基于实时UML顺序图的物联网交互模型

丛新宇[1,2] 虞慧群[1]

(华东理工大学计算机科学与工程学院 上海 200237)[1] (上海计算机软件评测重点实验室 上海 201112)[2]

摘 要 物联网是一个集计算、通信和控制于一体的智能系统,它通过监控和收集物理进程信息并将这些信息进行计算和分析,最终生成正确的控制指令用以执行,从而使物理环境变得更加安全和可靠。在物联网中,各物体通过网络连接或者本地连接的方式进行交互,这些交互具有时间性和地域性。物联网的建模和验证是物联网研究中一个重要的领域。文中提出一种基于实时UML顺序图的物联网交互模型,该模型将物联网中所有参与交互的物体建模为交互对象,并且通过实时UML顺序图对交互对象间的交互进行建模。使用时间自动机对交互对象的内部状态变化进行建模,以形成对交互模型的补充。最后根据转换规则将交互模型转换为时间自动机的形式以便于验证。通过一个实例,显示了如何具体应用物联网交互模型。进一步提出了物联网系统应该满足的一些性质,并使用UPPAAL模型检测工具对物联网交互模型进行分析和验证。

关键词 物联网,交互模型,实时UML顺序图,时间自动机,验证

中图法分类号 TP31 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.11.016

Interaction Model for Internet of Things Based on Real-time UML Sequence Diagram

CONG Xin-yu[1,2] YU Hui-qun[1]

(Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China)[1]

(Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai 201112, China)[2]

Abstract Internet of things (IOT) is an intelligent system which integrates computation, communication and control. It aims at monitoring the behavior of physical processes, analyzing information to generate correct instructions, and actuating actions to make physical environment work correctly and better. In Internet of things, different things interact locally or through network connections. These interactions are affected by time and locations. Modeling and verification of Internet of things are an important area in the research of Internet of things. This paper proposed an interaction model for IOT based on real-time UML sequence diagram. In our model, all things that interact in IOT are modeled as interactive objects between which interactions are modeled by real-time UML sequence diagram. And timed automata is used to model the internal state changes of interactive objects, which results in a complement to the interaction model. Finally according to the transition rules, the interaction model was transformed to the form of timed automata for the sake of verification. A case study was used to show how to specifically apply the interaction model. Further some properties which IOT should satisfy were presented. And the model checking tool UPPAAL was used to analyze and verify the interaction model for IOT.

Keywords Internet of things, Interaction model, Real-time UML sequence diagram, Timed automata, Verification

1 引言

随着传感器网络和嵌入式系统技术的发展,智能系统在人们生活中应用越来越普遍。其中由MIT Auto-ID Center提出的物联网(Internet of things)[1]被誉为继计算机和互联网之后的第三次信息技术革命。物联网作为一个集计算、通信和控制于一体的智能系统,通过监控和收集物理进程信息,并且将这些信息进行计算和分析,最终生成正确的控制指令用以执行,从而使物理环境变得更加安全和可靠[2,3]。如图1所示,物联网体系结构可分为4个层次,分别为:

• 环境层:环境层包括所有和物联网进行交互的物理实体,这些实体具有不同属性,这些属性被称为物理环境的信息。

• 设备层:设备层由具有信息收发功能的物理节点构成,其中一些节点除基本信息收发功能外还具有特殊功能。这些节点主要包括感知器和触发器。感知器能够收集环境信息,常见的感知器有RFID、摄像头、GPS等。触发器能够通过接收指令信息并执行来改变环境,常见的触发器有空调、灯具、加湿器等。

• 网络层:网络层能够传输信息,实现物联网各个部分的

到稿日期:2013-09-16 返修日期:2013-11-16 本文受国家自然科学基金(61173048,61073107)资助。

丛新宇(1986—),男,博士生,主要研究方向为物联网建模验证,E-mail:congcong1986good@126.com;虞慧群(1967—),男,博士,教授,主要研究方向为形式化方法、可信计算,E-mail:yhq@ecust.edu.cn(通信作者)。

