基于WSAN的物联网软件分布式知识框架及其实现方法

刘正浩[1] 张广泉[1,2]

(苏州大学计算机科学与技术学院 江苏 苏州 215006)[1]

(中国科学院软件研究所计算机科学国家重点实验室 北京 100190)[2]

摘 要 随着物联网的深入发展和智能传感器的出现,传统软件逐渐暴露出自身性能的不足。为满足设备自治的需求和利用边缘网络的计算资源,分析了物联网软件的特征,提出了物联网软件的分布式知识框架及其具体实现方法。通过识别物联网软件的环境逻辑并定义环境逻辑的演化规则,使环境逻辑可以嵌入底层无线节点。依靠智能传感器监控环境变化、触发软件逻辑,实现软件的正确执行。最后依据现实场景的运行情况及相关指标的分析,来说明所提方法的适用性。

关键词 物联网软件,环境逻辑,演化规则,智能传感器

中图法分类号 TP311.52 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.01.023

Distributed Knowledge Framework of IOT Software Based on WSAN and Its Implementation Strategy

LIU Zheng-hao[1] ZHANG Guang-quan[1,2]

(School of Computer Science and Technology, Soochow University, Suzhou, Jiangsu 215006, China)[1]

(State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Science, Beijing 100190, China)[2]

Abstract With the development of the IOT (Internet of things) and the emergence of intelligent sensors, the traditional software has gradually exposed the shortcomings of its own performance. In order to meet the needs of equipment autonomy and utilize the computing resources of the edge network, the characteristics of the IOT software were analyzed and a distributed knowledge framework with its implementation method was put forward. By identifying the environmental logic and defining its evolutionary rules, the logic was embedded in the underlying wireless nodes. Then smart sensors monitor environmental changes and trigger related software logic for the correct software operation. Finally, the operation performance of the method in the environment and the analysis of the relevant indicators explain the applicability of proposed method.

Keywords Internet of things software, Environmental logic, Evolutionary rule, Intelligent sensor

物联网是由麻省理工学院提出的,被认为是继计算机、互联网之后,全球信息产业的第三次经济浪潮[1-2]。目前,物联网还未出现一个被广泛接受的定义[3],其本质是实现物理世界、信息世界和人类社会三元空间的深度融合。物联网是一种新的应用模式,在未来智能通信领域起着中流砥柱的作用[4]。它是在互联网的基础上进行物理实体的扩展,使得互联网获得感知、控制、执行等能力的新型网络结构。相似的概念还有信息物理融合系统CPS[5-6]、社会物联网SIOT[7-8]等。

随着技术的发展,物理世界出现了一种功能强大的感知设备。这种设备拥有更强的数据处理能力和无线通信能力,通常被称为智能传感器mote。除了强大的功能,mote还提供了编程接口,使得开发面向物联网节点的应用成为可能[9]。它们通过自组网和互联网建立通信,组成动态松耦合的分布式智能系统,即物联网。新型设备的出现给物联网带来了新的潜力和发展方向——物联网边缘计算时代(云端到物端的转变)。这种物联网应用系统具有更高的执行效率。但由于节点资源的有限性,传统软件实体并不能直接运行在mote上,因此无法满足物联网的发展需求,需要对物联网中的软件进行深入研究。

本文旨在利用智能传感设备mote和元组空间概念,针对传统软件不能适应物联网发展趋势的问题,提出一种基于WSAN的物联网软件分布式开发方法。为解决软件实体无法在无线节点运行的问题,该方法抽离软件的环境逻辑并参考元组空间设立知识库,然后利用知识库结构化的数据空间存储环境逻辑,最后通过mote强大的处理能力和通信能力实现软件的正确运行,以此有效解决软件的执行限制,适应物联网的发展趋势。

本文第1节介绍物联网软件的相关研究现状和元组空间概念;第2节分析物联网软件的定义;第3节介绍物联网软件的分布式知识框架和具体实现方法;第4节结合农业物联网

到稿日期:2017-12-08 返修日期:2018-04-27 本文受江苏省自然科学基金项目(BK2011281),江苏省普通高校研究生科研创新计划(CXLX13_820),苏州市应用基础研究计划(SYG201241)资助。

刘正浩(1992-),男,硕士生,主要研究方向为物联网,E-mail:20154227020@stu.suda.edu.cn;**张广泉**(1965-),男,博士,教授,主要研究方向为网络软件工程与服务计算、物联网、CPS,E-mail:gqzhang@suda.edu.cn(通信作者)。

场景说明框架的使用，并通过理论分析和实验仿真证明所提方法的优势；最后总结全文。

1　相关工作

物联网的研究主要集中在体系结构设计[10-11]（以云为中心的物联网平台）、物理实体设计[12]（智能体）和设备交互设计[13]（ZigBee，WIFI）等方面。对于运行在物联网环境中的软件，作为实现功能和提供服务的主体并没有得到深入的研究，仍停留在物联网软件的特性、开发困难等方面的理论分析上。

意大利教授 Zambonelli 研究了系统化开发物联网软件的通用规则[14]。围绕物联网系统开发制定了关键概念和抽象：Stakeholders and users，Requirements，Groups and coalitions，Avatars 和 Smart things；并将物联网软件的开发过程分为分析、设计和开发 3 个阶段；最后依据关键抽象包含的实例对象实现对整个软件系统的全局把握。但可惜的是其没有给出软件实现的细化过程。

