计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
50(6): 1198-1209, 2013

一种基于智能物体的物联网系统及应用开发方法

孔俊俊　郭　耀　陈向群　邵维忠
(北京大学信息科学技术学院软件研究所　北京　100871)
(高可信软件技术教育部重点实验室(北京大学)　北京　100871)
(kongjj07@sei.pku.edu.cn)

An Approach to Building Systems and Applications of Internet of Things with Smart Things

Kong Junjun, Guo Yao, Chen Xiangqun, and Shao Weizhong
(*Institute of Software, School of Electronics Engineering and Computer Science, Peking University, Beijing* 100871)
(*Key laboratory of High Confidence Software Technologies (Peking University), Ministry of Education, Beijing* 100871)

Abstract With the rapid development and application of pervasive computing and Internet of Things, more and more artificial things are augmented by seamlessly embedding new capabilities such as sensing, actuating, communication and calculation. Compared with their traditional counterparts, the augmented artifacts can perform complex tasks more intelligently, automatically, and even collaboratively. These augmented artifacts are referred to as smart things (or smart objects), which are becoming basic building blocks of Internet of Things and driving the emergence of novel pervasive applications. However, programming smart things faces challenges because they are dynamic and inherently heterogeneous in terms of their built-in functionalities, computational capabilities, network interfaces, etc. This paper proposes an interaction mechanism and programming abstraction to support smart things-based application development. A middleware system is implemented to realize the interaction mechanism and programming abstraction while providing runtime support for heterogeneous smart things. With the help of the proposed programming abstraction and interfaces, application developers can easily program smart household appliances such as smart TV, smart air conditioner, smart projector, smart light, and so on. Case studies and experimental results show that the proposed smart things based approach can be utilized to develop systems and applications of the Internet of Things flexibly and effectively.

Key words smart things; the Internet of Things (IoT); programming abstraction; heterogeneity; middleware

摘　要　随着普适计算和物联网技术的发展和应用,人们不断地把传感、效应、通信和计算能力嵌入到现实世界的人工物体中,使其更加智能化、自动化甚至协同地完成复杂的任务,这类能力得以增强的新型物体被称为智能物体或者智能对象.智能物体将成为物联网的基本构造单元,促进新型物联网应用的涌现.但是,基于智能物体开发物联网系统和应用面临着智能物体的动态性以及它们在固有功能、计算能力、网络接口等方面的异构性带来的挑战.为此,提出了一种智能物体交互机制和统一的编程抽象来支

收稿日期:2012-12-31;修回日期:2013-04-09
基金项目:国家"九七三"重点基础研究发展计划基金项目(2009CB320703);国家自然科学基金项目(61103026);国家"八六三"高技术研究发展计划基金项目(2011AA01A202);国家自然科学基金创新群体基金项目(61121063)

持基于智能物体的应用系统开发,实现了一个中间件系统来完成该交互机制和编程抽象,同时提供运行时支持.基于提出和实现的智能物体编程抽象,应用开发者可以容易地开发基于智能电视、智能空调、智能投影仪、智能电灯等设备的应用程序.应用举例和实验结果表明,利用基于智能物体的开发方法能够灵活有效地开发物联网应用.

关键词 智能物体;物联网;编程抽象;异构性;中间件

中图法分类号 TP311

自从 Weiser 在 20 世纪 90 年代提出普适计算[1-2]的概念以来,出现了很多具有计算能力的新型物理设备.这类新型设备包括大到智能电视、智能空调等智能电器,小到智能手机、智能相机、智能灯泡(例如,LIFX 灯泡[3]和 Lumen 灯泡[4])、智能手表(如 Leikr 手表[5])、智能鞋子等.它们被称为智能对象[6-7](smart objects)或者智能物体[8-9](smart things).

智能物体一般是指无缝地嵌入了计算、通信、传感和效应能力的人工物体(artificial things).相对于传统人工物体,智能物体具有可编程扩展和交互协作的特性.它们为物联网技术的发展带来了新的机遇和应用场景,同时也面临新的挑战.一方面,随着物联网[10]技术和信息-物理融合系统[11]的发展,出现了大量基于智能物体的新型应用,例如:家电控制[3-4]、增强零售体验[12]、物体搜索[13]、智能交通和物流[7]等.但是另一方面,智能物体之间的异构性对构建物联网系统和应用带来巨大的挑战.

利用智能物体开发复杂的应用程序需要掌握大量的专业知识,包括特定物理设备的硬件接口、设备支持的网络接口和协议以及操作系统和编程语言等.这给应用程序的开发者带来了巨大困难,降低了开发效率,而且由于需要考虑大量的底层细节而容易产生错误.鉴于此,已经有很多研究提出用中间件来缓解这些问题[14-16],近期的研究工作有 Costa 等人的 TeenyLime[17];Kawsar 等人实现的 FedNet[18]以及微软研究院的 HomeOS[19]等.这些系统在一定程度上屏蔽了底层平台的复杂性,但是,它们通常仅支持特定领域的应用(例如无线传感器网络应用)或者不能有效地支持动态变化的设备集合.其中绝大多数中间件系统并不是针对智能物体而设计的,没有考虑到智能物体显著的动态性和异构性等特点.FedNet 支持基于智能物体的编程开发,但是它依赖于一个集中的协调控制节点,并不能充分支持智能物体之间直接的局部化交互.

利用智能物体编程需要考虑智能物体本身的特性:动态性和异构性.这些特性都为智能物体编程带来困难.1)分布在局部环境(例如:家居环境、办公室、实验室等)中的智能物体集合是动态变化的.一方面环境中的智能物体可以按需增加或移除;另一方面,很多智能物体本身就是移动的,诸如:智能手机、智能手表、智能鞋子等.此外,考虑到智能物体未来可能会密集分布在生活工作环境当中,空间上临近的智能物体有更多的机会发生交互,共享各自的功能和信息.这种局部化的直接交互是有意义的,而且未来将会普遍存在于智能物体之间.例如:智能手机可以直接与周围环境中的智能家居设备交互,辅助用户完成一些控制协调任务.智能设备的这种动态特性要求运行在智能设备上的应用程序必须基于一个动态变化的设备集合构建业务逻辑.2)智能物体在固有功能、计算能力、网络接口等方面的异构性使得基于智能物体编程更加困难.不同的智能物体可能具有完全不同的固有功能,例如:智能灯泡的基本固有功能是照明,而智能空调的基本固有功能是调剂空气温度.不同的智能物体也可能拥有不同的计算能力、网络接口和操作系统等,例如:前面提到的 LIFX 灯泡支持 Wi-Fi 网络而 Lumen 灯泡支持 Bluetooth 网络.

基于上述分析,本文提出了一种智能物体交互机制和统一的编程抽象(programming abstraction)来支持基于智能物体的物联网系统和应用开发,同时开发了一个中间件系统来实现该交互机制和编程抽象.本文的主要贡献如下:1)提出了一种智能物体局部化交互机制.该机制使得智能物体之间能够动态相互发现和局部化直接交互.2)提出了适用于智能物体的编程抽象.该编程抽象为应用程序开发者屏蔽了智能物体的异构性,使得开发者可以不必掌握智能物体底层实现的细节而较容易地开发基于智能物体的应用程序.3)实现了一个中间件系统为智能物体提供运行时支持.该中间件系统运行在每个智能物体上,实现了前述的智能物体交互机制和编程

抽象.智能物体应用程序就部署在该中间件层之上.

