用新建的原子服务。图1所示为基于推理的上下文感知工作流模型。

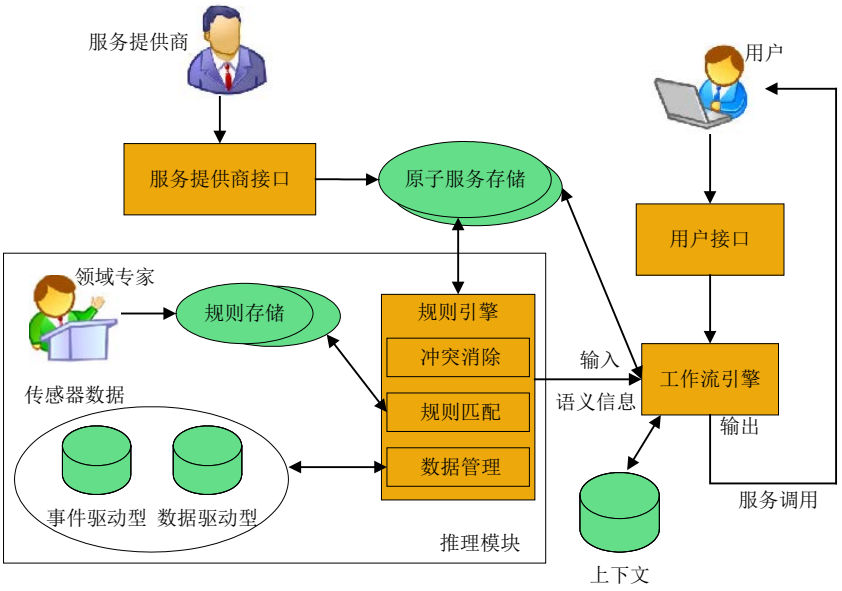

图1 基于推理的上下文感知工作流模型

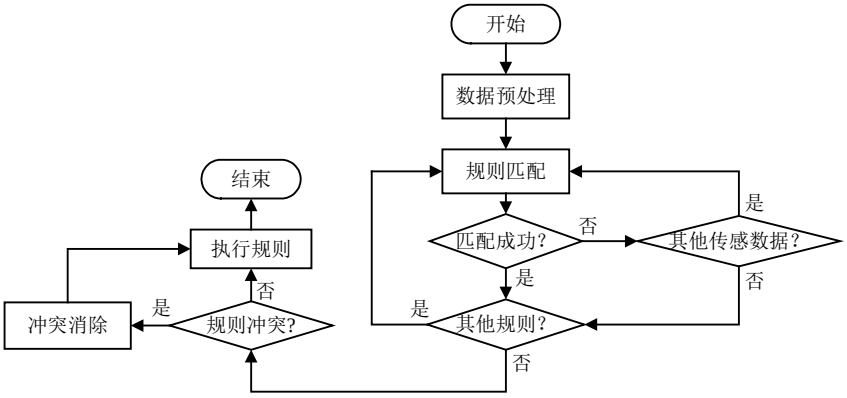

图2 规则引擎工作流程

其中，规则引擎是该模型的核心，通过预先定义的规则抽取出当前传感数据的语义信息；服务提供商建立原子服务，并使用抽取的语义信息对原子服务的输入输出进行刻画，实现上层业务逻辑和底层传感数据的松耦合，建立的原子服务存储在原子服务存储模块中；工作流引擎用于工作流的建立，

监控和执行。

规则引擎的工作包含规则匹配、规则选择和规则执行3个阶段。规则匹配过程中可能会存在多条规则被同时匹配成功，因此需要规则冲突算法选择适当的规则来执行。图2所示为规则引擎的工作流程。

其中，规则匹配阶段采用RETE算法进行高效实现，规则冲突解决(即规则选择)阶段采用本文提出的基于上下文感知优先级的冲突消除算法实现。

RETE算法是一个用来实现产生式规则系统的高效匹配算法[20]。该算法的核心思想是一种根据规则的条件部分生成RETE网络，数据由根节点进入网络进行匹配，过程中网络会储存匹配结果，达到显著降低计算量的效果[21]。

规则冲突的解决办法多是采用对冲突中的规则分配不同优先级的方式。传统的优先级取值是唯一的，而在上下文感知环境中，当前场景下某一规则可能有一优先级，而在另一场景下，该规则可能有另一不同的优先级取值。

本文提出的冲突解决算法基于上下文感知的优先级，同一规则在不同的场景下分配不同的优先级，从而可提高规则冲突解决方法的准确性。下面所示为本文的温度控制服务引入上下文感知优先级规则实例如下。其中有3个优先级，括号内表示该优先级适用的上下文，包括默认的优先级5，室内的优先级10和室外的优先级15。进行规则选择时，该上下文会与当前上下文进行匹配，若相同则为该规则分配该上下文对应的优先级。为冲突中的规则分配了不同的优先级后，冲突解决算法选择执行拥有最高优先级的规则。

rule name　“temperature”
attribute:　always
priority:　5(default), 10(indoor), 15(outdoor)
object:　int temperature.value;
　　boolean temperature.comfortable;
if:　temperature.value>26(℃) and
　　temperature.value<30(℃);
then:　temperature.comfortable = true;
end