诺基亚研究员 Antero 和赫尔辛基大学教授 Tommi 分析了物联网软件开发面临的挑战，说明了当今主流开发人员并不熟悉这些挑战。这些挑战包括：多设备编程，系统的被动特性和一直运行特性，异构性和多样性，软件的分布式、高动态和潜在的迁移性，可容错性和可防御性[15]。另外，他们指出了以云为核心的物联网体系的信息存储、信息处理和反馈信息的生成均在云端进行。最重要的是，他们还预测了物联网的发展趋势：now，Edge IoT Era，Universal IoT Era。

中国科学院的陈海明和崔莉对物联网软件的体系结构进行了划分，将之分成了“云端”和“物端”两大种类[10]。“云端”指服务的来源是云中资源提供的云服务；“物端”指服务的来源是智能物体提供的实体服务。同时他们对以“物端”为中心的物联网应用系统进行了进一步的研究，讨论了此类系统的形式化建模方法和模型检测方法。上述这些研究揭示了物联网软件的演化趋势，提供了开发物联网软件的参考信息。

元组空间最早在 Linda[16] 中被提出，其本质是一种数据共享空间，用于存储结构化的数据。它的基本共享单元是元组，例如〈“ID”，11〉。元组空间负责管理和维护元组数据，并对外提供操作接口。在 Linda 中，元组空间有 3 种基本操作：out（插入一个元组）、in（删除一个元组）、read（读取一个元组）。它的最主要特征是实现了空间和时间的解耦。

Benchi 等参考元组空间建立了 JION[17] 架构，实现了分离移动自组网（D-MANETs）的机会通信。通过 JION，基于元组空间的分布式应用可以部署在 D-MANETs 中并正常运行。除此之外，元组空间还可应用在协同通信上。例如，文献[9]利用元组空间建立了智能传感器的协同感知机制，使物联网应用可以更好地适应环境变化。

2　物联网软件

物联网软件不同于传统软件，它与物理环境更加紧密。在物联网软件中，环境是软件组成的一部分。因此物联网软件可以直接实现对环境的调控，以满足三元空间的需求。

从宏观来看，物联网软件是指运行在物联网中的软件实体。因为物联网是在互联网的基础上实现物与物的交互协作（物联网＝互联网＋物物扩展），所以传统软件在宏观视角下可以被称为物联网软件，即分布式 web 软件是物联网软件。因此宏观角度对物联网软件的理解过于简单宽泛，缺乏明确的含义。

从微观上看，物联网软件出现的目的是将物联网软件和传统软件进行区分，因而总结出它们的区别并加以描述，就可以得到物联网软件的明确定义。辨别这两种软件，必须了解它们的载体差异，即互联网与物联网之间的差异。载体差异又体现在载体的组成、结构和关联关系等方面。在分析研究若干文献资料和应用实例后，本文将两者的本质差别概括为两点：1）物体间的自主通信；2）对现实世界施加的直接影响。物联网软件的定义如下。

定义 1（物联网软件）　部署在物联网环境中，拥有感知、采集、存储、计算所处物理空间信息并且可对物理空间施加影响的能力的动态松耦合分布式系统软件。

3　物联网软件的分布式知识框架

3.1　框架概念

由上述定义可知，物联网软件与现实环境有着密不可分的联系，其所有环境操作都可归结为对底层感知执行节点的操控。在物联网中，软件利用传感器采集数据信息，并通过执行器调控物理环境。因此，本文基于无线传感器和执行器网络（Wireless Sensor and Actor Network，WSAN）来研究物联网软件。

在互联网中存在着形态各异的软件，如分布式软件、网构软件等。但从节点层次考虑，这些软件形态都不适用于物联网环境，软件实体的粒度不能匹配资源受限的节点。本文受元组空间和计算迁移的启发，提出了软件的分布式知识框架来适应物联网环境。该方法的分布对象不是软件实体，而是软件的环境逻辑。通过将环境逻辑从软件内部抽离并规则化，获得可以存储在 mote 节点上的关联条件规则（也称为知识），以此实现软件在节点层次的运行。条件规则在传统软件中普遍存在，它与软件代码高度耦合，是软件实体的一部分，并未独立出来。条件规则非常适合资源受限的无线节点，因为其存储空间小、处理简单，最重要的是可以完成与环境的逻辑交互。