1 智能物体

1.1 智能物体概念

计算技术的发展趋势是把传感、效应、通信和计算能力无缝地嵌入到现实世界中的物理设备,使其更加智能化和自动化,甚至能够协同地完成复杂的任务.这类能力得以增强的物理实体(augmented physical entities)被称为智能对象或者智能物体.由于二者的含义基本一致,因此一般可以将它们互换使用,本文中采用智能物体(smart things)这个概念.目前,智能物体的实际例子有智能灯泡(如LIFX和Lumen)、智能手机(如Android手机、iPhone)等.表1列出了几种典型的智能物体及其被增强的能力.

Table 1 Examples of Smart Things

表1 智能物体举例

Smart Things	Augmented Capabilities
Smart Phone	A plenty of sensors embedded (e.g., Android phone supports more than 10 kinds of sensors.), with a relatively powerful processor (some high-end devices already have quad-core CPU of more than 1 GHz.) and multiple kinds of network connections (such as Wi-Fi, Bluetooth, 3G, NFC, etc.)
Smart TV	A plenty of sensors embedded (e.g., camera, acoustic sensor, light sensor, and so on), as well as powerful computing capability (powerful CPU of 1 GHz or higher, 512 MB RAM memory or bigger) and diverse interaction abilities (LAN, Wi-Fi, USB, Bluetooth, etc.)
Smart Camera	A plenty of sensors added (including GPS sensor, accelerator, etc.), as well as network communication capability (Wi-Fi, 3G, and Bluetooth) and computing power (general-purpose CPU and mobile operating system added)
Smart Watch	Sensing capability (such as pedometer), networking capability (Bluetooth and/or 3G supported), and computing capability (CPU, memory and operating system embedded)
Smart Meter	Sensing capability (automatically monitoring energy consumption), actuating capability (remotely controllable), and communication capability (dual-way network connection between power meter and control center)
Smart Shoes	New capabilities are added: Computing, sensing (such as GPS sensor, pressure sensor, motion sensors, etc.), actuating (e.g., internal shape adjustable via software program), and networking (M2M, Wi-Fi, etc.)

智能物体是对加入了数字化和信息化能力的人工物体的抽象,与之类似的概念有"数字化物体"[20](digital artefact)、Smart u-Things[21]、智能对象等.最近,Kortuem等人对智能对象的定义是"被加入了传感、计算和网络能力的自治物理/数字化对象",并且提出用智能对象来构建物联网系统[6].

1.2 日常物体到智能物体的转化方法

通过无缝地嵌入计算、通信、传感和效应等硬件模块可以把日常物体(daily things)转变为智能物体(smart things).随着物联网及其相关技术的不断发展和应用,人类生活和生产当中日常物体的功能会逐步得以扩展和增强,从而具备可编程扩展和交互协作的特性.例如,Gellersen等人提出的Smart-Its[22]就是一种集成了处理器、传感器、效应器和无线通信模块的小型集成设备,可以用来把日常物体转化为智能物体.从组成结构上来看,Smart-Its与现在的无线传感器网络节点(即mote)很类似,但是前者作为物体的一个附加模块来使用.

本文采用一个集成的可附加单元把日常物体转化为智能物体,称之为智能物体可嵌入单元(smart thing embeddable unit),如图1所示①.它主要由3部分组成:1)计算硬件(hardware),包括处理器、内存等计算能力模块、传感器、效应器和网络模块;2)适配器(adapter);3)操作系统(OS),包括内核、运行库和常用服务等.其中计算硬件和操作系统都可以根据具体需求进行不同的配置.例如,可以根据需要选择传感器节点、单片机等作为计算硬件平台,而操作系统则可以根据智能物体的功能需要和硬件平台的限制来选择.适配器对于智能物体可嵌入单元至关重要,主要用来屏蔽日常物体的固有功能模块和接口的异构性.适配器可以把不同的设备特定交互接口转换为相对统一的接口展现给智能物体可嵌入单元.

① 本文给出的这种方法适用于把已有的日常物体转换为智能物体的情形.如果是设备制造商设计制造新型的智能物体,则可以采用软硬件协同设计的方法,而不一定采用这种"适配器"方案.

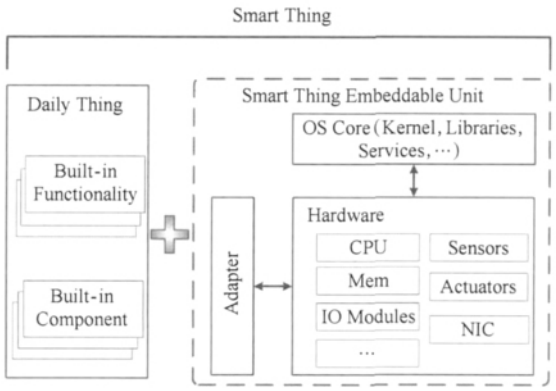


Fig. 1 Smart thing embeddable unit.

图 1 智能物体可嵌入单元

2 智能物体交互机制和通信模型

空间上临近的智能物体可以进行局部化直接交互(direct localized interaction),而不必经过特定节点的协调和控制.智能物体的功能一般是专有的特定功能,不容易被其他智能物体替代.分布在局部环境中的智能物体之间的交互有着明显意义,即共享各自的特定功能,组合出整体功能.智能物体可以动态地发现周围环境中的其他智能物体及其提供的功能(一般用服务的形式封装并且发布出来).在发现了临近的智能物体及其功能之后,智能物体可以通过服务调用的方式使用临近物体提供的功能.例如,一个智能手机可以动态地发现办公室内的智能电灯和智能空调等,并且可以与之交互来控制室内环境(如调节亮度和温度等).

为了使得智能物体具备动态发现和交互的能力,所有的智能物体都应该遵循一定的交互模式.但是,考虑到智能物体的异构性,即便是所有的智能物体都遵循相同的交互模式,它们之间也可能因为底层网络接口的不一致而无法相互发现和交互.这种情况是普遍存在的,例如,前面提到的 LIFX 智能灯泡内嵌了 Wi-Fi 网络接口而 Lumen 灯泡内嵌了 Bluetooth 网络接口,二者无法直接发现和交互.为了解决智能物体底层网络接口的异构性对动态发现和交互带来的问题,本文提出了一种智能物体通信网关来中介异构智能物体间的交互.

智能物体交互通过消息(message)交换来实现.下面给出的 STCP 通信模型为应用程序开发者提供一个方便易用的消息传输机制,屏蔽网络异构性,使得应用程序可以通过同一个抽象接口与不同类型的智能物体交换消息.

2.1 交互模式

智能物体能够互相发现和交互的基本前提是它们都遵循 4 阶段交互模式(interaction pattern):广告、发现、通信及协作.

1) 广告(advertising).智能物体启动后,会通过可用的网络通道多播或者广播其标识信息(identity),例如:UUID、智能物体类型、其他辅助描述信息等.可用的网络通道包括无线局域网(WLAN)、蓝牙(Bluetooth)、ZigBee 等.