3　服务请求并发控制

目前，Web服务请求主要采用IBM、Microsoft、UserLand和DevelopMentor在1998年共同提出，并于2000年提交给万维网联盟(W3C)的SOAP[22]协议，当前的最新版本为SOAP1.2。SOAP是建立在XML语言基础上的一个轻量的协议，用以规范在一个分布式环境下进行结构化的信息交换。一条SOAP消息就是一个普通的XML文档，包含Envelope、Header、Body和Fault共4个基本元素，通过HTTP进行承载，如图3所示。

对于搭载在HTTP上的Web服务请求，相应的并发控制机制如图4所示。

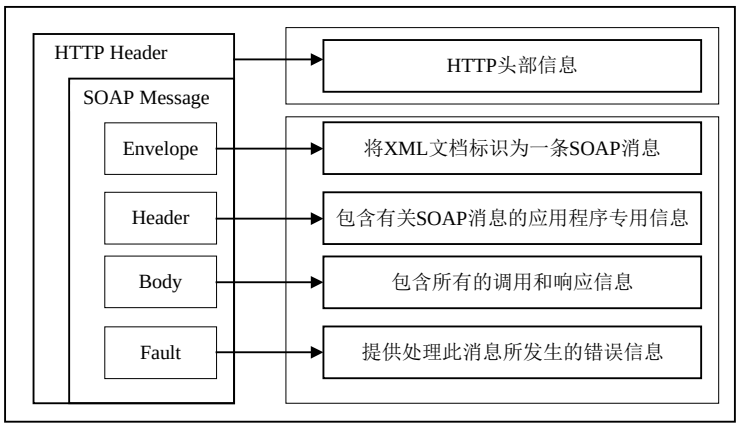

图3　搭载在HTTP上的Web服务请求

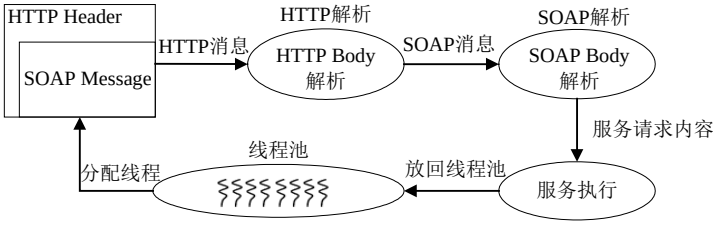

图4　传统Web服务并发控制流程

搭载Web服务请求的HTTP消息到达Web服务器后，线程池分配线程用以处理HTTP消息。首先经过HTTP解析得到搭载的SOAP消息，进一步进行XML解析，获得SOAP Body包含的服务请求内容，由相应的Web服务进行处理，经过与上述过程相反的过程对响应信息进行封装并返回给服务请求端。最后，线程释放该Web服务请求实例并放回线程池以循环执行Web服务请求。以上Web服务并发控制的核心在于：线程池维护确定数量的等待线程，每当有Web服务请求到达，线程池就分配一个线程为请求服务。为了运行这些线程，处理器为每个独立线程安排一些处理器时间，处理器以顺序轮换方式向线程提供时间片，实现多线程的并发处理。线程池的优势在于服务器运行之初一次性创建多个线程并循环使用，将线程创建开销分摊到多个任务上，可避免频繁创建、销毁线程带来的巨大资源开销，以及创建过多的线程耗尽系统资源。

由图4可知，Web服务器在收到服务请求时即分配线程进行响应，而具体的服务请求内容是在经过HTTP及XML解析后获得。传统的Web服务并发控制方法缺陷在于：线程的分配没有考虑HTTP消息中封装的服务请求内容，多个HTTP请求被分配独立的线程单独执行。如果当前的多个HTTP请求封装相同的服务请求，以及服务请求之间有冲突或有执行优先级之分，有可能会造成线程浪费、资源冲突和错误的发生。以提供游客实时视频浏览的公园旅游场景为例，用户*A*用请求服务S_1询问摄像头是否空闲，而用户*B*用请求服务S_2移动摄像头查看景点。当用户*A*与用户*B*的请求同时到达时，目前的Web服务并发控制方法是分配两个独立线程并随机执行Web服务响应。但是，这两个服务请求是存在优先级之分的，用户*B*的请求应该优先于用户*A*的请求执行，以避免用户*A*查询到摄像头空闲而请求操作时摄像头已被占用的矛盾。