图 1 是物联网软件分布式知识框架的结构图，其从实现的角度对框架进行了分层描述。

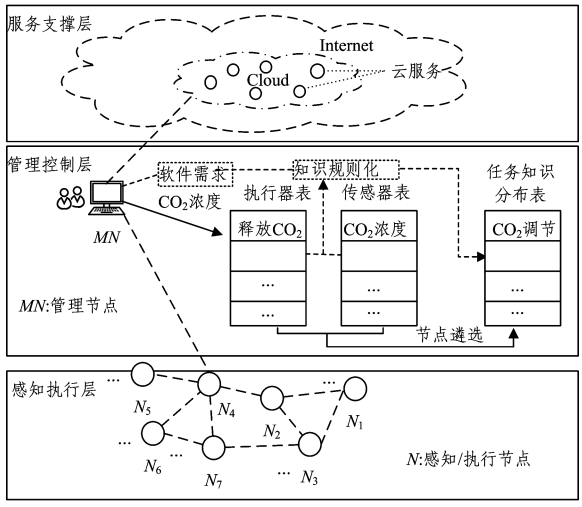

图 1　物联网软件的分布式知识框架图

Fig. 1　Distributed knowledge framework of IOT software

第一层是感知执行层,由传感器和执行器组成;第二层是管理控制层,由计算机组成,是物联网软件主体的运行位置;第三层是服务支撑层,是互联网上的云平台。框架的重点在感知执行层和管理控制层上,将运行在管理控制层的软件逻辑抽取出来生成关联知识,并将其离散分布到感知执行层的节点中,使软件逻辑与底层节点相融合,进而发挥边缘网络的计算能力,促使物体间自主交流,实现软件功能。

各层的主要职责如下:感知执行层利用节点知识实现对物理环境的调控;管理控制层管理底层知识的分布和更新,采用发布订阅机制获取想要显示的环境数据,并且执行其他与环境无关的操作;服务支撑层在管理控制层无法满足软件需求的情况下,提供云端服务,通过服务接口来满足需求。

通过上述分析可知,该方法的优点有:1)节点脱离管理端操控,实现网络的自主运行,保证时间敏感事件的实时性;2)一定程度上实现节点的智能化,发挥物联网的海量计算特性;3)缓解节点的通信能耗,延长 WSAN 节点的生命周期。

3.2 分布式知识框架的设计

为实现物联网软件的分布式知识框架,完成软件逻辑的正确抽取、分布和协作,本文在管理控制层建立了 3 个模块:知识分布模块、感知设备模块和执行设备模块。这 3 个模块对应图 1 的任务知识分布表、传感器表和执行器表。知识分布模块是管理环境逻辑的核心,记录软件逻辑分布的节点位置以及协作顺序,为底层知识的管理提供依据。感知设备模块和执行设备模块记录底层节点的属性特征,为建立和修改知识分布模块提供先验知识。

同时,感知执行层的节点与传统节点不同。传统节点的内部逻辑主体分为 4 个部分,分别是感执器、处理器、通信器以及能量单元。传统节点被管理端控制,由管理端接收感知信息触发事件并启用与事件相关的其他节点。本文提出的方法参考元组空间在传统无线节点中建立了新的逻辑主体——知识库。依靠知识库存储软件的环境逻辑,使程序的执行脱离管理节点,以适应物联网设备的自治需求。该节点的内部结构如图 2 所示。

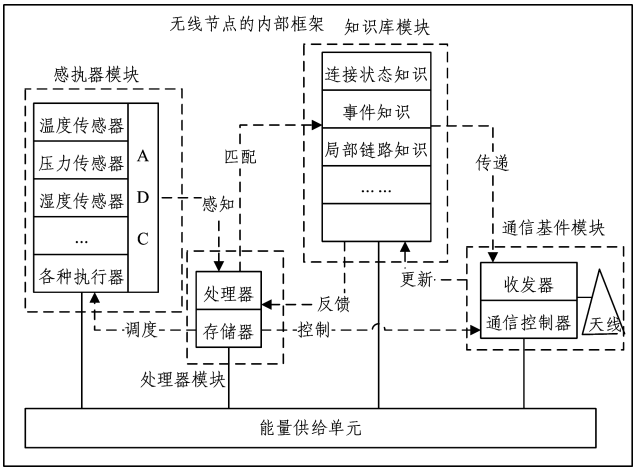

图 2 无线知识节点的内部结构图

Fig. 2 Internal structure diagram of wireless knowledge node

(1)处理器模块:节点的控制核心,管理知识库、处理数据、触发事件。

(2)感执器模块:探测外界环境的接口,是事件发生的源头;也是控制环境的接口,执行调控环境的动作。

(3)通信基件模块:负责无线节点间的相互通信。

(4)能量供给单元:提供其他模块工作所需的能量。

(5)知识库模块:分布式知识框架的核心是虚拟信息空间,用于存储软件的环境逻辑。

图 2 中的处理器通过感执器感知环境信息,循环匹配知识库并得到反馈;调度感执器通过影响环境或控制通信模块实现节点间的交流;知识库通过通信模块实现知识的更新;通信模块借助知识库完成与特定节点的消息传递。

知识库模块是实现节点智能化的原动力。该模块与其他模块不同,非物理实在,是存储知识的虚拟信息结构。在无线节点中,微处理器集成了内存、闪存、数字 I/O。因此知识库的存储可以选择程序闪存,无需改变节点的物理结构,只需分配固定的存储区,不会增加成本负担。另外,通过分析事件的执行流程,本文将知识库的逻辑结构分为 3 部分:连接状态知识、事件知识和局部链路知识。连接状态知识代表 mote 节点间的连接状态,用于探测网络拓扑的改变,修复事件知识;事件知识是软件的环境逻辑知识规则化后生成的离散知识语句,用于软件功能的实现;局部链路知识是事件关联节点间的路由信息,用来完成局部节点间的快速通信响应。这 3 部分是框架稳定运行的必备要素。连接状态知识的信息结构是 N 元祖。例如$\langle NL,N,SI\rangle$,其中 NL 代表本地节点,N 代表邻居节点,SI 代表信号强度。事件知识的信息结构是 IF-THEN 规则或 ECA 规则[18]。局部链路知识的信息结构与连接状态知识相同。除此之外,软件设计者还可以根据领域需求对知识库进行扩展,但是必须考虑节点资源的有限性。

