基于微服务架构的物联网中间件设计

吴斌烽
(浙江工业大学计算机科学与技术学院 杭州 310000)

摘 要 以传统 SOA 架构搭建的物联网系统在扩展能力和异构设备持续集成方面存在不足,且随着物联网生态系统概念的成熟,如今物联网更强调与第三方物联网系统间的互操作支持能力。为解决上述问题,提出了基于微服务架构的物联网中间件,阐述了其内部组成和功能目标,详细研究了其中异构设备的服务抽象过程以及多用户环境下的冲突解决机制。通过微服务架构的灵活性和服务间的松耦合特点,提出的物联网中间件除了能保证异构设备的动态集成和统一的服务化抽象外,还能有效支持第三方物联网系统的接入。最后通过实例验证了该中间件设计的可行性。

关键词 物联网,微服务,中间件,异构性,互操作性,可扩展性
中图法分类号 TP393 **文献标识码** A

Design of IoT Middleware Based on Microservices Architecture

WU Bin-feng
(College of Computer Science & Technology, Zhejiang University of Technology, Hangzhou 310000, China)

Abstract IoT (Internet of Things) systems based on traditional SOA (Service-Oriented Architecture) are poor in scalability and are hard to support heterogeneous devices with continuous integration. Moreover, IoT platforms' interoperability with third-party becomes crucial as the IoT ecosystem enhances. Thus this paper proposed a general IoT middleware based on microservices architecture to solve the problems mentioned above, throughoutly researched the internal components and their effects, especially studied the service abstraction process of heterogeneous devices and the conflict resolution mechanism in multi-user enviroment in detail. Through the flexibility of the microservice architecture and the loose coupling between services, not only heterogeneous devices but also third-party IoT systems can be integrated at runtime as services. In the last place, actual devices are used to verify the applicability of this middleware.

Keywords Internet of things, Microservices, Middleware, Heterogeneity, Interoperability, Scalability

1 引言

随着工业 4.0 时代的到来,物联网成为近年来工业、学术界研究的热点。物联网的设计意图是承担更多的感知执行设备的接入和交互任务,实现物理设备的互联互通,将原本一对一的信息隧道模式变成多对多的互联网模式。

但物联网的落地过程一直存在难点:不仅有海量设备带来的连接压力,更突出的是物联网设备的异构问题,具体表现为各物联网设备在支持的通信协议、依赖的传输网络、数据的解析和处理分级 3 个方面截然不同,无法通过统一的框架、协议解决所有的连接和控制问题。同时,随着硬件技术的成熟和物联网设备的广泛部署,物联网生态系统的概念愈发成熟,如今物联网系统的设计重心也由强调单个物联网系统的处理能力转向追求无限扩展性以及与第三方物联网系统之间良好的互操作支持能力。

为解决上述问题,基于中控集团浙江浙大信息技术有限公司的物联网平台的研发经历,设计了基于微服务架构的物联网中间件 SupconIoTP。其内部实现了彻底的服务化,借助微服务架构服务独立的特点,能针对异构物联网设备、物联网网关以及第三方物联网系统提供独特的接入微服务,提高了系统的可扩展性和互操作性。此外,还设计有动态的服务组合和多用户场景下的服务冲突解决机制,增强了系统灵活性和服务可靠性。

下文将首先介绍基于面向服务架构以及微服务架构的物联网中间件的相关研究成果,接着展示本中间件的内部架构组成以及其中各服务组件的功能,然后对中间件中的关键实现部分进行详细的技术描述。随后在验证环节,基于实例验证异构设备的接入以及多用户场景下的冲突解决。最后总结本中间件的优缺点,并展望未来的工作方向。

2 相关研究

2.1 面向服务架构中间件

陈海明等[1-2]总结了面向服务的物联网中间件的一般组织形式,并根据从物端资源中抽象出服务的软件构件的位置不同将服务分为物理、虚拟实体服务。中间件层或收集物理实体服务,或创建虚拟实体服务,针对这些实体服务进行服务的注册、发现和组合,并提供给编程接口供应用层调用。