互连。信息包括：感知器感知到的物理环境信息、计算机处理后生成的控制指令信息，以及其他信息诸如设备状态信息等。

• 控制层：控制层主要由计算机构成，其功能是把网络层传输来的环境信息进行分析和处理，做出正确的控制和决策，实现智能化的管理、应用和服务。

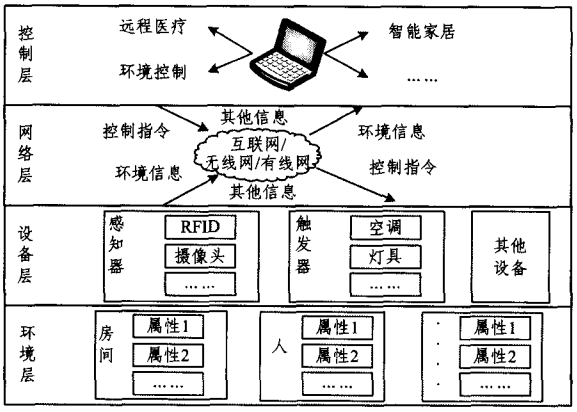


图 1　物联网体系结构

在物联网中，不同物体通过网络或本地连接的方式进行实时信息交换和通信，实现物与物、物与人的互连，以达到智能化识别、监控和管理的目的。在设计物联网系统时，通过对不同物体间交互的描述，物联网各部分的静态联系、系统的动态行为以及物联网自身的需求诸如实时性等能够得以体现。因此，着重体现物联网各部分间的交互成为了物联网建模方法中的一个方向。文中提出一种以交互为中心的物联网建模方法，该方法从交互的角度将物联网中参与交互的物体建模为交互对象，并使用实时 UML 顺序图[4,5]对各交互对象间的交互进行描述。由于实时 UML 顺序图不适合刻画不受交互影响的交互对象内部状态变化，文中使用时间自动机[6]对其建模，对交互模型进行补充，以得到完整的物联网交互模型。最后将实时 UML 顺序图描述的交互模型转换为时间自动机的形式，并结合时间自动机描述的交互对象内部状态变化，使用 UPPAAL 验证系统性质。整个建模与验证过程如图 2 所示。

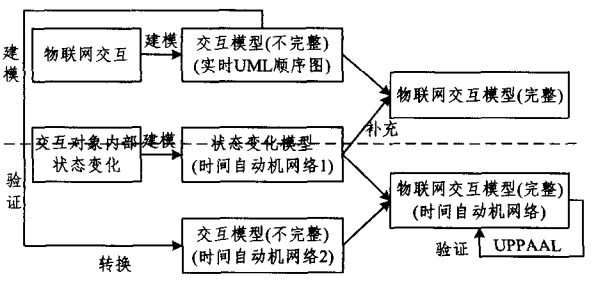


图 2　物联网建模与验证

本文第 2 节简单介绍物联网交互模型所使用的建模语言，包括实时 UML 顺序图和时间自动机，并对实时 UML 顺序图转换时间自动机网络的规则进行描述；第 3 节主要对物联网交互进行分析并提出建模方法，最后讨论物联网交互模型的验证；第 4 节以智能家居温度控制系统为例，说明如何具体建模，并使用 UPPAAL 对系统性质进行验证；第 5 节总结全文。

2　物联网建模语言

2.1　实时 UML 顺序图

文中使用实时 UML 顺序图对物联网交互进行建模。实时 UML 顺序图是一个表示交互的二维图。纵向是时间轴，自上而下表示交互执行的先后顺序。横向表示参与交互的对象，不同对象以不同命名的方框代表。每个方框下都延伸有一条直线，称为对象的生命线，用于表示对象存在的时间。对象间的交互可用消息表示，其显示为从一个对象生命线出发至另一个对象生命线的箭头，箭头的指向表示消息的传递方向。下面对实时 UML 顺序图的语法和语义进行简单介绍。

A. 语法

一个实时 UML 顺序图可表述为以下形式：

$\mu ::= (Frame, Clocks)$，

其中，$Clocks$ 表示顺序图的时钟集合，$Frame$ 表示顺序图中所有对象的行为单元，其定义如下：

$$Frame ::= (r_1 : A_1, \cdots, r_n : A_n) \mid Frame; Frame \mid if(g, Frame) \mid alt(g_1, Frame, g_2, Frame) \mid loop(g, Frame) \mid par(Frame, Frame)$$