3.3 分布式知识框架的实现

由于软件环境逻辑的执行是通过事件完成的,因此可用事件表示环境逻辑。本文从 3 个方面来考虑方法的实现:事件知识的抽取、事件知识的分布和事件的控制流程。

3.3.1 事件知识的抽取和规则化

软件的环境逻辑通常以事件的形式展现在我们面前。因此,这里将环境逻辑称为事件知识,将抽离过程称为事件知识的规则化,而环境逻辑的识别可以通过用户需求来定义事件规则。

定义 2(事件规则) 某一事件包含的事件要素按照特定规范生成一组相互关联的规则语句。

事件要素指事件源(对象)、事件(触发条件)以及事件处理方式(动作)。特定规范指生成关联规则语句所遵循的预定义方法。最终生成的离散规则语句就是事件规则。举个简单的例子,在智能会议室,当用户使用演示功能时,需要拉上窗帘,同时打开投影降下帷幕。事件知识如下:IF user needs ppt THEN close curtain AND open projector AND down screen。其中包含 4 种对象设备,分别是用户交互设备 I、窗帘开关 C、投影仪 P、幕布开关 S。I 是感知器,C,P,S 都是执行器。在识别出设备对象后,上述事件知识可演化为 4 个部分。

I:IF get info_use_ppt THEN send msg_c_C and send msg_o_P and send msg_d_S

C:IF get msg_c_C THEN functionC(close)

P:IF get msg_o_P THEN functionP(open)

S：IF get msg_d_S THEN functionS(down)

这就是事件知识的抽取和规则化，也称作软件逻辑的知识规则化过程。此外没有使用特定规范。

3.3.2　事件知识的分布

事件知识可以实现物联网软件的环境控制功能，完成物联网的物物交互任务。那么如何选择事件知识的存储节点？针对知识分布，本文提出了以下 4 个问题。

问题 1：当 IoT 中存在相同节点时，如何选择存放事件知识的节点？

问题 2：IoT 中的节点损毁后，如何迁移损毁节点的知识？如何更新相关节点的知识？

问题 3：当 IoT 中的节点知识分配不均时，如何解决知识在某些节点的堆积？

问题 4：如何判断 IoT 节点的区域差异，以及事件知识包含节点的范围？

这 4 个问题的依据分别是物联网节点的冗余性、易损性、资源受限性和区域差异性。冗余性意味着物联网中存在功能相似的节点，这使得节点的选择充满不确定性。易损性使节点设备具有不可靠性，如能源耗尽，因此必须考虑节点内部知识的迁移。节点资源的受限性要求知识分布不能积聚在某些节点中，要尽可能地平均分配。人类社会明确的区域划分导致了节点的区域差异性，知识分布必须满足区域要求才能正确实现物联网软件的功能。这些问题考虑了三元空间的需求，涵盖了知识分布需要解决的所有问题。

图 3 是 WSAN 的局部场景图。图中有温度传感器（SN_2，SN_4）和空调执行器（SN_6）。正常思维下，事件知识应存储在 SN_2 和 SN_6 中。那么如何实现计算机的节点选择？下面定义 5 个启发式规则来辅助计算机选择节点。

$RULE_1$：若与事件关联的一类传感器或执行器在 WSAN 中仅有一个，则事件选择此节点；否则，随机选择。

$RULE_2$：根据 $RULE_1$ 选定的节点，使剩余节点的选择满足与已选节点路由间距最小（奥坎姆剃刀）。

$RULE_3$：满足 $RULE_1$ 和 $RULE_2$ 且备选数目不为 1 时，选择知识量少的节点。

$RULE_4$：当节点损坏时，寻找最近的拥有相同功能的节点替换。

$RULE_5$：事件知识优先分布在同一语义组内，以保证事件的语义完整性。

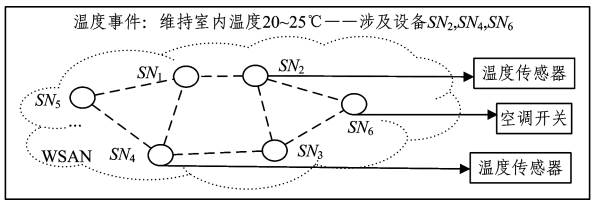

图 3　无线传感与执行器网络的局部节点示意图

Fig. 3　Escene of partial nodes in WSAN

由图 3 可知，事件任务关联的设备有两种，分别为温度传感器和空调执行器。$RULE_1$ 规则下，空调执行器 SN_6 唯一，先选择。$RULE_2$ 下，SN_2 节点和已选节点 SN_6 路由最小为 1，选择 SN_2。此时知识分布在 SN_2 和 SN_6 中。SN_2 损坏后，$RULE_4$ 下，由 SN_4 替代 SN_2 的职责，执行对应的启发式规则。$RULE_4$ 涉及的知识更新由管理控制层根据事件知识分布模块进行处理。上述启发式规则不一定能得到最优的事件分布节点，多数情况下还是次优分布。有些研究者认为不需要求解最优解，通常通过启发式或贪婪算法得到的次优解就足够[19]。