目前已有不少基于 SOA 的物联网中间件研究成果,如 Physicalnet[3],Physical Mashup[4],两者以物理实体服务为主,存在相似的问题,即在部署应用时使用的服务以及服务组合无法动态变化,如此构建的应用在可扩展性上差强人意。WInternet[5]设计了独有的软件功能模块——网件,负责设备或网关数据的处理并以虚拟实体服务的形式发布,网件之间

吴斌烽(1994—),男,硕士生,CCF 会员,主要研究方向为物联网,E-mail:wubf.george@foxmail.com(通信作者)。

能通过发现协议使用其他网件提供的服务。但上述 3 种中间件均缺少对服务的有效管理,无法保证服务可靠性。Xu 等[6-7]基于 OSGI 基础框架,设计了设备集成中间件、边缘层中间件、云端中间件 3 层架构;通过设备集成中间件预先烧录的驱动程序实现设备接入,但整个系统包含软硬件开发,复杂度高,且被限制在同个技术栈和开发标准中,对后续新技术、新需求的引入造成了阻碍,系统缺乏可扩展性。

基于 SOA 的物联网中间件由于本身依赖于企业服务总线 ESB 的设计实现,大都存在复杂度高、开发和维护成本居高不下的情况。其中提供实体服务的软件构件如网关等,可复用性、可维护性较弱,这也限制了系统对异构设备的持续集成和动态扩展。

2.2　微服务架构中间件

微服务架构[8]是目前构建分布式互联网应用的流行选择。与 SOA 相比,基于微服务架构的系统内部组件化和服务化更彻底,且各个微服务部署于独立的操作系统进程中,因此可以基于不同技术栈实现,微服务之间使用轻量级的通信协议取代 ESB,实现了真正的服务解耦,这是微服务架构在解决物联网设备异构问题上与生俱来的优势。且以微服务架构搭建的物联网系统的复杂度更低,扩展能力更强。同时使用微服务架构时强制要求缩小服务粒度来提高系统的并发能力。这些特点都契合物联网应用快速部署、演化式扩展的要求。

因此,许多研究者开始关注微服务架构与物联网中间件的结合。比如,Namiot 等[9]总结了微服务架构的优缺点和设计原则,并验证了其在 M2M 应用中的可行性。彭昭[10]总结了基于微服务架构的物联网中间件必须具备的四大核心功能,以及实现过程中涉及的关键技术。Vresk 等[11]关注物联网设备的异构特点,设计了独特的数据模型和寻址模型来建立虚拟设备,并基于虚拟设备提供事件消息,以事件消息进行服务交互。Krylovskiy 等[12]展示了智慧城市平台 DIMMER 的初期设计,体现出微服务架构良好的可扩展特点,强调平台中物联网服务与其他系统提供的云服务之间的互操作。Bak 等[13]则针对移动应用设计了基于地理位置和上下文信息的特殊微服务,展示了微服务在物联网应用中的无限可能性。Sun 等[14]设计了通用型物联网中间件,其能动态加载通信协议相关插件的微服务,有效屏蔽设备在通信协议方面的异构问题。

但上述诸多研究都未充分考虑物联网设备在支持的通信协议、依赖的传输网络、数据的解析和处理分级 3 个方面均存在不同,不能一概而论,且不够重视抽象化的服务和设备请求之间的差异,对多用户场景下的服务冲突问题以及第三方物联网系统的互操作支持都缺少详细的考虑。

3　系统架构及构建目标

物联网设备的异构特点决定了想要使用统一的框架和协议解决设备的接入问题是不现实的,对于异构设备应当提供相应的接入方式和数据处理。因此,物联网中间件应当具备能随设备的多样性发展而不断迭代更新持续扩展的能力。

此外,随着物联网生态系统的成熟,物联网在不同领域不断扩展和细分,针对各大领域设计专用物联网中间件势在必行。如此一来,跨行业应用的通用型中间件更需强调与其他物联网系统之间的互操作能力。