其中，$r_i (1 \leqslant i \leqslant n)$ 表示顺序图的对象，A_i 表示对象 r_i 的行为，其定义如下：

$A ::= nil \mid v := z \mid send(i, m) \mid recv(i, m) \mid delay(d) \mid reset(x) \mid A; A$。其中 $d, z \in Z^+$ 表示正整数，$x \in$ Clocks，$i \in \{1, \cdots, n\}$ 表示标识，$v \in V$ 表示交互对象的数据变量，m 表示消息。

B. 语义

$Frame; Frame$ 表示不同 $Frame$ 的顺序执行，即不同 $Frame$ 中相同对象行为的顺序执行；if$(g, Frame)$表示 $Frame$ 的条件执行，当条件 g 为真时，$Frame$ 执行，否则 $Frame$ 不执行；$alt(g_1, Frame_1, g_2, Frame_2)$ 表示不同 $Frame$ 的选择执行，当 g_1 为真时，$Frame_1$ 执行，当 g_2 为真时，$Frame_2$ 执行，当 g_1 和 g_2 都为空时，表示对 $Frame_1$ 和 $Frame_2$ 的不确定执行；$loop(g, Frame)$表示 $Frame$ 的条件循环执行，当条件 g 为真时，$Frame$ 执行，否则 $Frame$ 停止执行；$par(Frame, Frame)$表示不同 $Frame$ 的并发执行。

nil 表示空操作；$v := z$ 表示由于交互所导致的交互对象数据变量 v 的值变化；$send(i, m)$和 $recv(i, m)$分别表示将消息 m 发送给对象 r_i 和从对象 r_i 接收消息 m。消息收发也可用来传递变量值，当 m 表示传递变量 v 的值时，$send(i, m)$和 $recv(i, m)$ 分别改变形式为 $send(i, val(v_s))$ 和 $recv(i, val(v_r))$；$delay(d)$表示延迟 d 个单位时间；$reset(x)$表示将时钟 x 重置为 0；$A; A$ 表示对象行为的顺序执行。

实时 UML 顺序图的操作语义包括以下 2 类：空变迁和动作变迁。空变迁具有一般形式：$(F, x) \longrightarrow (F', x) (F, F' \in Frame; x \in Clocks)$，该变迁执行不改变时钟值。动作变迁具有一般形式：$(F, x) \xrightarrow{act} (F', x') (F, F' \in Frame; x, x' \in Clocks; act \in A)$，该变迁执行可能改变时钟值。空变迁和动作变迁的具体规则可参考文献[7]。

2.2　时间自动机

定义 1(时间自动机)　一个时间自动机是一个多元组 $(L, l_0, Chan, \sum, V, C, E, Inv)$，其中 L 是有限状态集；$l_0 \in L$ 是初始状态；$Chan$ 是通道集；$\sum = \{\alpha? \mid \alpha \in Chan\} \cup \{\alpha! \mid \alpha \in Chan\} \cup \{\tau\}$是动作集合，包括输入动作 $\alpha?$、输出动作 $\alpha!$和内

部动作 τ；V 是有限变量集；C 是有限时钟集($C\cap V=\emptyset$)；$E\subseteq L\times\sum\times L\times 2^C\times G$ 是转换规则集，一条转换规则 $e=(l,\alpha,l',R,g)$ 表示通过执行动作 α，系统从状态 l 转换到状态 l'，集合 R 包含状态转换后需要重置的变量和时钟，g 为转换发生的条件；Inv 是一个函数，功能为赋予单个状态一个不变式。

定义 2(时间自动机网络) 一个时间自动机网络 $NTA=TA_1|\cdots|TA_i(1<i\leqslant n)$ 由 n 个时间自动机构成，这些时间自动机共享一些时钟变量和数据变量，但都有各自的状态。