在传统的Web服务并发控制基础上，本文将在解析出SOAP Body的内容后，设计新增的相应的并发控制模型实现Web服务并发控制方法。

本文首先从SOAP协议入手，在SOAP Body中定义服务设备资源统一描述框架，从而为下一步的优先级计算和并发控制奠定基础。如表1所示。

采用上述8个属性对服务设备资源进行描述。其中，ResourceURL、OperationName和OperationInput、OperationOutput取自SOAP本身，另外定义了ResourceID、ResourceType、ResourceList和OperationType等4个属性。传感器网络Web服务高度依赖底层传感器设备，因此需要唯一的ResourceID来标识服务设备资源，而传感设备的高度互联，同一传感器可能被多个Web服务调用，不同传感器设备也可以协同工作实现同一Web服务。本文采用ResourceList表示Web服务所关联的多个服务设备资源，是一个由各服务设备资源ResourceID组成的集合。ResourceType指明服务设备资源的类型，如摄像头、加湿器等，显然，一个ResourceType可以对应多个ResourceID。OperationType指明可以对服务设备资源进行的操作类型，如查询、操作。OperationInput和OperationOutput分别是服务设备资源操作的输入和输出，从服务I/O中获取。例如，查询摄像头是否空闲的服务请求可以按照服务设备资源属性定义统一刻画。由于SOAP是基于XML描述的，因此只要增加解析处理，自定义的设备属性标签〈ResourceList〉和〈Operation〉等可以很容易地嵌入SOAP。图5所示为使用服务设备资源统一描述框架描述的SOAP请求消息。

表1 服务设备资源属性定义

属性名	说明
ResourceID	服务设备资源ID
ResourceType	服务设备资源类型
ResourceList	同属一个服务的设备资源列表
ResourceURL	设备资源所属服务的访问地址
OperationName	服务设备资源的操作名称
OperationType	服务设备资源的操作类型
OperationInput	服务设备资源操作的输入变量
OperationOutput	服务设备资源操作的输出变量

使用服务设备资源统一描述框架对SOAP请求消息进行扩展，可增加服务设备资源信息。在解析出SOAP Body中的服务请求内容后，消息即被送到并发控制模块，此时先前的线程释放Web服务请求实例并放回第一线程池。通过新增并发控制模块，本文提出改进的并发控制模型，如图6所示。与传统的Web服务并发控制流程不同，该并发控制维护两个线程池。并发控制模块接收到多线程送来的Web服务请求消息，根据一定的策略进行请求服务设备资源的重复和冲突检测，优化服务请求内容及其执行顺序，由第二线程池分配线程通过执行模块完成相应的服务请求。通过并发控制模块，可避免线程浪费、资源冲突和错误的发生。并发控制模块内部组成如图7所示。