而微服务架构以其突出的开发周期短、部署速度快、扩展更灵活、内部组件可独立实现的特点取代 SOA 成为了物联网中间件的转型的架构首选。

3.1　总体架构

本文设计的中间件的总体架构如图 1 所示。

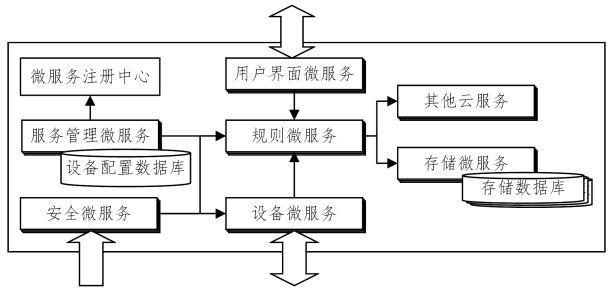


图 1　中间件架构图

中间件主要由安全微服务、用户界面微服务、存储微服务、服务管理微服务、规则微服务、设备微服务组成。各个微服务又包含多个实例,在容器技术的帮助下,可实现快速部署和水平扩展。

3.2　各服务构建目标

1)安全微服务

负责设备和用户的验证、授权以及用户组管理。依据终端性能和场景安全等级要求,可以提供从简单的预分配秘钥到更为复杂的单向/双向验证的不同级别安全认证服务。另外,还有部分特殊实例一直保持启动状态,负责内部微服务的性能指标管理、健康状况监控以及报警信息记录。

2)用户界面微服务

主动访问所涉及的内部 API 或订阅相关规则微服务发布的主题,负责将状态同步到 UI 控件中。在中间件中构建的应用可以在 Web 终端里组装预期界面,用户界面微服务将负责展示实时信息。

3)存储微服务

存储经由规则微服务处理过后的数据,所有涉及存储数据的操作均通过存储微服务。这是因为物联网设备收集的数据各具特点,需要使用不同类型的数据库存储。出于对可扩展及解耦的考虑,专门建立了存储微服务来负责数据存储读取操作。

4)服务管理微服务

管理设备微服务实例的生命周期。在设备微服务实例创建前验证用户是否具备创建权限,随后根据设备描述文件选择设备模板,创建设备微服务实例。同时作为第三方注册器注册和注销设备微服务到服务注册处(service registry),也承担服务发现职责,用户每次在查询可用服务时,都将经过服务管理微服务的权限检查,唯有开放相应访问权限的微服务会被返回相应信息。

5)规则微服务

负责微服务之间的服务组合。采用订阅/发布的方式实现一对多通信,能与多个设备微服务、其他规则微服务、存储微服务以及其他云服务如大数据分析云服务等交互,是物联网应用与中间件内部微服务之间的交互入口。同时,规则微服务也具备一定的数据处理能力,可以通过制定规则,动态地构建起预期的数据处理,实现对数据的简单聚合、筛选。利用规则微服务,可以组合多个微服务,实现设备的自动反控、数据去冗余、实时报警等诸多功能。

6)设备微服务

由服务管理微服务管理生命周期,根据设备描述文件创建,负责与物端的通信工作,能依据创建者设置的逻辑顺序对数据进行初步处理并提取有效信息。其中与物理设备的通信细节实现对中间件内部服务调用方来说完全透明,以统一形式提供由设备抽象出的服务。

4 关键服务设计

借由微服务架构的服务独立部署开发特点,可以保证物联网系统为异构设备提供不同的接入微服务。但如何动态地将异构设备服务化,如何解决设备服务化后带来的服务调用冲突,具体应用又如何有效组合服务,都依赖于具体的微服务组件设计。

4.1 设备微服务

设备微服务负责从设备到服务的抽象过程以及服务调用与设备请求之间的相互转化。要完成上述职责,首先需要一种统一的从设备中抽象出服务的抽象机制,该机制要求能完全屏蔽设备在通信协议、传输网络、数据处理和分级方面的异构特点,其次还需要处理服务调用与真实设备请求之间在可靠性、并发性支持、冲突控制方面的差异。