2) 发现(discovery).智能物体通过收听周围物体的广播消息来发现临近的智能物体.与已有的 UPnP,INS 等服务发现协议[23]不同,智能物体发现支持在多个异构网络上发现邻居节点而不依赖中央协调节点.每个智能物体都拥有这样的分布式物体发现能力.

3) 通信(communication).智能物体的交互对象总是动态发现的智能物体,这就保证了每个智能物体总是能够与当前可用的设备交互,而不必依赖于静态配置.这种设计可以很好地应对局部环境中设备集合的动态性.智能物体通信面临的问题是,不同智能物体支持不同的网络接口和协议.为此,本文提出了智能物体通信网关和 STCP 模型来支持任何两个异构智能物体之间的消息交互.

4) 协作(cooperation).智能物体在消息交换的基础上完成互操作.智能物体把自己的特定功能封装为服务,并且发布出来让别的物体使用.其中涉及的主要过程有服务发现和服务调用,它们都是通过智能物体消息交换来实现的.

2.2 智能物体通信网关

智能物体通信网关是一个多网络接口智能物体,专门用于中介内置的不同网络接口的异构智能物体之间的发现和交互.它通过提供两个特殊的服务来实现网关的功能:发现代理和消息代理.网关拥有最多的底层网络接口,因此能够发现局部环境中所有的智能物体.其他物体只需要查询该网关就可以发现原本不能发现的物体.同时,网关也提供消息转发代理,使得不同网络接口的智能物体之间可以交换消息,进而可以使用各自的服务.图 2 所示为本文实验中所采用的一个通信网关示意图.比如,Android 智能手机(支持 Wi-Fi 和 Bluetooth)不能直接和传感器节点(支持 ZigBee 通信)发现和交互,但是,在通信网关的帮助下,Android 手机可以和智能传感器交互并且获取传感数据.

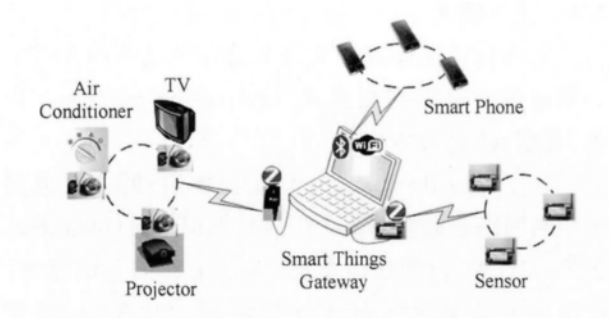


Fig. 2 Smart things communication gateway.

图 2 智能物体通信网关

2.3 STCP 通信模型

STCP 是指智能物体通信管道(smart things communication pipe),用来在智能物体之间建立类似于 TCP 的数据连接. 智能物体的异构性之一就是网络接口的多样性. 很多医疗设备、传感设备、电子消费产品并不支持 IP 网络,而一般仅支持无网络基础设施的 Ad Hoc 网络,典型的例子有蓝牙和 ZigBee 等。不同的智能物体可能内置了不同的通信模块,这使得应用开发者需要同时掌握多种网络协议和接口的使用方法才能完成必要的交互. STCP 专门为网络异构的智能物体而设计,简化了异构智能物体之间的消息传输(message transportation). 利用 STCP,智能物体应用程序可以和任意网络的智能物体交互而不必考虑底层网络的细节和差异性.

STCP 通过包装和适配现有的网络协议实现对智能物体底层网络异构性的屏蔽,如图 3 所示. 图中虚线下方的 W,B 和 Z 分别代表 Wi-Fi、Bluetooth(蓝牙)和 ZigBee. STCP 通信管道为上层提供了数据流(STCP stream),而数据流是通过收发 STCP 消息实现的. STCP 消息可以通过任意可用的底层网络和相关协议传输到通信对方. STCP 依赖于现有的网络和通信协议:基于 IP 网络的 TCP 协议、蓝牙 RFCOMM 协议和 ZigBee 协议等. RFCOMM 协议和 ZigBee 协议支持无基础设施辅助的点对点通信. 因此,STCP 可以基于 IP 网络和 Ad Hoc 网络灵活地连接智能物体.

STCP 利用错误检测和修复机制应对无线网络连接的不可靠性. 由于智能物体的移动性,网络拓扑容易发生变化,底层网络可能会随时中断. STCP 管理模块支持出错重传机制. 当前使用的底层网络断开时,管理模块会立即尝试其他的网络接口重传失败的 STCP 消息. 这样,STCP 利用底层多接口(图 3 中所示的 WBZ)来提高数据交换的可靠性.

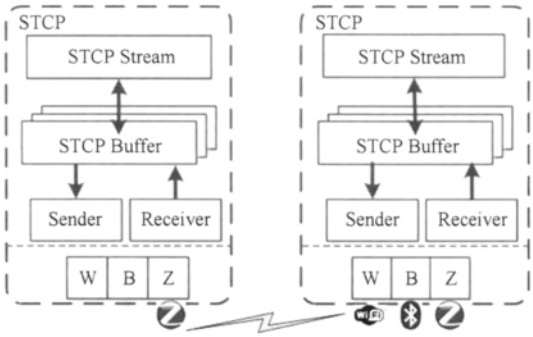


Fig. 3 Smart things communication pipe.

图 3 智能物体通信管道

STCP 通信管道带来的优点有:1)屏蔽底层网络接口的异构性,简化智能物体之间的消息交换;2)通过错误检测和修复机制提高智能物体消息交互的鲁棒性;3)适配性,即在运行过程中动态地匹配最合适的网络接口和协议.

3 基于智能物体的编程抽象

基于上述的智能物体交互机制和 STCP 通信通道,异构的智能物体之间可以相互发现并且传输消息,从而解决智能物体底层网络的异构性带来的问题. 但是,基于智能物体开发物联网应用系统仍然面临着智能物体在固有功能、编程语言等方面的差异性带来的困难. 智能物体编程抽象专门解决后面的问题.

智能物体编程抽象包括 3 部分内容:1)智能物体;2)智能物体服务;3)智能物体统一编程接口. 前面介绍过 1),下面重点介绍 2)和 2).

3.1 智能物体服务

通过智能物体服务可以把智能物体固有的特定功能封装为服务(service). 例如,一个智能灯泡有这些固有功能:1)调节光照颜色;2)调节光照亮度. 通过把固有功能封装为智能物体服务,该智能灯泡对外提供两项服务:改变颜色(changeColor)和改变亮度(changeBrightness). 其他智能物体可以通过访问这两项服务来使用它提供的特定功能. 尽管不同的智能灯泡可能有不同的内部接口和实现,但是它们都可以提供相同的服务(因为它们拥有相同的内在功能). 从而使得基于智能物体服务的应用程序具有更好的适用性. 智能物体服务为访问异构物体的特定功能提供了一种途径.

3.2 智能物体统一编程接口

应用程序可以利用一个统一的编程接口来对智

能物体编程.该编程接口定义了对智能物体进行操作的方法,并且把处理智能物体异构性的大量任务交给中间件来完成.