2.3 实时 UML 顺序图转换时间自动机网络

总体来说，将实时 UML 顺序图转换为时间自动机网络时，对象转换为时间自动机，对象的行为可转换为时间自动机的动作。转换规则[7]如表 1 所列。

表 1 转换规则

对象行为转换			
nil	$init_{nil}=end_{nil}$	v:=z, z∈Z^+	$init_{v:=z}$ v:=z $end_{v:=z}$
send(i,m)/send(i,val(v_s)),c_m∈Chan	$init_{send}$ $c_m!$ end_{send}	recv(i, val(v_r)),c_m∈Chan	$init_{recv}$ $c_m?$ $v_r:=v_s$ end_{recv}
recv(i,m), c_m∈Chan	$init_{recv}$ $c_m?$ end_{recv}	reset(x)	$init_{reset(x)}$ x:=0 $end_{reset(x)}$
delay(d), d==d_1	$init_{delay(d)}$ x<=d_1 x==d_1 $end_{delay(d)}$	delay(d), $d_1\leqslant d\leqslant d_2$	$init_{delay(d)}$ x<=d_2 x>=d_1 $end_{delay(d)}$
$A_1;A_2$	$init_{A1}$ ... end_{A1} $init_{A2}$ ... end_{A2}		
Frame 转换			
(r_1:A_1,..., r_n:A_n)	TA(A_1) ..... TA(A_n)		
$F_1;F_2$	对象i i=1,…,n TA(A_{1i}) $init_{A1i}$ ... end_{A1i} TA(A_{2i}) $init_{A2i}$ ... end_{A2i}		
if(g, F)	对象i i=1,…,n g==ture TA(A_i) $init_{Ai}$ ... end_{Ai}		
alt(g_1,F_1, g_2,F_2)	对象i i=1,…,n g_1==ture TA(A_{1i}) $init_{A1i}$ end_{A1i}=end_{A2i} g_2==ture TA(A_{2i}) $init_{A2i}$		
loop(g, F)	对象i i=1,…,n g==ture TA(A_i) $init_{Ai}$ ... end_{Ai}		
par(F_1,F_2)	对象i i=1,…,n TA(A_{1i}) TA(A_{2i})		

由于实时 UML 顺序图和时间自动机都可根据其操作语义被定义为标识变迁系统(labeled transition system)，转换规则的正确性可通过证明实时 UML 顺序图和其对应的时间自动机网络中的任何轨迹可相互模拟来进行验证。首先给出标识变迁系统中轨迹的定义。

定义 3(轨迹) 在标识变迁系统中，系统轨迹由一系列变迁 $t_1\cdot t_2\cdot\cdots\cdot t_n$ 组成，其中：

· $t_i(i\in\{1,\cdots,n\})$是一条轨迹。

· 若 s 是一条轨迹，则 $s\cdot t_i(i\in\{1,\cdots,n\})$是一条轨迹。

然后给出定理 1，即实时 UML 顺序图和其对应的时间自动机网络中的任何轨迹可相互模拟。

定理 1 通过转换规则将一个实时 UML 顺序图转换为其对应的时间自动机网络，假设 s 是实时 UML 顺序图的一条轨迹，那么在对应的时间自动机网络中一定存在一条轨迹 t，使 $\varphi_1(s)=\varphi_2(t)$，反之亦然。其中：

$$\varphi_1(s)=\begin{cases}s, & \text{if } s\in\{delay(d)\mid d\in Z^+\}\cup\{send(i,m),recv(i,m)\mid i\in(1,\cdots,n)\}\\ s_1\cdot\varphi_1(s_2), & \text{if } s=s_1\cdot s_2,s_1\in\{delay(d)\mid d\in Z^+\}\cup\{send(i,m),recv(i,m)\mid i\in(1,\cdots,n)\}\\ \varphi_1(s_2), & \text{if } s=\tau\cdot s_2,\tau\in\{reset(x)\mid x\in Clocks\}\cup\{v:=z\mid v\in V,z\in Z^+\}\end{cases}$$

$$\varphi_2(t)=\begin{cases}t, & \text{if } t\in Z^+\cup\sum,t\neq\tau\\ t_1\cdot\varphi_2(t_2), & \text{if } t=t_1\cdot t_2,t_1\in Z^+\cup\sum,t_1\neq\tau\\ \varphi_2(t_2), & \text{if } t=\tau\cdot t_2\end{cases}$$