在本文的设备微服务设计中,采用设备描述机制、设备模板、设备映像来满足上述要求。

4.1.1 设备描述机制

图 2 给出了创建设备微服务并实现自动反控功能的时序图。

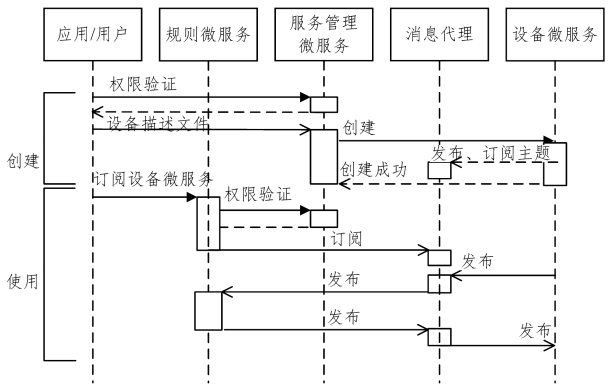


图 2 设备微服务创建使用时序图

表 1 设备描述文件配置表

配置项	含义
DeviceID/ProductID	设备/产品线标识
DeviceKey/ProductKey	设备/产品线密钥
DeviceProtocol	设备通信协议
DeviceURL	设备地址
DeviceType	设备类型
AttributeType	属性类型
AttributeName	属性名
AccessPermission	属性衍生服务的访问权限
OperationURL	属性的访问地址
OperationTrigger	返回判断触发条件
OperationReturnType	返回结果类型,决定如何解码
OperationReturn	返回结果的处理函数
OperationInterval	单次访问间隔
OperationPolling	轮询间隔
OperationSave	是否以队列形式保存请求
OperationParmEnable	是否包含参数

从图 1 中可以看到设备微服务依据设备拥有者提供的设备描述文件创建,以订阅/发布的形式对外提供服务。设备描述文件包含设备在通信协议、传输网络、数据的解析和处理 3 个方面的异构特点描述以及最终对外暴露服务的具体内容。设备描述文件中主要的配置项如表 1 所列。

在设备描述文件中直接指明设备使用的通信协议,在创建时使用预定义的通信模板构建设备微服务。

根据应用实际需求不同,物联网设备会选择合适的传输网络,比如智能家具设备通常部署于无线局域网中,而智能水电表往往使用 NB-IoT 等低功耗广域网技术连接至中间件。不同的传输网络在功耗、带宽、连接数、服务质量方面都大相径庭,但在应用层看来这些都可以视为对设备访问频率的限制,因此在描述文件中提供了对属性的单次访问间隔的配置选择。

在描述文件中还可以设置属性的访问地址以及返回消息的触发判断、编码类型、信息提取。利用这种可配置的解析方式可实现对异构数据的个性化解析和处理。

此外,考虑到采集数据的敏感性和使用控制接口的专业性要求,以及安全性,在设备描述文件里还引入了对属性的访问权限控制,借此设备拥有者可以实现对细粒度的访问控制。同时,因为描述文件中的属性只是现实中物理属性的抽象,与现实属性并非一一对应关系,因此设备拥有者可以自由创建多条虚拟的属性指向同一设备访问地址,但数据处理方式、信息暴露程度均不相同,以此可以达到针对特定群体展示不同信息的目的。

设备描述机制的作用是从设备中抽象生成服务,即以面向对象的思维方式将物联网设备视为服务提供对象,自然而然,设备采集和控制的物理世界中的各个参数成为了设备本身的属性,各感知控制就抽象为了服务。

另外,设备描述机制不仅仅针对物联网设备,还针对目前广泛存在的以网关形式连接受限设备并代替受限设备发声的物联网网关以及其他各种架构形式搭建的第三方物联网系统,只要具备暴露接口,均能在设备描述文件中进行配置且它们最终除提供的服务不同外不存在其他区别。

采用设备描述机制理论上可以完全屏蔽设备的异构性、提高与第三方物联网系统的互操作性,但配置设备描述文件仅仅是设备服务化抽象的开始,具体的实现还需要依赖于良好的设备模板设计。