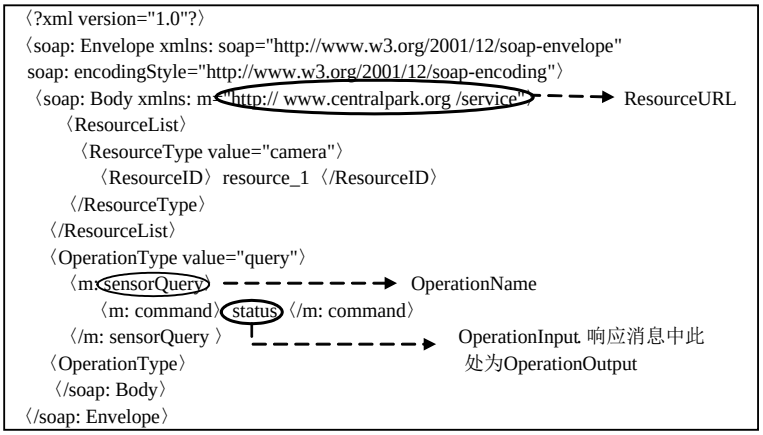

图5　SOAP请求消息格式

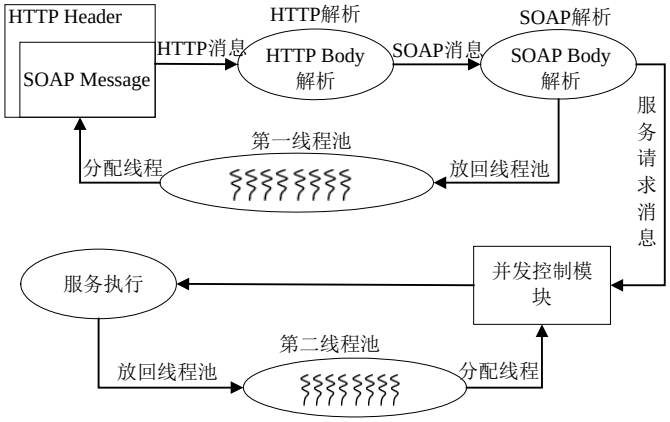

图6　服务并发控制原理

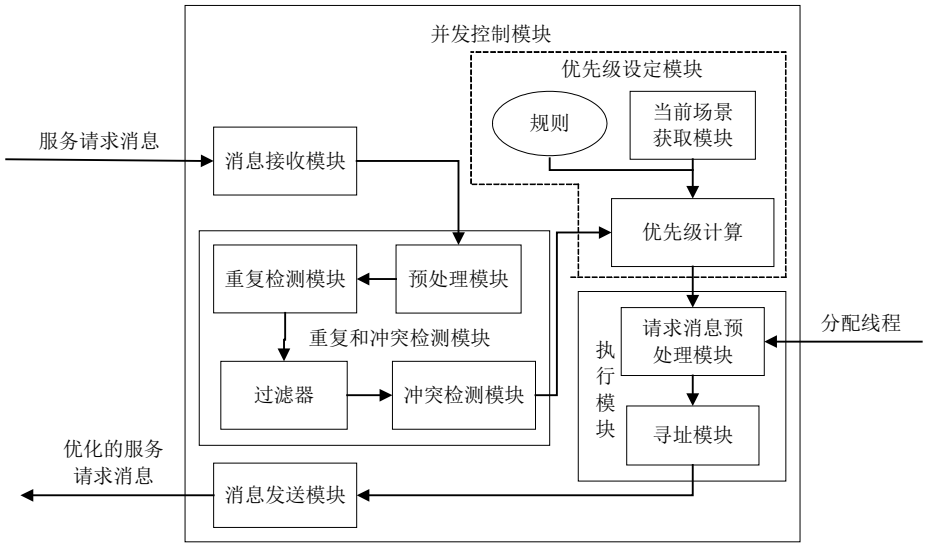

图7　并发控制模块结构

并发控制模块主要包括重复和冲突检测、优先级设定和执行模块。消息接收模块负责接收解析的SOAP服务请求消息，并将请求消息送到预处理模块。预处理模块用于将解析得到的信息进行数学表示。首先，利用重复检测方法进行重复检测，使用过滤器过滤掉重复的服务请求，并送到冲突检测模块进行服务请求冲突检测。若存在冲突，需要对冲突的服务请求赋予不同的优先级，优先级的设定依靠预先定义的规则及当前的场景。请求消息预处理模块对标记优先级的请求进行排序，分配线程，并进一步使用寻址模块通过消息发送模块执行服务请求。

3.1　重复和冲突检测

Web服务并发控制的出发点是避免请求的重复与冲突。若服务请求存在重复，则可以使用相同的

线程完成相应的请求内容；若服务请求存在冲突，则需要对冲突的服务请求线程赋予不同的优先级。以下首先将重复和冲突检测中用到的服务设备资源进行相应的数学定义和描述，接着给出具体的检测计算方法。

服务器收到SOAP请求后将通过解析获得各个具体属性的取值。S服务的设备资源列表ResourceList可定义为$RL(S)=\{RI_1,RI_2,\cdots,RI_N\}$，其中$RI_i$为设备编号，且属于自然数集合$N$。设备资源RI的设备操作类型可定义为$OT(RI)=\{OT_1,OT_2,\cdots,OT_p\}$，其中$OT_1$定义的查询操作为客户端发送请求并需要等待服务器端的响应；OT_2定义的操作为客户端发送请求后，除了确认信息外，不需要服务器端返回其他信息。由于目前该文提出的方法支持的仅有查询(query)和操作(do)两种操作类型，因此OT_1='query'，OT_2='do'，p=1，2。服务S的访问地址URL可定义为$URL(S)=L$，其中L为字符串变量。同理，服务设备资源RI的操作名称OperationName可定义为$ON(RI)=p$，其中p为字符串变量，且p=1,2。服务设备资源操作的输入OperationInput可定义为$OI(RI)=\{OI_1,OI_2,\cdots,OI_M\}$，其中$OI_i$为输入变量。

服务请求的重复检测需要通过比较两个服务请求的访问地址URL、OperationName和OperationInput来判断，若全部相同则表示两个服务请求是重复的；若至少有一项不同，则两个服务请求是不同的。假设有两个服务请求R_1和R_2同时到达，通过解析可以得到R_1的服务访问地址为$R_1\cdot URL(S_1)$，服务设备资源的操作名称OperationName为$R_1\cdot ON(RI_1)$，服务设备资源的操作输入OperationInput为$R_1\cdot OI(RI_1)$；同样R_2的服务访问地址为$R_2\cdot URL(S_2)$，OperationName为$R_2\cdot ON(RI_2)$，OperationInput为$R_2\cdot OI(RI_2)$。则判断服务请求重复的条件表达式如下：

$$\{R_1\cdot URL(S_1)=R_2\cdot URL(S_2)\}\&\&\{R_1\cdot ON(RI_1)=R_2\cdot ON(RI_2)\}\&\&\{R_1\cdot OI(RI_1)=R_2\cdot OI(RI_2)\}$$