表2给出了智能物体统一编程接口,其中的方法可以分为3组:

Table 2 Programming Interface of Smart Things

表2 智能物体编程接口

Method Name	Description
QueryThings()	Query all active smart things in the vicinity of this smart thing.
ThingAttr()	Retrieve all attributes of the smart thing specified by a smart thing ID.
ThingSvc()	Retrieve all services of the smart thing specified by a smart thing ID.
ThingCall()	Invoke a smart things service by specifying smart thing's ID and service ID, along with proper operation arguments.
RegThingSvc()	Register a local smart things service to the underlying infrastructure (i.e., service manager in the ThingWare middleware system) to make it available to surrounding smart things.
UnregThingSvc()	Unregister a specified local smart things service from the ThingWare system.

1)发现临近的智能物体及其属性(QueryThing和ThingAttr);2)使用智能物体服务(ThingSvc和ThingCall);3)本地注册和注销服务(RegThingSvc和UnregThingSvc).QueryThings查询该智能物体发现的其他智能物体;ThingAttr获取指定智能物体的属性.ThingSvc查询指定智能物体提供的服务;ThingCall在指定的智能物体上调用指定的服务.RegThingSvc向当前的智能物体中间件注册一个智能物体服务,使得其他智能物体可以查询且使用该服务.UnregThingSvc则从所在智能物体中间件服务管理器注销一个智能物体服务.服务注销以后其他智能物体不能发现和调用它.

4 系统设计和实现

本文采用一个中间件系统ThingWare来实现前面描述的智能物体动态交互机制、STCP通信模型和智能物体编程抽象.ThingWare运行在每个智能物体上,为基于智能物体及其编程接口开发的应用程序提供运行时支持.下面分别讨论ThingWare系统设计和ThingWare在多个平台上的实现.

4.1 ThingWare系统设计

ThingWare中间件系统的主要目标有:

1)提供对应用友好的编程接口来访问智能物体服务;

2)管理发现的临近智能物体及其智能物体服务等;

3)处理智能物体交互,隐藏智能物体在网络接口和协议方面的异构性.

图4所示为ThingWare中间件系统结构.每个智能物体可以分为5层结构,从下往上依次为:物理实体(包括了固有功能和模块)、计算硬件、操作系统、ThingWare中间件、应用程序.

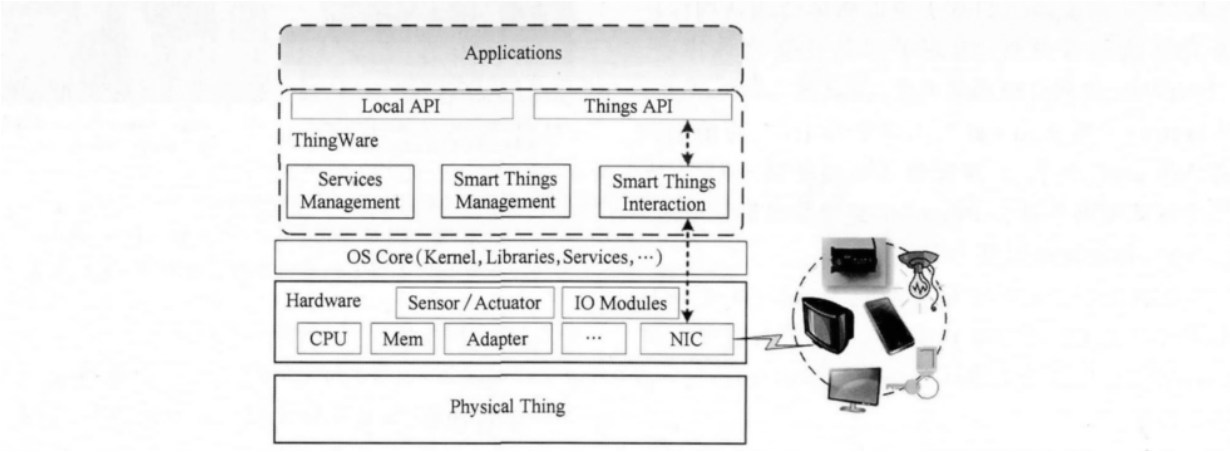


Fig. 4 ThingWare middleware structure.

图4 ThingWare中间件结构

ThingWare中间件有4个关键模块(module)来实现其设计目标,分别是:1)智能物体统一编程接口(Things API);2)智能物体交互(smart things interaction);3)服务管理(servicesmanagement)和

4)智能物体管理(smart things management).智能物体编程接口实现智能物体编程抽象,提供智能物体统一编程接口所定义的访问方法.智能物体交互模块实现 STCP 通信模型以及智能物体的动态发现.动态发现的智能物体由智能物体管理模块维护和管理,它记录每个可发现智能物体的属性、标识信息、网络接口等.服务管理模块管理智能物体服务,同时维护发现的智能物体提供的服务以方便应用程序查询.

智能物体统一编程接口的实现依赖于上述 2)~4)三个模块.智能物体管理模块能够支持 QueryThings 和 ThingAttr 编程接口.该模块动态地维护已经发现的临近智能物体及其相关属性(如名称、智能对象类别等).当 QueryThings 被应用程序调用时,Things API 模块从模块 4)获得当前发现的智能物体列表.类似地,服务管理模块为智能物体编程接口的服务相关方法(即 ThingSvc, ThingCall, RegThingSvc 和 UnregThingSvc)提供支持.ThingSvc 和 ThingCall 的实现需要依赖服务管理模块和智能物体交互,其更多细节将在 4.2.2 节介绍.RegThingSvc 和 UnregThingSvc 分别在本地的服务管理模块中注册和注销一个本地智能物体提供的服务.当调用 RegThingSvc 时,服务管理模块在本地服务列表中查找是否已经注册该服务,若未找到则将新服务添加到本地服务列表中.UnregThingSvc 使得模块执行相反的操作.

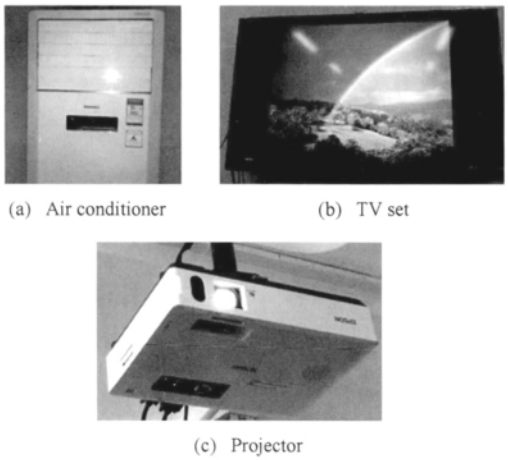
4.2 系统实现

本文实现了一个 ThingWare 中间件系统原型.同时,为了验证提出的基于智能物体的物联网应用开发方法的有效性,实现了多种智能物体原型.ThingWare 中间件原型实现在 TinyOS 2.1, SunSpot, Windows 7 和 Android 2.3.3 平台上,支持的编程语言为 nesC 和 Java.智能物体原型是基于 IRIS 无线传感器网络节点、SunSpot 传感器节点等实现的.