$RULE_1$ 和 $RULE_2$ 解决了问题 1；$RULE_4$ 解决了问题 2；$RULE_3$ 解决了问题 3。那么问题 4 该如何解决？在家庭物联网中，两个相邻房间 A 和 B 都有照明设备，进入 A 时要求打开 A 的照明设备。前 4 个启发式规则不能解决，因为它们功能相同且距离相近，解决方案是 $RULE_5$，赋予节点不同的语义层次。借鉴已建立的领域本体，为系统的运行环境建立语义层级。若在人类社会中某一区域有完整的语义（房间 A），则将其内的物联网设备分为一组。组内的感执设备可以继续分组，以实现区域的局部控制；组与组之间可以组合成更高层次的语义；最高层次的语义为系统自身。

3.3.3　事件的控制机制

在获得环境逻辑的分布节点后，决定节点间的协作通信顺序。合理安排节点的协作跳转可以简化软件逻辑的知识规则化过程。按软件逻辑涉及的节点数目对物联网场景进行分析。定义启发式规则 6 和规则 7。

$RULE_6$：先处理生产信息的节点，后处理消费信息的节点。

$RULE_7$：节点间的通信机制采用时间串行协作机制或空间并行协作机制。

场景 1：单节点

该场景最简单，事件仅含一个节点，无须考虑协作关系，因此触发事件的消息由自身提供并由自身使用。

场景 2：双节点

$RULE_6$ 将节点分为两类：产生数据的生产节点和使用数据的消费节点。这时事件的协作顺序有两种：1）一个节点生产信息供另一个节点消费；2）两节点共同生产信息供其中一个节点消费。

场景 3：多节点

当参与事件的节点数大于 2 时，节点间的协作通信顺序会迅速增加。为简化问题，$RULE_7$ 将事件的控制流程定为并行和串行两种模式。从时间上看，事件运行是一条时间线，是串行运行机制，按时间先后进行节点规则的演绎；从空间上看，事件运行是多点并发的空间线，是并行运行机制，按空间特征进行节点规则的演绎。图 4 展示了同一任务在不同模式下知识规则化的结果。

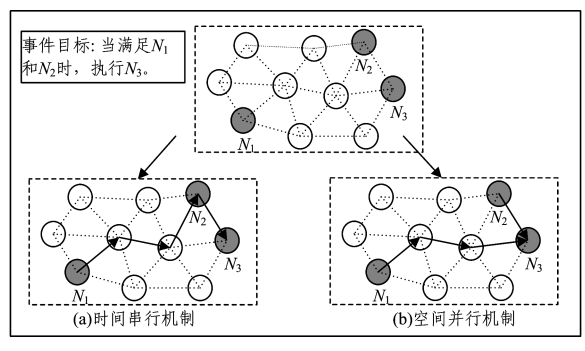

图 4　事件的两种控制机制

Fig. 4　Two management mechanisms of events

根据上述启发式规则，将环境逻辑的整体抽离过程抽象

为算法，具体如算法 1 所示。知识迁移算法如算法 2 所示。

算法 1 软件逻辑知识规则化算法

输入：网络特征信息及环境逻辑信息

输出：ECA 规则

```
Topology wsan_topology; %生成底层对象
devices=wsan_topology.devices; %获取设备信息
nodeset=Ø; %初始化节点集合
planning(Lt)⇒Lt={t1∪t2∪t3∪…∪tn}; %获取逻辑包含的节点类型
FOR EACH type ti in Lt,0<i<n+1 %RULE1选择初始节点
  IF devices(ti) contains only one device
    fnode=devices(ti);
    nodeset=nodeset∪fnode; %添加节点 nodeset
    updateLType (Lt,ti); %删除类型 ti
    break;
  END-IF
  IF i==(n+1) %随机选择,RULE5
    t=random(Lt);
    fnode=random(devices(t));
    nodeset=nodeset∪fnode;
    updateLType(Lt,t);
  END-IF
END-FOR
FOR EACH type ti in L,0<i<n %循环选择剩余节点
  IF ShortPathNode(devices(ti),fnode) has one node
    spnode=ShortPathNode(devices(ti),fnode); %RULE2,RULE5
    nodeset=nodeset∪spnode;
    updateLType(Lt,ti);
  END-IF
  ELSE
    find spnode with least knowledge;%RULE3
    nodeset=nodeset∪spnode;
    updateLType(Lt,ti);
  END-ELSE
END-FOR
nodeset=SortDistributedNode(nodeset); %RULE6重排列
IF Lc=="time_serial" % Lc(时空特征)
  rules=Construct_Rule_by_time(nodeset);%RULE7串行
END-IF
IF Lc=="space_parallel"
  rules=Construct_Rule_by_space(nodeset);%RULE7并行
END-IF
save rules in the distributed knowledge table;
send rules to WSAN.
```

算法 2 知识迁移算法

输入：损坏节点的信息

输出：新的知识分布状态

```
Topology wsan_topology; %生成拓扑对象
WHILE true DO %循环检测节点损坏状况
  node_ruin=read(node_error); %读取损坏信息
  IF node_ruin==NULL %没有损坏节点
    continue;
  END-IF
  ELSE
    updateTopology(wsan_topology,node_ruin);%更新网络
    recognize Logics={l1∪l2∪l3∪…∪ln} related with node_ruin;
    FOR EACH logic li∈Logics %RULE4
      find the nearest node which is able to complete li;%重选节点
      update related knowledge in distributed knowledge table;%更新
      send new knowledge to the related nodes;%分发
    END-FOR
  END-ELSE
END-WHILE
```