表达式为真则说明服务请求重复。

服务请求的冲突检测不仅涉及两个服务请求关联的设备资源是否有重复，而且依赖服务请求对设备资源的操作类型。假设有两个服务请求R_1、R_2，关联的设备资源列表分别表示为$R_1\cdot RL(S_1)$、$R_2\cdot RL(S_2)$，首先服务请求存在冲突的前提条件为$RL(S)=R_1\cdot RL(S_1)\cap R_2\cdot RL(S_2)\neq\varnothing$，即两个服务请求$R_1$，$R_2$关联有相同的服务设备资源，因此存在冲突的可能。其次，服务请求R_1、R_2对其关联的相同服务设备资源的不同操作类型决定了冲突是否存在。判断依据如下：若R_1、R_2对其关联的某一相同设备资源操作类型同为查询，则不存在冲突；若$R_1(R_2)$为查询、$R_2(R_1)$为操作或R_1、R_2同为操作则存在冲突。判断条件如下：

$$\text{if}\{!(R_1\cdot OT(RI_i)=R_2\cdot OT(RI_i)=\{OT_1\})\}\ \text{then}\{\text{conflict=true}\}$$

其中，$RI_i\in RL(S)$。

3.2 优先级计算和服务执行

对于检测到服务请求冲突的情况，需要进行优先级计算，并对冲突的服务请求赋予不同的优先级。与其他优先级计算方法不同的是，本文方法中优先级计算由静态优先级计算和动态优先级计算两部分组成，其中静态优先级的计算又综合考虑了规则和场景两个因素，从而实现了上下文感知和动静结合两大特色功能。

静态优先级是指优先级的计算值将不随时间变化而改变，主要考虑利用预先定义的规则和场景计算服务请求的优先级。采用ECA规则(事件−条件−动作)预先指定部分可能冲突的优先级，其中E刻画触发该规则的冲突事件，C刻画规则条件中包含的服务请求，A设定冲突服务请求的优先级。当有两个服务请求R_1、R_2到达，且R_1、R_2均对RI_1表示的摄像头进行了操作，由冲突检测可知服务请求R_1与R_2冲突，由此触发规则r_1。规则r_1的条件部分表示当R_1的操作名称为“turn to east”，而R_2的操作名称为“turn to south”，且由当前场景获取模块得知当前摄像头的方位为north时，出于减少摄像头操作考虑，在规则r_1的动作部分，设定R_1的优先级为10，高于R_2的优先级，其中优先级值属于自然数集合N。事实上，在Web服务中，某个服务请求可能在一个场景下优先级较低，但在另一场景下优先级最高。因此，在规则定义中引入当前场景使得优先级的设定更加准确。

前面提到，目前该文提出的方法支持的服务设备资源操作类型仅有查询(query)和操作(do)两种，冲突检测依据的也是不同服务请求对服务设备资源的操作类型。假设有两个服务请求R_1和R_2，对于服务请求冲突的一种情况，即对服务设备资源的操作类型$R_1(R_2)$为查询、$R_2(R_1)$为操作，表示为$R_1\cdot$OT (RI)= ={OT_1}，且$R_2\cdot$OT (R_1)= ={OT_2}，利用规则设定操作的优先级永远高于查询的优先级，即do·Priority > query·Priority；对于服务请求冲突的另一种情况，即对服务设备资源的操作类型R_1和R_2均为操作，表示为$R_1\cdot$OT (RI)= =$R_2\cdot$OT (RI)= ={OT_2}。优先级的计算

仅使用规则是不够的，还需要综合静态优先级和动态优先级。

从规则中，可以直接得到预先设定的优先级取值，表示为X。由于ECA规则和场景是预先定义的，因此无法考虑服务执行时间对静态优先级的影响。若有一组冲突的服务请求R_1，R_2，…，R_N，按执行时间由长到短排序，可以得到一组序列R_T={1,2,…,N}。假设冲突服务请求R_S($1\leqslant S\leqslant N$)在R_T序列中的序号为Y($1\leqslant Y\leqslant N$)，则静态优先级计算公式可表示为$P_{\text{static}}=pX+qY$，($p+q=1$)。其中，p表示规则预定义优先级数值的权重，而q表示执行时间对静态优先级的计算权重。

动态优先级是指优先级随服务请求在队列中的等待时间而动态变化，即随等待时间增加优先级逐渐提升。系统通过不断更新服务的等待时间，按从小到大顺序产生一组等待时间序列R_W={1,2,…,M}。与执行时间序列不同的是，R_W是每隔固定间隔更新的。假设冲突服务请求R_S($1\leqslant S\leqslant M$)在R_W序列中的序号为Z($1\leqslant Z\leqslant M$)，则动态优先级计算公式为$P_{\text{dynamic}}=Z$。

综合考虑静态和动态优先级，得到统一的优先级计算公式：

$$p=\alpha P_{\text{static}}+\beta P_{\text{dynamic}}$$

式中，α和β分别为静态和动态优先级对整个优先级计算的权重，$\alpha+\beta=1$。

服务请求存在重复和冲突两种情况。并发控制模型需要对这两种情况设定不同的策略。对于服务请求重复的情况，可以通过共用线程来减少不必要的线程使用，首先通过过滤器将重复的服务请求转化为一个服务请求，然后第二线程池分配线程，执行完毕后将响应结果返回给重复服务请求组内的每一个源地址。对于服务请求冲突的情况，按照本文提出的优先级计算公式，对冲突的服务请求赋予不同优先级，按优先级先后顺序执行。总之，并发控制模块收到一系列服务请求后首先会对其进行排序。排序整体按照先来先服务的原则，但对于存在冲突的服务请求按照优先级高低顺序排序。之后，服务请求被依次放入执行队列，通过寻址模块由第二线程池分配线程执行服务请求。