4.1.2 设备模板

设备模板由在微服务内部使用的基于消息的通信模块、设备通信模块、报文解析模块、设备映像模块动态组合而成。其中设备通信模块将根据描述文件中设备通信协议自动选择,报文解析模块在填充描述文件中用户配置的返回触发条件、返回处理函数等后生效。

目前 SupconIoTP 系统已集成包括 TCP,HTTP,CoAP,MQTT 协议在内的多种常见通信协议的通信模板。对于智能程度较高且具备网络通信能力的物联网设备、第三方物联网系统、物联网网关来说,上述几种通信协议模板足以满足其连接要求。

对于不具备通信能力的设备,如使用 OPC,Modbus 等工业总线协议的设备,也提供了相应的解决方案,即在现场建立通信服务器或者给设备添加额外的网络通信模块以实现设备

连接。由于报文解析模块是根据用户配置信息动态建立,其创建难度和修改代价都远小于建立现场通信服务器或开发物联网网关,因此,在开发部署的初期,追求快速应用能力时,可以选择为物联网设备添加通信模块以透传方式传递数据,在设备描述文件中设计完备的消息解析和处理方式以从透传数据中提取信息。

4.1.3　设备映像

设备服务化后还存在一系列问题。首先,由于网络波动以及设备本身的原因,事实上是无法保证设备时时在线的,每次对设备发送的请求的可靠性是无法预测和保证的,但服务化后,每次服务调用无论设备状态如何均应当具备明确的返回,必须保证服务时时可用。其次,设备原本只接受来自内部网络小部分用户的请求,但服务化后,面向的用户数量剧增,需要考虑高并发情况下的服务响应能力。最后,在多用户场景下的高并发服务调用也会带来服务冲突问题。而设备映像模块正是被设计来解决这些问题的。

设备映像的本质是对设备采集数据、并记录设备的预期状态。在设备描述文件里,将设备属性分为了可读、可写两类,设备映像中对两者采取了不同的处理方式。对于设备的可读属性,设备映像会保存其返回消息的最新处理结果,在给定的单次访问周期内,所有请求都将直接返回保存结果。对可写属性又将其分为两类,部分对操作顺序有依赖关系的,将以队列形式按调用顺序保存每条控制请求;其他对前后操作无依赖的,则只保存最新的控制请求。在满足属性访问权限限制的基础上,来自高权限用户的服务调用将覆盖低级用户的服务调用。最后在单次访问周期结束后,或设备重新上线时,将主动发送缓存的控制请求。图 3 为设备映像对服务调用的处理流程图。

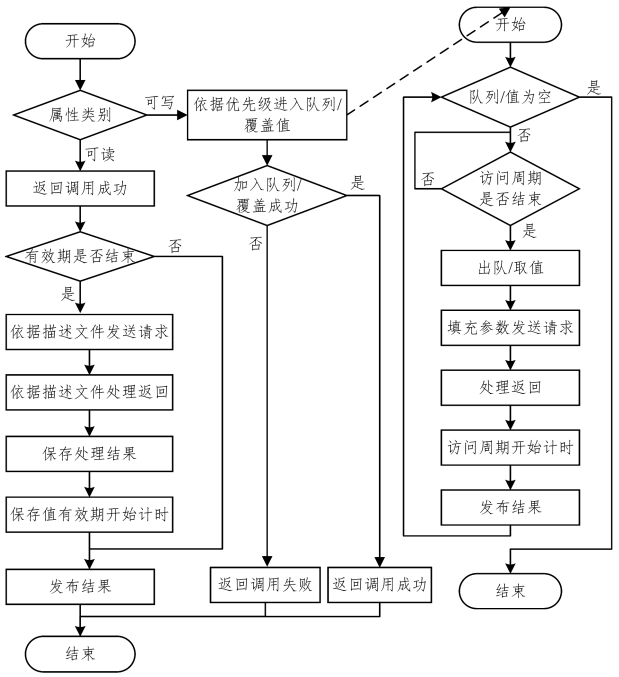


图 3　设备服务调用流程图

面对服务调用,设计了两段式的返回,每次同步返回调用受理情况,具体执行结果则以异步形式发布。对于可写属性衍生的服务的调用,在冲突发生时,将依据权限优先级、调用顺序执行覆盖或入队,被抛弃的服务调用将以执行失败事件形式发布。