4 实验与分析

4.1 案例场景

本文以温室灵芝培育系统为例来展示方法在实际场景中的应用，场景如图 5 所示。设备包括温度传感器 T、酸碱度传感器 P、湿度传感器 H、CO_2 传感器 C、光线传感器 L 以及对应的执行器 AT，AP，AH，AC，AL。同类型的设备用小写字母 a 和 b 区分。

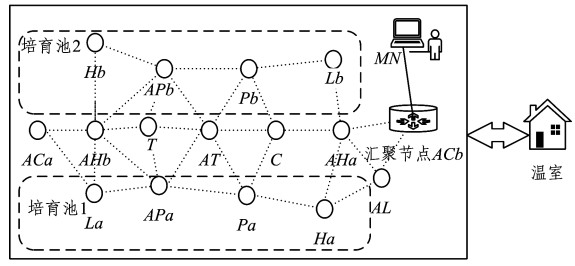

图 5 小型物联网温室智能灵芝培育系统场景图

Fig. 5 Scene of intelligent reishi cultivation system

传感器周期性地读取周围的环境信息。当发现所读取的信息与自身存储的知识相匹配时，根据匹配得到的知识执行相应操作。在该场景中，设备间的通信链路由图 5 的虚线表示。AT 是空调，AP 是酸碱溶液喷头，AH 是雾化水喷头，AC 是风扇。AC 选择风扇的原因是空气中的 CO_2 浓度满足灵芝生长的条件，过高过低都会产生反作用。AL 是灯具，为方便研究，光线只考虑室内灯光。场景的汇聚节点是 ACb，它与管理端直接相连。

4.2 框架应用

以菌丝体阶段为例，分析灵芝培育系统的功能需求，识别系统的环境逻辑。在菌丝体阶段，灵芝正常生长的温度范围为 18～35℃，空气的相对湿度为 70%～80%，不需要光线，最适 PH 值为 5～6，CO_2 浓度低于 0.1%，这实际就是菌丝体阶段的用户需求。

根据所提方法，首先建立管理控制层的感知设备模块和执行设备模块，分别存储传感器的属性特征和执行器的属性特征。根据场景的实际需求，简单划分了设备的语义信息并将其存储在对应表中，具体如表 1 和表 2 所列，这里省略知识分布表。为更加规范地描述事件规则，定义如下描述形式：

E(*event_name*)

ON(*condition*) | [*message*]

DO[*message*{*id*}] | ⟨*action*⟩

该描述形式将事件规则分为 E，ON，DO 3 个字段，分别代表事件、条件和动作。E 字段是事件名称；ON 字段是事件

的触发条件，其中 *condition* 是条件表达式，[*message*]代表接收消息；DO 字段的操作有两种，[*message*{*id*}]指发送消息到标识为 *id* 的节点，〈*action*〉表示执行动作 *action*。首先识别用户需求包含的设备对象，然后根据算法将用户需求知识规则化后分布到相应节点。该场景各节点的知识如下。

温度：

T：E(*temp*) ON(*data*＜18) DO[*t_up_msg*{*AT*}]

E(*temp*) ON(*data*＞35) DO[*t_down_msg*{*AT*}]

E(*temp*) ON(*data*＝26) DO[*t_stop_msg*{*AT*}]

AT：E(*temp*) ON[*t_up_msg*] DO〈*function*(up)〉

E(*temp*) ON[*t_down_msg*] DO〈*function*(down)〉

E(*temp*) ON[*t_stop_msg*] DO〈*function*(stop)〉

湿度(Hb，AHb 略)：

Ha：E(*hum*) ON(*data*＜70) DO[*ha_up_msg*{*AHa*}]

E(*hum*) ON(*data*＝80) DO[*ha_stop_msg*{*AHa*}]

AHa：E(*hum*) ON[*ha_up_msg*]DO〈 *function*(open)〉

E(*hum*) ON[*ha_stop_msg*]DO〈 *function*(stop)〉

PH 值(Pb，APb 略)：

Pa：E(*ph*) ON(*data*＜5) DO[*ph_up_msg*{*APa*}]

E(*ph*) ON(*data*＞6) DO[*ph_down_msg*{*APa*}]

E(*ph*) ON(*data*＝5.5) DO[*ph_stop_msg*{*APa*}]

APa：E(*ph*) ON[*ph_up_msg*]DO〈*function*(up)〉

E(*ph*) ON[*ph_down_msg*]DO〈 *function*(down)〉

E(*ph*) ON[*ph_stop_msg*]DO〈 *function*(stop)〉

CO_2 浓度：

C：E(*co2*) ON(*data*＞0.1) DO[*co2_down_msg*{*ACb*}]

ACb：E(*co2*) ON[*co2_down_msg*] DO〈*function*(down)〉

此时环境逻辑演化为事件规则，委托给 mote 节点管理。当满足触发条件时，事件自动执行。其中，*function* 指执行器的对外接口。需要注意的是，设备间语义需要相配，以保证事件的正确执行。