4.2.1 智能物体原型

实现的智能物体原型有智能传感器(基于 IRIS 节点和 TinyOS)、智能电视(smart TV)、智能空调(smart AC)、智能投影仪(smart projector)等,如图 5 所示.

图 6(a)所示为智能电视、智能空调和智能投影仪使用的智能物体可嵌入单元原型.该原型由两部分组成:SunSpot 通信节点和红外适配器模块(IR adapter).SunSpot 节点作为计算和通信模块存在,运行 ThingWare 中间件原型.红外适配器模块(图中圆形部件)起到桥接 SunSpot 通信节点和其他红外控制设备的作用,使得该可嵌入单元能够正确地封装与之连接的设备的固有功能.红外适配器模块能够学习一定的红外控制信号序列,并且把该信号序列和一个指定的操作码(operation code)关联.由于一个红外信号序列对应红外控制设备① 的一个特定功能,因此,一个操作码和一个特定功能对应.这样,经过红外适配器模块的中介,SunSpot 节点可以通过操作码来控制红外控制设备.

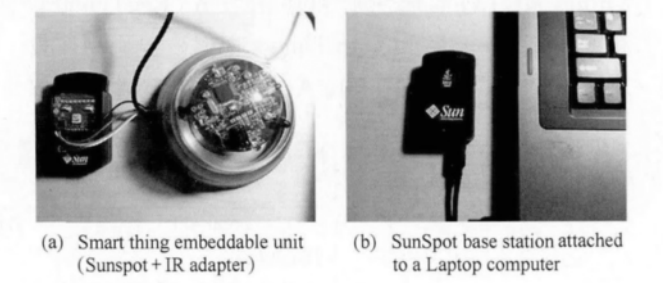


(a) Air conditioner (b) TV set

(c) Projector

Fig. 5 Prototypes of smart things.

图 5 智能物体原型



(a) Smart thing embeddable unit (Sunspot + IR adapter) (b) SunSpot base station attached to a Laptop computer

Fig. 6 Prototype implementation of smart thing embeddable unit and the SunSpot base station.

图 6 智能物体可嵌入单元原型实现和 SunSpot 基站

4.2.2 ThingWare 中间件系统实现

ThingWare 中间件主要实现智能物体交互机制和编程抽象,为智能物体提供运行时支持.如图 7 所示,ThingWare 主要提供三大管理功能:智能物体管理、STCP 通信管道管理和智能物体服务管理.其中左起第 1 个圆角方框中的 Discovery 和 Smart

① 例如:现有大部分家电设备都是红外控制设备.

Things Pool 对应后面两个方框中的 Smart Things Mgmt.(智能物体管理),Data Transfer,STCP Buffer 和 STCP Stream 一起对应后面的 STCP Mgmt.(通信通道管理).

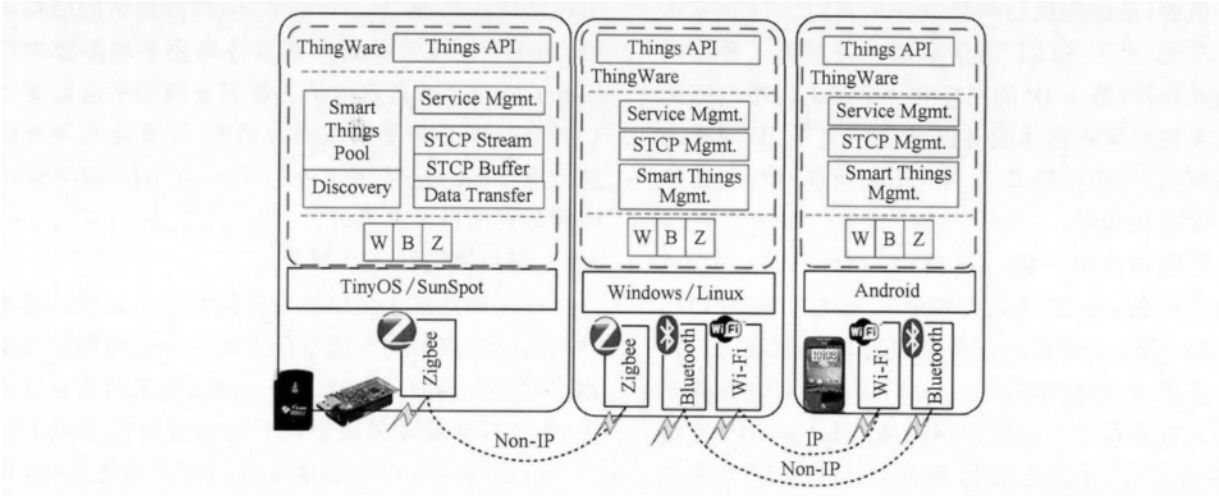


Fig. 7 Implementation of the ThingWare middleware.

图 7 ThingWare 的实现

1) 智能物体交互机制的实现. 运行在不同操作系统平台的 ThingWare 都实现了基本的智能物体动态发现协议. 该协议要求每个智能物体对于其他智能物体的发现请求给予正确的应答. ThingWare 交互模块把该智能物体的标识信息(包括 UUID、类型、其他辅助描述信息等)用 JSON 数据交换格式编排并且广播或者应答. 标识信息交换是通过任何可用的网络实现的,目前支持的网络有 Bluetooth,Wi-Fi 和 ZigBee 三种. 基于 TinyOS 的智能物体(如智能传感器)使用活动消息(Active Message)广播标识信息. 基于 SunSpot 的智能物体(如智能电视、空调等)使用 RadioGram 数据包交换标识信息. 运行在 Android 和 Windows 上的 ThingWare 则基于 UDP 广播和蓝牙 SDP 协议发现临近的智能物体.

STCP 通信管道的实现依赖于现有的网络接口和通信协议,诸如基于 Wi-Fi(WLAN)网络的 TCP 协议、蓝牙 RFCOMM 传输协议(SPP 协议)、基于 ZigBee 协议的 Active Message 和 RadioGram 等. 建立 STCP 通信管道的过程:首先探测本地支持的网络,然后与对方支持的网络匹配,最后从匹配的网络中选择最佳者建立连接. 例如,一个智能手机和支持蓝牙和 Wi-Fi 的笔记本建立 STCP 管道时,ThingWare 中的 STCP 管理模块(即 STCP 管理器)探测到有蓝牙和 Wi-Fi 两种网络接口可以用,为了提高通信性能而会选择 Wi-Fi 网络建立底层链路传输 STCP 消息.

2) 智能物体编程抽象的实现. 智能物体编程抽象包括 3 要素:智能物体、智能物体服务和编程接口. 编程接口定义了一组操纵智能物体和服务的方法.

智能物体服务的实现包括两部分,即服务发现和服务调用. ThingWare 能够在智能物体发现的基础上进一步发现临近智能物体上的服务. 服务调用是对服务的一次访问,通过 STCP 通信管道来传递参数和返回执行结果. 服务调用的参数和执行结果都按照 JSON 数据格式编排,用智能物体消息封装后交换. 与传统的 Web Service 相比,ThingWare 支持的智能物体服务更加轻量,并且专门为资源受限的设备进行优化. 本文实现的智能物体服务既支持基于 IP 网络的服务调用,又支持基于 Ad Hoc 网络的服务发现和调用,因此更能适应智能物体异构和动态的特性.