4　实验与案例分析

本文按服务构建与并发控制两部分对提出的服务管理方法进行实验与案例分析。通过提供一个入侵检测的应用场景，首先在服务构建实验与案例分析部分，分析规则引擎匹配性能、上下文感知优先级在规则选择上的应用，以及基于推理的上下文感知工作流如何实现传感器网络的资源重用。其次在并发控制案例分析部分，分析改进的并发控制方法在减少不必要的服务执行次数及冲突请求消除上，相对传统并发控制方法在资源受限的传感器网络中应用的优势。

4.1　服务构建实验与案例分析

首先，给出一个应用场景。一个家庭购买了多个传感器用于构建入侵检测服务，包括红外传感器和声音传感器，如图8所示。传统的工作流模型中，家庭男主人使用红外和声音传感器构建了工作流H，而女主人使用声音传感器构建了工作流W。其实，工作流H和工作流W含有相同的语义信息，即入侵是否发生。因此，面向传感器网络的服务管理模型引入推理抽取出该语义信息，建立新的原子服务(入侵探测)替代已有的原子服务(红外探测和声音探测)。尽管男女主人使用不同的传感器组合，构建工作流时可以共用同一原子服务，可实现原子服务的重用，构建的工作流即工作流R。

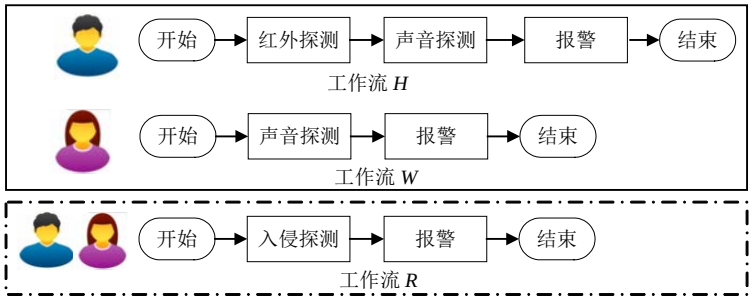

图8　基于推理上下文感知工作流实现资源重用

为了获取传感数据中的语义信息，首先需要为男女主人建立一些规则文件。其中，男主人对应规则H，女主人对应规则W。假设当前传感数据为：红外探测到周围有人，并且声音传感器数值为750。那么，规则W将被选择执行，输出存在入侵检测。如考虑春节场景，人们会燃放鞭炮，产生巨大声响，

导致入侵检测系统误报警。因此，领域专家建立规则*E*来解决该问题。规则*E*具有较低的默认优先级5和上下文感知优先级10。同样在春节场景下，当传感器数据为：红外探测到周围有人，并且声音传感器数值为850，规则*H*、规则*W*和规则*E*同时匹配，冲突产生，但规则*H*的优先级为6，规则*W*的优先级为7，而规则*E*的上下文感知优先级为10，由规则冲突消除算法可知，规则*E*会被选择执行，即入侵检测并没有发生，由此避免了误报警的发生。入侵检测服务规则文件如下：

rule name “rule *H*”
attribute: always
priority: 6(default)
object:
(boolean) infrared.status = whether some people is around or not;
(int) sound intensity = the environment sound intensity;
if: (infrared.status= =true) and (sound intensity) > 800);
then: intrusion.status = true;
end

rule name “rule *W*”
attribute: always
priority: 7(default)
object:
(int) sound intensity = the environment sound intensity;
if: (sound intensity) > 700);
then: intrusion.status = true;
end

rule name “rule *E*”
attribute: always
priority: 5(default), 10(Chinese New Year)
object:
(int)sound intensity = the environment sound intensity;
if: (infrared.status= =true) and (sound intensity) < 900);
then: intrusion.status = false;
end

表2 规则列表

规则	入侵检测条件	入侵状态	优先级
rule *H*	infrared.status=true and sound intensity>800	true	6(default)
rule *W*	sound intensity>700	true	7(default)
rule *S*	infrared.status=true or light>800	true	4(default)
rule E_1	infrared.status=true and sound intensity<900	false	5(default), 10(Chinese New Year)
rule E_2	sound intensity>850 and light<1 000	false	5(default), 9(Chinese New Year)

表3 传感器数据列表

传感器数据组合	描 述
composition1	infrared.status=true, light=930, sound intensity=650
composition2	infrared.status=false, light=1050, sound intensity=850
composition3	infrared.status=true, light=920, sound intensity=750
composition4	infrared.status=true, light=700, sound intensity=670
composition5	infrared.status=true, light=950, sound intensity=780