表 1　传感器表

Table 1　Sensor table

标识	功能	单位	量程	区域	…
T	温度	℃	−40～65	温室	…
Ha	湿度	%	0～100	温室右区	…
Hb	湿度	%	0～100	温室左区	…
La	光照	Lx	0～100000	温室	…
Lb	光照	Lx	0～100000	温室	…
Pa	酸碱度	ph	0～14	培育池 1	…
Pb	酸碱度	ph	0～14	培育池 2	…
C	CO_2	%	0～2	温室	…
…	…	…	…	…	…

表 2　执行器表

Table 2　Actuator table

标识	功能	区域	…
AT	温度	温室	…
AHa	湿度	温室右区	…
AHb	湿度	温室左区	…
AL	光照	温室	…
APa	酸碱度	培育池 1	…
APb	酸碱度	培育池 2	…
ACa	CO_2	温室	…
ACb	CO_2	温室	…
…	…	…	…

4.3　案例分析

系统应用所提方法后，当传感器 *T* 感知室内温度小于 18℃时，匹配知识库成功，触发温度调节事件发送升温消息给执行器 *AT*。*AT* 接收消息后搜索知识库，随后成功匹配到规则，执行事件操作 *function*(up)，打开空调，使室内升温。当空调运行一段时间后，传感器 *T* 感知室内温度为 26℃，再次匹配知识库成功，发送关闭消息给执行器 *AT*。*AT* 接收消息匹配成功，执行事件操作 *function*(stop)。至此，一轮温度调节结束。整个过程 mote 节点自主交流，脱离管理端操控。物联网软件的分布式知识框架满足了物联网边缘时代的设备的自治需求，利用了边缘网络的计算能力。除此之外，该方法还有其他性能优势。下面通过理论分析和实验仿真两种形式来说明该方法的优点。

规定室内环境因素分布均匀且一天内的事件参数如下：温度变化触发的事件次数为 n，湿度、酸碱度、CO_2 变化触发的事件次数分别为 m，k 和 g。总通信跳数是各事件跳数之和。新框架触发一次温度调节产生两次消息传输，一次启动空调，另一次关闭空调，因此温度跳数为 $2n\times(A,AT)=2n$。同理，湿度跳数为 $2m\times(Ha,AHa)+2m\times(Hb,AHb)=4m$。酸碱度跳数为 $4k$。CO_2 跳数有细微区别，因需求差异触发事件只引起一次通信，所以跳数为 $g\times(C,ACb)=2g$。因为场景的汇聚节点 ACb 与服务端通信需要 1 跳，所以传统软件的温度通信跳数为$((T,MN)+(MN,AT))\times2\times n$，即$(5+4)\times2\times n=18n$。同理，湿度跳数为$((Ha,BS)+(BS,AHa))\times2m+((Hb,BS)+(BS,AHb))\times2m=34m$。酸碱度跳数为 $36k$，CO_2 跳数为 $4g$。最终得到式子：

$$\frac{2n+4m+4k+2g}{18n+34m+36k+4g}$$

因此使用所提方法后，软件执行的通信跳数比原先最少可降低 50%，最多可降低 88.89%。本文还使用了 MATLAB 软件对温室灵芝培育系统进行仿真，除应用方法不同外其余参数完全相同。图 6 是培育系统的仿真结果图，图 6(a)为通信跳数对比图，图 6(b)为消息传递时间比对图。在通信跳数对比图和消息传递时间图中，对于环境逻辑嵌入底层节点的软件，其性能明显优于传统软件。因为事件在执行过程中的通信跳数减少，所以事件具有更高的实时性。另外，节点传输消息比处理数据更消耗能量。1 kb 数据包传输 100 m 的能耗与计算能力为 100MIPS/w 的处理器执行 3 百万条指令的能耗相当，因此本文所提方法在物联网环境中还可以缓解物联网节点的能量消耗。因此，其适合物联网环境。

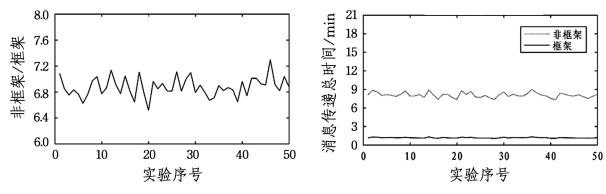

图 6　MATLAB 仿真结果

Fig. 6　Results of MATLAB simulation

结束语　通过上述实验场景的分析可以看到，使用本文

方法开发系统时，每个节点都建立了自己的知识库。节点通过自身知识库中的环境逻辑实现与其他节点的自治通信。知识库的存在使节点的处理能力得到了充分发挥，进而体现出物联网的海量计算特性。除此之外，由于节点数据的本地处理，软件减少了消息的传递，从而保证了软件运行的实时性，延长了底层节点的使用寿命。

本文提出的物联网软件分布式知识框架与传统软件框架的区别主要分为 4 点：1)应用主体不同。物联网软件分布式知识框架针对的是新兴的物联网软件。而物联网软件与传统软件之间存在巨大差异。2)针对场景不同。传统软件框架的应用场景是高端计算设备(电脑、手机等)，而物联网软件分布式知识框架的应用场景是高端计算设备和低端节点设备。3)分布对象不同。传统框架的分布对象是软件组件，而物联网软件分布式知识框架的分布对象则是软件内部逻辑。4)知识表达不同。传统框架中的知识是用来辅助软件正确运行的指导性规则，而物联网软件分布式知识框架中的知识本质上是软件内部的环境逻辑，是由软件逻辑经过特定处理演变而成的。