智能物体服务发现和调用是基于智能物体发现和 STCP 通信管道实现的. 智能物体服务发现并不依赖于专门的服务发现代理服务器,而是由每个智能物体内嵌实现. 在智能物体发现机制的基础上实现服务发现. ThingWare 的服务管理模块为临近智能物体提供服务查询(QueryServices)和服务查找(SearchService)两项基本服务. 当 ThingWare 主动查询一个智能物体 S_{thing} 提供的服务时,它首先通过建立的 STCP 通信管道调用 S_{thing} 的 QueryServices 服务,然后将返回的结果(S_{thing} 服务列表)存储到本地,以便 Things API 的 ThingSvc 接口查询. 服务调用的过程与服务发现的过程是类似的. 首先客户端

ThingWare 通过 STCP 通道将编排好的参数包发送到服务端,然后服务端 ThingWare 回调已经注册服务函数,最终把执行结果编排后通过 STCP 发送回客户端. 由于 STCP 通信通道已经屏蔽了底层网络的异构性(基于 IP 的网络和 Ad Hoc 网络),因此本文实现的智能物体服务既能支持基于 IP 网络的服务调用,也能支持基于 Ad Hoc 网络的智能物体服务发现和调用.

智能物体统一编程接口(Things API)是通过 Java 库(library)的形式实现的①,称为 Things API 库. ThingWare 的核心管理和服务模块单独运行在一个进程中,编程接口(Things API)被通过库代码的形式包含在每个应用程序内部. Things API 库提供前面定义的智能物体编程接口,内部实现与 ThingWare 进行通信,通过 ThingWare 的服务实现 Things API 的功能.

表 3 所示为实现的智能家电设备原型所支持的智能物体服务,从左到右依次为智能电视(TV)、智能空调(AC)、智能投影仪(projector)提供的服务. 除了共有的"打开关闭"服务外,其他服务都是因物而异的. 比如:智能电视提供调整音量和频道的服务;而智能空调则提供调整温度和风速等服务. 这些服务都是对各自内置固有功能的封装.

Table 3 Smart Things Services

表 3 智能物体服务

Smart TV	Smart AC	Smart Projector
Turn ON/OFF	Turn ON/OFF	Turn ON/OFF
Mute	Sleep Mode	Search Signal
Increase Volume	Increase Temperature	Show Menu
Decrease Volume	Decrease Temperature	Zoom In
Previous Channel	Change Wind Speed	Zoom Out
Next Channel		

5 实验评估

基于实现的智能物体原型和 ThingWare 中间件系统原型,本文开发了一些基于智能物体的应用实例,并且通过一系列实验评估了 ThingWare 系统的性能. 从基于智能物体编程接口(Things API)的应用实例代码可以看出,本文提出的编程抽象能够极大地简化对智能物体的编程,只需要用大约 10 行的 Java 代码就可以实现动态地发现并且打开智能电视的应用功能. 从 ThingWare 性能评估的结果来看,对于智能传感器的一次服务调用平均需要大约 30 ms,而对于智能家电的一次服务调用平均需要约 1 s(1 000 ms). 智能物体固有功能、计算能力和通信能力等方面的差异性导致了 Things API 服务调用时间延迟的巨大差异.

5.1 应用举例和 API 展示

智能物体之上可以部署应用程序来完成特定的功能. 比如,部署在智能空调上的一个应用程序来根据房间内人员的存在情况和室内温度来调节自身行为(例如打开或者关闭空调). 运行在智能手机上的一个应用程序可以动态地发现临近的智能物体,并且生成用户界面允许用户控制这些发现的智能物体(如智能家电).

1) 实验室智能空调自动开关应用. 该应用程序运行在智能物体可嵌入单元中. 与之交互的有两类智能物体,分别是智能传感器和智能 RFID 读卡器. 通过智能传感器提供的温度传感服务就可以获取室内温度,通过智能 RFID 读卡器的 RFID 标签查询服务可以获得室内人员存在情况(假定每个实验室工作人员都佩戴一个主动式 RFID 标签). 当室内有人并且温度高于 25℃时,该应用主动打开空调;而当室内无人时该应用主动关闭空调以节约电能.

基于本文提出并实现的智能物体编程接口,上述应用逻辑的实现会非常简洁. 首先应用开发者不必了解智能传感器和智能 RFID 读卡器的实现细节,也不必关心底层网络通信的细节,而只需要关注高层应用逻辑即可. 其次,该应用所依赖的两种智能物体是动态发现和绑定的,这为实验室设备升级和替换带来灵活性.

2) 智能手机智能物体控制台应用(console application). 智能手机已经是一种非常流行而且功能强大的智能物体,可以运行很多应用程序来辅助用户完成工作和生活中的大量任务. 基于 ThingWare 中间件和 Things API 库,开发者可以轻松地开发各类应用程序来发现和控制临近空间内的各种异构智能物体. 具体例子包括智能家电控制、智能物体搜索、智能购物等.

图 8 所示为 Android 智能手机应用使用智能物

① 由于目前无法在 TinyOS 上动态部署应用程序,因此本文工作并未在 TinyOS 上提供独立的 API 库,而是直接基于 ThingWare 源码开发 nesC 应用.

体编程接口(Things API)打开智能电视的代码片段.可见,利用该编程抽象可以简洁地完成异构智能物体交互的功能.从图中代码可知,大约需要10行(54行到67行,除去注释行)的Java代码就可以让一个Android应用发现并打开一个智能电视.

```
54    List<Thing> thingsList = null;
55    ThingsManager thingsManager = ThingsManager.getDefaultManager();
56    //get a list of surrounding things
57    thingsList = thingsManager.queryThings();
58    //filter out a smart thing of interest
59    int tv = ThingType.HOUSEWARE_TV; //note: this value is shared globally
60    Thing tvThing = findThing(thingsList, tv);
61    if (tvThing == null) { return;  }
62    //get services provided by a smart thing
63    String thingID = tvThing.getUuid();
64    List<ThingService> servicesList = thingsManager.thingSvc(thingID);
65    if (!findService(servicesList, "turnOnOff")) { return; }
66    //perform a service call on that thing
67    ThingServiceCall serviceCall = new ThingServiceCall("turnOnOff", null);
68    ThingServiceResult result = null;
69    try {
70       result = thingsManager.thingCall(thingID, serviceCall);
71    } catch (ThingServiceException e) {
72       e.printStackTrace();
73    }
74    if (result == null) {System.out.println("Service invocation on a TV failed."); }
75    else {
76       String retTpye = result.getType();
77       String retValue = result.getValue();
```

Fig. 8 Code snippet of controlling a smart TV with Things API.

图8 使用智能物体编程接口控制智能电视的代码片段

5.2 ThingWare系统性能

本文旨在智能物体编程抽象为基于异构物体编程带来的高效率和灵活性,而ThingWare系统实现了该编程抽象,且为智能物体的动态发现、服务发现和调用提供运行时支持.从应用程序开发角度来看,基于Things API开发的智能物体应用主要依赖于智能物体发现和服务调用.因此重点从两个方面来评估Thingware系统性能:1)智能物体发现的开销,包括发现过程的时延和发现率;2)智能物体服务调用带来的开销,用服务调用的平均时延来度量.