基于推理上下文感知工作流较传统工作流增加了规则推理部分，因此增加了额外的计算时间。本文扩展了应用场景，增加了规则*S*和规则E_1、规则E_2，如表2所示，其中规则E_1、规则E_2为领域专家为解决潜在的冲突而建立的规则，本文假定领域专家定义的规则为最优规则。与传感器数据数值的关系如图9所示。增加的规则推理部分在规则匹配过程的响应时间，实验结果显示，规则匹配所增加的额外计算时间是可以接受的。

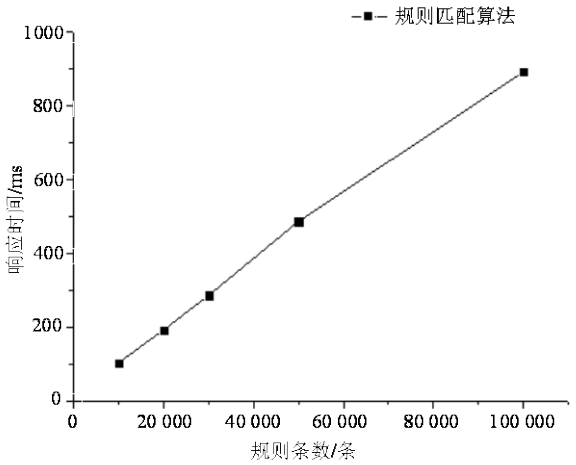

图9 规则匹配性能分析

为了验证基于上下文感知优先级在规则冲突解决中较高的准确性，选择表3所示的一组传感器数据进行规则匹配和冲突规则选择，结果如表4所示。可见基于上下文感知的优先级较传统优先级在冲突规则的选择上具有更高的准确率。

表4　基于上下文感知和传统优先级准确性比较

传感器数据组成	传统优先级	上下文感知优先级	当前上下文
composition1	1	1	Chinese New Year
composition2	1	1	\
composition3	0	1	Chinese New Year
composition4	1	1	\
composition5	0	1	Chinese New Year
precision/(%)	60	100	\

4.2　并发控制案例分析

在服务请求方面，本文的并发控制方法的出发点是优化服务请求，避免重复请求相同的服务和消除服务请求的冲突。同样以入侵检测服务为例，房间有成员*A*、*B*、*C*、*D*，利用摄像头camera_1、camera_2、camera_3，以及红外传感器和声音传感器组成的入侵检测服务(工作流)。入侵检测服务提供用户查询当前是否存在入侵，以及提供用户使用摄像头进行视频监控。

表5　入侵检测服务请求列表

服务请求	入侵检测服务请求
R_1	家庭成员*A*请求入侵检测服务以获悉房间是否发生入侵检测
R_2	家庭成员*B*请求入侵检测服务以获悉房间是否发生入侵检测
R_3	家庭成员*C*请求操作摄像头camera_1进行视频监控
R_4	家庭成员*D*请求查询是否有摄像头空闲

假设有表5所示服务请求同时到达，传统的Web服务并发会分别执行表中的服务请求。但本文提出的Web服务请求并发控制方法会在解析出具体服务请求内容后，执行重复和冲突检测，并按照既定策略执行服务。

由于家庭成员*A*、*B*请求同一房间的入侵检测服务，则$S_1=S_2$，R_1·URL(S_1)= R_2·URL(S_2)。另外，R_1、R_2调用入侵检测服务的同一API，即R_1.ON(S_1)= R_2·ON(S_2)=“Status query”，R_1·OI(S_1)=R_2·OI(S_2)= “intrusion”。由服务请求重复判断条件可得R_1、R_2为重复的服务请求。

对于服务请求R_3、R_4来说，R_3·RL (S_3)={camera_1}，R_4·RL (S_4)={camera_1，camera_2，camera_3}，R_3·RL (S_3)$\cap$R$_4$·RL (S_4)$\neq\varnothing$，满足服务请求冲突的前提条件，即R_3、R_4服务请求关联了相同的设备资源(camera_1)。此外，R_3·OT (camera_1)= “do”，R_4·OT(camera_1，camera_2，camera_3)= “query”，由服务请求冲突判断条件可得服务请求R_3、R_4存在冲突。

并发控制流程为：1) 重复检测模块检测到R_1、R_2为重复的服务请求后，会合并服务请求，入侵检测服务执行结束后向家庭成员*A*、*B*返回执行结果，避免了不必要的服务执行。2) 冲突检测模块检测到R_3、R_4为冲突的服务请求后，会进行优先级计算，R_3·OT(camera_1)=“do”，R_4·OT(camera_1，camera_2，camera_3)=“query”。因此，R_3具有更高的优先级，会优先执行，避免了错误的发生。入侵检测服务中改进的并发控制与传统的并发控制的比较如表6所示。可见，改进后的并发控制方法减少了不必要的服务执行次数，并避免了服务执行中的冲突发生，更加适应资源受限的传感器网络的服务执行。

表6　改进的并发控制与传统的并发控制的比较

方法	服务执行次数	冲突消除
传统的并发控制方法	4	否
改进的并发控制方法	3	是

5　结　束　语

由于传感器网络能量、通信能力、计算存储能力有限，基于传感器网络的服务管理需要强调传感器网络的资源重用，避免不必要的资源浪费。本文提出的面向传感器网络的服务管理方法，使用基于推理的上下文感知工作流构建服务，并改进服务请求并发控制，实现了资源的重用，更加适应资源受限的传感器网络。无线传感器网络作为复杂网络在工程技术领域的典型应用，其研究对于复杂网络的研究将会发挥一定的推动作用。

参　考　文　献

[1] STROGATZ S H. Exploring complex networks[J]. Nature, 2001, 410: 268-276.