在本文工作中，规则的协作还仅仅局限在一些比较简单的形式上。但在现实场景中，节点间的协作可能会更加复杂，因此需要研究规则的协作，以优化节点间的通信序列来进一步实现节点的节能。为此，完善规则的协作是一项重要的工作。未来的工作还有：节点知识库的形式化描述；物联网软件分布式知识框架实验平台的搭建；物联网软件分布式知识框架自适应能力的研究等。

参 考 文 献

[1] GERSHENFELD N, COHEN D. Internet 0: Interdevice Internetworking-end-to-end Modulation for Embedded Networks[J]. IEEE Circuits and Devices Magazine, 2006, 22(5): 48-55.

[2] WANG J P, CAO Y, SHI Y Z. Analysis of IOT Software Industrial Chain[J]. China Soft Science, 2011(8): 27-32. (in Chinese)
王建平，曹洋，史一哲. 物联网软件产业链研究[J]. 中国软科学，2011(8)：27-32.

[3] GUBBI J, BUYYA R, MARUSIC S, et al. Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions [J]. Future Generation Computer Systems, 2013, 29(7): 1645-1660.

[4] 刘云浩. 物联网导论[M]. 北京：科学出版社，2010.

[5] FITZGERALD J, GAMBLE C, LARSEN P G, et al. Cyber-Physical Systems Design: Formal Foundations, Methods and Integrated Tool Chains [C] // Proceedings of the Third FME Workshop on Formal Methods in Software Engineering. IEEE, 2015: 40-46.

[6] LIU C, HUANG R R, ZHANG W, et al. Analyzing Software Requirements for Cyber Physical Systems[J]. Chinese Journal of Computers, 2016, 39(11): 2344-2354. (in Chinese)
刘春，黄冉冉，张伟，等. 信息物理融合系统的软件需求分析[J]. 计算机学报，2016，39(11)：2344-2354.

[7] ATZORI L, IERA A, MORABITO G, et al. The Social Internet of Things (SIoT)—When Social Networks Meet the Internet of Things: Concept, Architecture and Network Characterization [J]. Computer Networks, 2012, 56(16): 3594-3608.

[8] NITTI M, GIRAU R, ATZORI L. Trustworthiness Management in the Social Internet of Things[J]. IEEE Transactions on Knowledge and Data Engineering, 2014, 26(5): 1253-1266.

[9] WANG R Z, SHI T X, JIAO W P. Collaborative Sensing Mechanism for Intelligent Sensors Based on Tuple Space[J]. Journal of Software, 2015, 26(4): 790-801. (in Chinese)
王睿智，史庭训，焦文品. 一种基于元组空间的智能传感器协同感知机制[J]. 软件学报，2015，26(4)：790-801.

[10] CHEN H M, CUI L. Design and Model Checking of Service Oriented Software Architecture for Internet of Things: A Survey [J]. Chinese Journal of Computers, 2016, 39(5): 853-871. (in Chinese)
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.

[11] XIE K B, CHEN H M, CUI L. PMDA: A Physical Model Driven Software Architecture for Internet of Things[J]. Computer Research and Development, 2013, 50(6): 1185-1197. (in Chinese)
谢开斌，陈海明，崔莉. PMDA：一种物理模型驱动的物联网软件体系结构[J]. 计算机研究与发展，2013，50(6)：1185-1197.

[12] FORTINO G, GUERRIERI A, RUSSO W. Agent-oriented Smart Objects Development [C] // International Conference on Computer Supported Cooperative Work in Design. IEEE, 2012: 907-912.

[13] DUNKELS A. The uIP Embedded TCP / IP Stack [EB/OL]. [2006-06-01]. http: www. sics. se.

[14] ZAMBONELLI F. Key Abstractions for IoT-Oriented Software Engineering[J]. IEEE Software, 2017, 34(1): 38-45.

[15] TAIVALSAARI A, MIKKONEN T. A Roadmap to the Programmable World: Software Challenges in the IoT Era[J]. IEEE Software, 2017, 34(1): 72-80.

[16] GELERNTER D. Generative Communication in Linda[J]. ACM Computing Surveys, 1985, 7(1): 80-112.

[17] BENCHI A, LAUNAY P, GUIDEC F. A P2P tuple space implementation for disconnected MANETs[J]. Peer-to-Peer Networking and Applications, 2015, 8(1): 87-102.

[18] LI Q S, WANG L, CHU H, et al. Agent-Based Software Adaptive Dynamic Evolution Mechanism [J]. Journal of Software, 2015, 26(4): 760-777. (in Chinese)
李青山，王璐，褚华，等. 一种基于智能体技术的软件自适应动态演化机制[J]. 软件学报，2015，26(4)：760-777.

[19] ANSOLA P G, GARCIA A, MORENAS J D L. IoT Visibility Software Architecture to Provide Smart Workforce Allocation [M] // Service Orientation in Holonic and Multi-Agent Manufacturing. Springer International Publishing, 2016: 223-231.