5.2.1 实验环境配置

在评估ThingWare原型系统性能的过程中,主要使用了IRIS传感器节点(IRIS-XM2110 mote)、SunSpot节点和笔记本计算机.笔记本计算机配置:Core i5 CPU;4 GB内存;运行Windows 7系统.

为了评估IRIS智能传感器物体发现和服务调用的时间开销,在实验室环境(约6 m×9 m)依次部署1~9个传感器,并且用笔记本计算机收集性能数据.为了评估基于SunSpot改造的智能家电设备的服务调用开销,在会议室(约5 m×6 m)中部署了智能电视、智能空调和智能投影仪来收集性能数据.

5.2.2 实验结果

1) 智能物体发现的开销.图9所示为智能物体发现的性能,其中的智能物体都是基于IRIS节点的智能传感器(IRIS smart sensor).随着环境中IRIS智能传感器数量逐步增加,智能物体发现所消耗的总时间线性地增加.平均而言,发现一个IRIS智能传感器的时延约为10 ms.可以发现,当临近的智能物体数量大于5时,每次主动查询获得的邻居节点数量小于真实部署的节点个数.经过多次实验发现,这是由于ZigBee协议丢包导致的.尽管平均情况如此,但经过多次查询之后所有部署的IRIS智能传感器都能被找到.值得说明的是:这些数据适用于由运行在IRIS智能传感器上的ThingWare主动发起智能物体发现的情形.ThingWare通过广播发起查询请求时,临近的IRIS传感器同时受到请求并且立即应答,从而导致信道冲突而容易丢包.

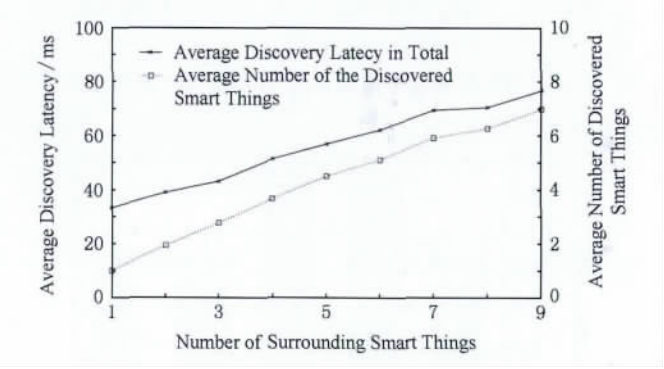


Fig. 9 Performance of smart things discovery.

图9 智能物体发现的性能

2) 智能物体编程接口(Things API)的开销.图10展示了IRIS智能传感器服务调用的时间开销.从图10可知,调用一次传感器服务的延迟大约为30 ms.那么可以在1 s内对一个智能传感器进行30多次查询.这样的采样频率可以满足一般应用的需求,因此智能物体编程接口引入的时间开销是可以接受的.

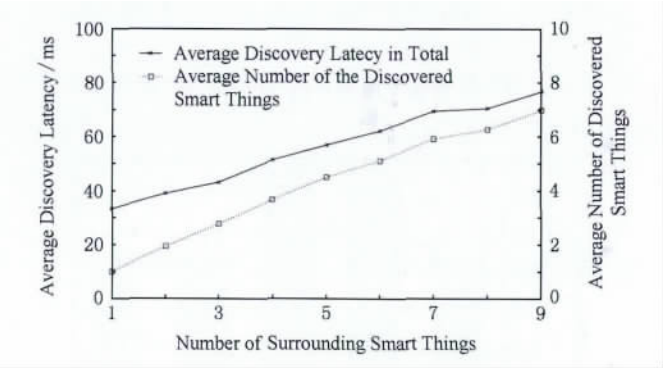


Fig. 10 Latency of Things API service invocation on IRIS smart sensor.

图10 IRIS智能传感器服务调用的时间开销

图 11～13 分别描述了对智能电视、智能空调和智能投影仪的不同服务进行调用的时间开销. 可以发现,智能物体不同服务调用的延迟有着较为明显的区别. 例如,通过 Things API 调用智能电视的"Turn ON/OFF"服务的平均延迟为 1 027.8 ms,而调用调节音量服务(即"Increase Vol"和"Decrease Vol")的开销只有 820～840 ms. 服务调用延迟的差异性反映了这些服务所对应的功能本身的差异性.

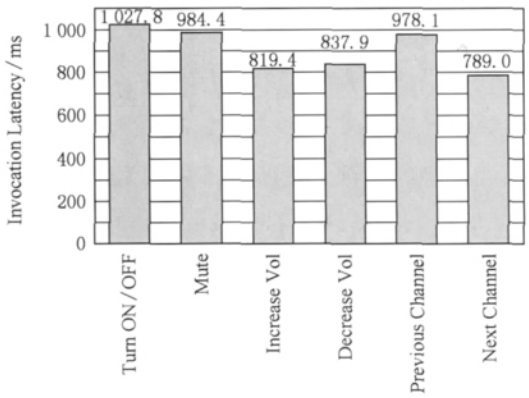


Fig. 11 Things API invocation latency of different services on smart TV.

图 11 对智能电视不同服务的 Things API 调用延迟

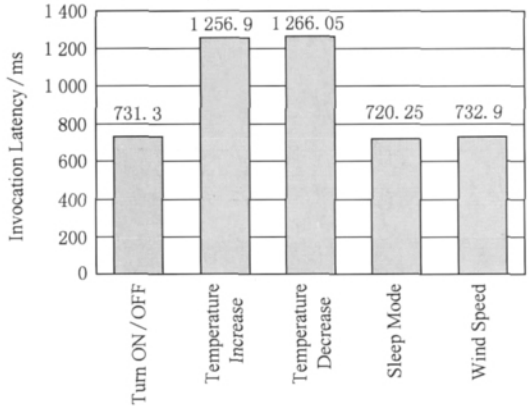


Fig. 12 Things API invocation latency of different services on smart AC.

图 12 对智能空调不同服务的 Things API 调用延迟

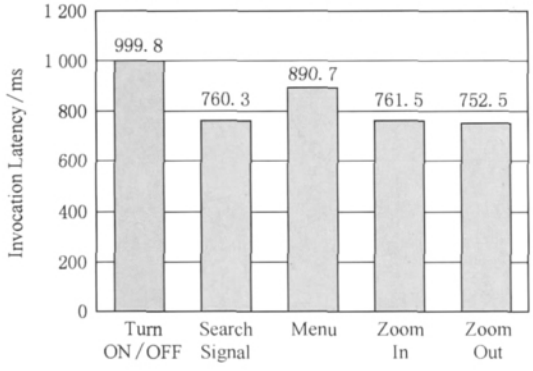


Fig. 13 Things API invocation latency of different services on smart projector.