[2] NEWMAN M E J. The structure and function of complex networks[J]. SIAM Review, 2003, 45:167-256.

[3] CHANG S S, YOUNG-BAE K. Design and implementation of intelligent home control systems based on active sensor networks[J]. IEEE Trans on Consumer Electronics, 2008, 3(54): 1177-1184.

[4] CHUN Ou-yang, ROSA M L, HOFSTEDE A H M, et al. Toward web-scale workflows for film production[J]. IEEE Internet Computing, 2008, 5(12): 53-61.

[5] LIMA G H E L, SILVA L C , NETO P F R. WSN as a tool for supporting agriculture in the precision irrigation[C]//Proc of IEEE ICNS’2010. Cancun: IEEE, 2010: 137-142.

[6] WANG Xue, WANG Sheng, BI Dao-wei. Distributed visual-target surveillance system in wireless sensor networks[J]. IEEE Trans on Systems, Man, and Cybernetics, Part B: Cybernetics, 2009, 5(39): 1134-1146.

[7] DELICATO F, PIRES P, PIRMEZ L, et al. A flexible middleware system for wireless sensor networks[C]//Proc of

ACM/IFIP/USENIX Middleware'2003. New York：ACM，2003: 474-492.

[8] LEGUAY J, LOPEZ-RAMOS M, JEAN-MARIE K， et al. An efficient service oriented architecture for heterogeneous and dynamic wireless sensor networks[C]//Proc of LCN'2008. Montreal：IEEE，2008: 740-747.

[9] NIU Wen-jia, LI Gang, ZHAO Zhi-jun, et al. Multi-granularity context model for dynamic Web service composition[J]. Journal of Network and Computer Applications, 2011, 1(34)：312-326。

[10] LUH Y, PAN C. Collaborative workflow solution for distributed product development[C]//Proc of IEEE CSCWD'2008. Xi'an: IEEE, 2008: 594-599.

[11] TSAI M. The workflow development for electronic manufacturing industry[C]//Proc of IEEE CSCWD'2008. Xi'an: IEEE, 2008: 688-692.

[12] WEISER M. Ubiquitous Computing[J]. Computer, 1993, 10(26): 71-72.

[13] WIELAND M, KACZMARCZYK P, NICKLAS D. Context integration for smart workflows[C]//Proc of PerCom'2008. Hong Kong: [s.n.], 2008: 239-242.

[14] CHAARI T, EJIGU D, LAFOREST F, et al. A comprehensive approach to model and use context for adapting applications in pervasive environments[J]. Journal of Systems and Software, 2007, 12(80): 1973-1992.

[15] MODAFFERI S, BENATALLAH B, CASATI F, et al. A methodology for designing and managing context-aware workflows[M]. International Federation for Information Processing, Mobile Information Systems II. NewYork: Springer, 2005: 91-106.

[16] ARDISSONO L, FURNARI R, GOY A, et al. A framework for the management of context-aware workflow systems[C]//Proc of WEBIST'2007. Noordwijkerhout：ACM, 2007: 1-9.

[17] JONGSUN C, YONGYUN C, JAEYOUNG C. A context-aware workflow system supporting a user's dynamic service demands in ubiquitous environments[C]//Proc of MUE'2007. Seoul: IEEE, 2007: 425-430.

[18] Kang DongHyun, Han Saeyoung, Yoo SeoHee, et al. Prediction-based dynamic thread pool scheme for efficient resource usage[C]//Proc of CIT Workshops'2008. Sydney: IEEE, 2008: 159-164.

[19] WANG Hao, TONG Yi-zhu, LIU Hong, et al. Application-aware interface for SOAP communication in web services[C]//Proc of CLUSTR'2006. Barcelona: IEEE, 2006: 1-8.

[20] CHARLES F. Rete: A fast algorithm for the many pattern/many object pattern match problem[J]. Artificial Intelligences, 1982, 1(19): 17-37.

[21] DING Xiao, ZHONG Xiao-an. Improving rete algorithm to enhance performance of rule engine systems[C]//Proc of ICCDA'2010. Qinhuangdao: IEEE, 2010: 572-575.

[22] GUDGIN M, HADLEY M, MENDELSOHN N, et al. SOAP version 1.2 Part 1: messaging framework (2nd ed). [EB/OL][2010-02-02]. http://www.w3.org/TR/soap12-part1/.

编 辑 蒋 晓