设备映像保证了在单个访问周期内,仅会执行一次访问请求。在服务冲突发生时,只对最高优先级的服务调用负责。设备映像组件能有效提高服务并发处理能力,减轻网络通信压力。即使是能力受限的设备亦无须考虑服务化后带来的访问压力,这对低功耗设备、内存受限设备、部署在特殊传输网络的设备来说十分友好。

4.2　规则微服务

规则微服务内部集成了基于订阅/发布形式的通信模型,可与多个微服务交互,支持在运行时制定规则,自由实现服务的组合和数据信息的进一步处理。

规则代表了服务的一次组合,对应实际生产过程中的一个自动化控制流程。完整的规则由数据来源、数据处理、数据流向 3 部分组成。数据的来源和最终流向都是中间件内部的微服务(不仅限于设备微服务,可以是其他规则微服务、存储微服务或其他类型的云服务)发布、订阅的主题。数据的处理可以使用中间件提供的基础处理函数实现包括逻辑判断、数据聚合、数据筛选等诸多基本数据处理功能。规则以简单的类 SQL 中 select 的语句组织,使用者以填充参数的方式制定规则。图 4 为规则的代码模板。

```
SELECT {parameter} AS {return} FROM {sub-topic}
WHERE {trigger} TO {pub-topic}
```

图 4　规则代码模板

图 4 中各个参数的含义如表 2 所列。

表 2　规则参数表

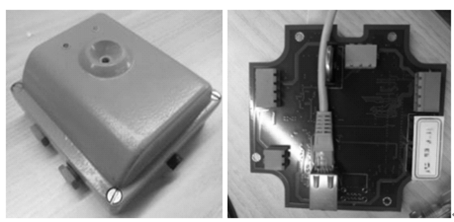
参数	含义
parameter	数据来源 JSON 中的字段名
return	数据流向 JSON 中的新字段名
sub-topic	订阅的某服务的主题
trigger	判断条件
pub-topic	本服务发布的主题

在规则制定时,可以使用由中间件提供的基础处理函数,通过处理函数的组合实现复杂的数据处理功能。规则的制定自由度高,规则微服务与其他微服务搭配使用可以实现许多实际生产中的重要功能,如与用户界面微服务搭配使用可以实现监控数据的实时报警,与存储微服务搭配使用可以自动去冗余并记录监测值等。

5　实例验证

本次验证环境基于 Docker 容器[15]和 Kubernetes 管理工具开发的 supconIoTP 中间件,使用自主研发的某红外物联网设备验证异构设备的接入、多用户的服务冲突解决。

该红外设备使用红外线监测并记录向内向外的行人数,采用 HTTP 协议通信,以 GET 方式提供人流量统计清零接口和实时数据查询接口。图 5 为本次使用的红外设备原型。



(a) 外观图　　(b)内部摄图

图 5　红外设备原型图

红外设备描述文件的部分关键信息如表 3 所列。

表 3 红外设备描述配置表

配置项	配置值
DeviceProtocol	http
DeviceURL	{address}
AttributeName1	realtime_data
AttributeType	readable
AccessPermission	group-only
OperationURL	http://{address}:{port}/readltime_data
OperationTrigger	true
OperationReturnType	application/json
OperationReturn	result
OperationInterval	5 * 1000
OperationPolling	1 * 60 * 1000
AttributeName2	count
AccessPermission	group-only
OperationURL	http://{address}:{port}/count
OperationReturnType	application/json
OperationTrigger1	result['result']==“OK”
OperationReturn1	“清零成功”
OperationTrigger2	result['result']!=“OK”
OperationReturn2	“清零失败”
OperationInterval	24 * 60 * 60 * 1000
OperationPolling	24 * 60 * 60 * 1000
OperationSave	False
OperationParmEnable	False