图 13 对智能投影仪不同服务的 Things API 调用延迟

3) 不同类型智能物体性能的比较. 表 4 比较了在不同类型智能物体上进行物体发现和服务调用的时间开销. IRIS 智能传感器(IRIS smart sensor)和智能家电(SunSpot-powered smart appliance)的平均发现延迟分别是 8 ms 和 38.5 ms. 它们的服务调用平均延迟分别为 29.8 ms 和 893.5 ms. 可见二者差别非常大. 这种巨大的差异本质上是由异构智能物体固有功能的差异所导致的. 经过测试发现,用来收集性能数据的笔记本计算机和智能家电可嵌入单元中的 SunSpot 节点之间的通信开销大约为20 ms. 那么,服务调用延迟的绝大部分是由红外适配器模块导致的(消耗了超过 97%的时间). 由于不同的智能物体具有不同的固有功能,而智能物体服务是对固有功能的封装,因此,固有功能的差异性导致智能物体服务执行时间的差异非常明显.

Table 4 Comparison of Average Discovery and Service Invocation Latency on Different Types of Smart Things

表 4 不同类型智能物体中物体发现和服务调用平均延迟的比较 ms

Smart Thing	Discovery	Invocation
IRIS Smart Sensor	8	29.8
SunSpot-Powered Smart Thing	38.5	893.5

6 总 结

智能物体是嵌入了传感、效应、通信和计算能力的人工物体,是计算技术发展的一个新的趋势. 相对于传统人工物体,智能物体具有可编程扩展和交互协作的特性,它为物联网技术的发展带来了新的机遇和应用场景,同时也面临着动态性和异构性所带来的挑战.

本文提出了一种基于智能物体开发物联网系统和应用的方法. 该方法把智能物体作为物联网系统的基本构造单元,其核心是智能物体编程抽象. 智能物体编程抽象对应用程序屏蔽了底层智能物体的异构性和实现的复杂性,从而有助于开发基于智能物体的应用程序. ThingWare 中间件系统实现了智能物体编程抽象并且对智能物体提供运行时支持. 应用举例和实验结果证实了该方法的灵活性和有效性.

参 考 文 献

[1] Xu Guangyou, Shi Yuanchun, Xie Weikai. Pervasive/ubiquitous computing [J]. Chinese Journal of Computers, 2003, 26(9): 1042-1050 (in Chinese)
(徐光祐, 史元春, 谢伟凯. 普适计算[J]. 计算机学报, 2003, 26(9): 1042-1050)

[2] Weiser M. The computer for the 21st century [J]. Scientific American, 1991, 265(3): 94-104

[3] Bosua P. LIFX: The light bulb reinvented [EB/OL]. [2010-10-01]. http://www.kickstarter.com/projects/limemouse/lifx-the-light-bulb-reinvented

[4] Indiegogo. Lumen Bluetooth LED lightbulb [EB/OL]. [2012-12-01]. http://www.indiegogo.com/lumen

[5] Leikr. Leikr Linux watch [EB/OL]. [2012-10-06]. http://leikr.com/devices/

[6] Kortuem G, Kawsar F, Fitton D, et al. Smart objects as building blocks for the Internet of things [J]. IEEE Internet Computing, 2010, 14(1): 44-51

[7] López T S, Ranasinghe D C, Patkai B, et al. Taxonomy, technology and applications of smart objects [J]. Information Systems Frontiers, 2009, 13(2): 281-300

[8] Langheinrich M, Mattern F, Römer K, et al. First steps towards an event-based infrastructure for smart things [C] // Proc of Ubiquitous Computing Workshop (PACT 2000). Piscataway, NJ: IEEE, 2000: 1-13

[9] Kuniavsky M. Smart Things: Ubiquitous Computing User Experience Design [M]. San Francisco: Morgan Kaufmann, 2010

[10] Atzori L, Lera A, Morabito G. The internet of things: A survey [J]. Computer Networks, 2010, 54(15): 2787-2805

[11] Li Renfa, Xie Yong, Li Rui, et al. Survey of cyber-physical systems [J]. Journal of Computer Research and Development, 2012, 48(6): 1149-1161 (in Chinese)
(李仁发, 谢勇, 李蕊, 等. 信息-物理融合系统若干关键问题综述[J]. 计算机研究与发展, 2012, 48(6): 1149-1161)

[12] Jode M D, Barthel R, Rogers J, et al. Enhancing the 'second-hand' retail experience with digital object memories [C] //Proc of the 2012 ACM Conf on Ubiquitous Computing (UbiComp'12). New York: ACM, 2012: 451-460

[13] Mayer S, Guinard D, Trifa V. Searching in a Web-based infrastructure for smart things [C] //Proc of the 3rd Int Conf on the Internet of Things (IoT 2012). Piscataway, NJ: IEEE, 2012: 119-126

[14] Mohamed N, Al-Jaroodi J. A survey on service-oriented middleware for wireless sensor networks [J]. Service Oriented Computing and Applications, 2011, 5(2): 71-85

[15] Raychoudhury V, Cao J, Kumar M, et al. Middleware for pervasive computing: A survey [J]. Pervasive and Mobile Computing, 2013, 2(9): 177-200

[16] Mottola L, Picco G P. Programming wireless sensor networks: Fundamental concepts and state of the art [J]. ACM Computing Surveys (CSUR), 2011, 43(3): 19: 1-19: 51

[17] Costa P, Mottola L, Murphy A L, et al. Programming wireless sensor networks with the TeenyLime middleware [C] //Proc of Middleware 2007. Berlin: Springer, 2007: 429-449

[18] Kawsar F, Nakajima T, Park J H, et al. Design and implementation of a framework for building distributed smart object systems [J]. The Journal of Supercomputing, 2010, 54(1): 4-28

[19] Dixon C, Mahajan R, Agarwal S, et al. An operating system for the home [C] //Proc of the 9th USENIX Conf on Networked Systems Design and Implementation (NSDI'12). Berkeley: USENIX Association, 2012: 25

[20] Beigl M, Gellersen H, Schmidt A. Mediacups: Experience with design and use of computer-augmented everyday artefacts [J]. Computer Networks, 2001, 35(4): 401-409

[21] Ma J. Smart u-Things and ubiquitous intelligence [C] //Proc of the 2nd Int Conf on Embedded Software and Systems (ICESS'05). Berlin: Springer, 2005: 776

[22] Gellersen H. Smart-Its: Computers for artifacts in the physical world [J]. Communications of the ACM, 2005, 48(3): 66

[23] Zhu F, Mutka M W, Ni L M. Service discovery in pervasive computing environments [J]. IEEE Pervasive Computing, 2005, 4(4): 81-90

Kong Junjun, born in 1984. PhD candidate in the Institute of Software at Peking University. Student member of China Computer Federation. His main research interests include ubiquitous and pervasive computing, system software and software engineering.

Guo Yao, born in 1976. Associate Professor in the Institute of Software at Peking University. Senior member of China Computer Federation. His main research interests include system software, low power system design, compiler technology and software engineering, etc(yaoguo@sei.pku.edu.cn).

Chen Xiangqun, born in 1961. Professor in the Institute of Software at Peking University. Senior member of China Computer Federation. Her main research interests include operating systems, embedded systems, software engineering, and so on(cherry@sei.pku.edu.cn).

Shao Weizhong, born in 1946. Professor in the Institute of Software at Peking University. Senior member of China Computer Federation. His main research interests include software reuse, component technology, object-oriented methods, software engineering, etc(wzshao@pku.edu.cn).