默认可读属性 realtime_data 衍生服务的成功调用结果将直接发布到 realtime_data/data 主题下。所有服务的失败调用将发布到该设备的 err 主题下。同时,在描述文件中也设定了 realtime_data 属性衍生服务将以 1 次/分钟的频率自动轮询。

对可写属性 count 即清零接口的返回消息设置了两个判断触发条件,将根据返回消息中 result 字段的值发布清零成功或清零失败的信息到 count/data 主题下。又因为统计清零服务不依赖于服务调用顺序,因此选择以值覆盖的形式保存最新的服务调用,并设置该服务单次执行间隔和轮询间隔均为 1 天。

设备微服务创建成功后,在规则微服务中制定了相应的规则来获取实时统计数据、清零接品调用信息、失败信息。设置的规则如图 6 所示。

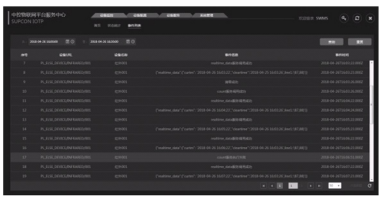
```
#Rule1:
select * as 'realtime_data',GETDATE() as 'optime' from {DeviceID}/get/realtime_data/data to {用户界面微服务}/{DeviceID}/get/realtime_data/success
#Rule2:
select * as 'success',GETDATE() as 'optime' from {DeviceID}/update/count/data to {用户界面微服务}/{ DeviceID }/get/count/success
#Rule3:
select * as 'err',GETDATE() as 'optime' from {DeviceID}/err to {用户界面微服务}/{ DeviceID }/get/err
```

图 6 红外设备规则代码

在设置规则后,可在平台提供的 web 界面上查看该红外设备的实时情况,如图 7(a)所示。其中标记处是该红外设备的实时设备映像,可以看到其中保存有 count,realtime_data 属性,根据描述文件,对 realtime_data 属性的查询服务调用结果将保存 5s,count 属性则保存了来自级别为 3 的设备拥有者的轮询服务调用。

(a)服务冲突发生前

(b)服务冲突发生后



(c)事件消息截图

图 7 红外设备运行截图

接着将使用优先级别更高的管理账号对 count 属性的衍生服务进行调用,如此便会与设备拥有者服务轮询之间的服务控制冲突。此时优先级别更高的服务调用将覆盖原设备映像中的保存值,如图 7(b)所示。

另外,在设备拥有者的事件列表里(见图 7(c)),也可以看到对 count 属性的服务调用在被覆盖后自动返回的服务执行失败事件。

结束语 本文设计并实现了一种基于微服务架构的通用型物联网中间件,该中间件能有效提高物联网的扩展能力。使用设备描述文件可以对不同物端进行描述,基于设备描述可以生成具有针对性的独特设备微服务,能有效屏蔽物端在通信协议、传输网络、数据处理方面的异构特点,在与第三方物联网的水平互联和构建物联网生态系统方面提供了一种解决方案。此外,设备映像的设计能减少并解决多用户场景下的服务冲突,提高服务的可靠性,减少设备功耗和网络压力,为受限设备提供友好的交互方式。

在实践过程中,中间件快速部署、开发可扩展、有效屏蔽异构性的特点十分突出。但当出现网络波动时,无法保证服务的实时性要求;在存在大量设备直接接入的情况下,中间件端面临巨大的网络压力考验。因此,未来的工作方向将集中在与边缘计算的协同发展上。

参 考 文 献

[1] 陈海明,石海龙,李勐,等. 物联网服务中间件:挑战与研究进展[J]. 计算机学报,2017,40(8):1725-1749.

[2] 陈海明,崔莉. 面向服务的物联网软件体系结构设计与模型检测[J]. 计算机学报,2016,39(5):853-871.

[3] VICAIRE P A,Z X,HOQUE E,et al. Physicalnet:A Middleware for Programming Concurrent,across Administrative Domain Sensor and Actuator Networks[C]// Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems. New York:ACM,2009:317-318.

(下转第 604 页)