定理 1 的证明可参考文献[7]。

3 物联网交互模型

本节主要对物联网的交互进行分析并且建模，最后讨论物联网交互模型的验证。

3.1 物联网交互特点

根据物联网的特点，对物联网的交互进行建模时需体现以下特点：

并发性：并发是物理环境中事件发生的基本特征。在物联网中，各物体之间的不同交互可能以并行的方式进行。

实时性：在物联网中，由于计算过程和物理进程相互交互，时间因素不可抽象。实时性是体现物联网时间因素的一种性质，它指物联网中交互对象的行为必需在一定期限内完成。

分布性：在物联网中，各物体通常分布在不同地点。处于不同位置的物体间的交互方式各不相同。

移动性：在物联网中，物体的位置是动态变化的。物体位置的变化会对物体间的交互方式造成影响。

3.2 物联网交互建模

A. 交互对象

在物联网中，不同物体通过网络连接或者本地连接的方式进行交互，这些参与交互的物体在交互模型中被建模为交互对象。如图 3 所示，根据第 1 节中描述的物联网体系结构，交互对象可被归纳为以下几类：

(1)控制类：控制类是控制层的抽象表示，控制类对象把传输来的环境信息进行分析和处理，生成相应的指令，并且将指令发送给设备类对象。计算机类是一个常见的控制类子类。

(2)设备类：设备类是设备层的抽象表示，它包括所有能够进行信息传输的物理节点。其子类主要有感知器类和触发

器类。感知器类包括所有监控环境的设备，这些设备定期收集环境信息并且将信息发送给控制类对象；触发器类包括所有改变环境的设备，这些设备接收指令以执行，使物理环境变得更加安全可靠。

(3)环境类：环境类是环境层的抽象表示，它包括物联网环境中的所有物理实体。环境类按照其参与物联网交互中的角色可被分为 3 种子类：被动实体类、主动实体类以及混合实体类。被动实体类包括所有在物联网交互过程中被动提供环境信息的物理实体；主动实体类包括所有在物联网交互过程中主动发送指令的物理实体；有些物理实体既能够被动提供环境信息，也能够主动发送指令，这类物理实体被归属为混合实体类。

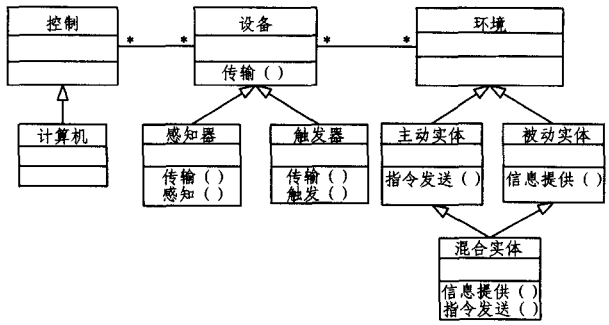


图 3　物联网交互对象类图

B. 基本交互

定义 4(基本交互)　如图 4 所示，一个基本的物联网交互可建模为：主动实体类对象，触发器类对象，控制类对象，感知器类对象，环境类对象。首先感知器类对象从环境类对象获取信息并且将信息作为数据发送给控制类对象，然后控制类对象对数据进行处理以生成指令并且将指令发送给触发器类对象，最终触发器类对象按照指令执行相应动作。与此同时，物联网环境中可能有主动实体类对象通过并行的方式向触发器类对象发送指令，触发器类对象收到指令后执行相应动作。模型描述如下：

$$BasicInteract = par(F_1, F_2)$$

$$F_1 = (e:A_{11}, s:A_{12}, c:A_{13}, a:A_{14}, p:nil)$$

$$A_{11} = recv(s, obtainInfo); send(s, val(info)); nil$$

$$A_{12} = send(e, obtainInfo); recv(e, val(info)); send(c, val(data)); nil$$

$$A_{13} = recv(s, val(data)); send(a, ins); nil$$

$$A_{14} = recv(c, ins); nil$$

$$F_2 = (e:nil, s:nil, c:nil, a:A_{24}, p:A_{25})$$

$$A_{24} = recv(p, ins'); nil$$

$$A_{25} = send(a, ins'); nil$$

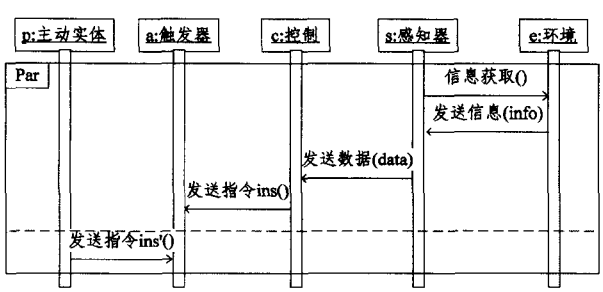


图 4　物联网基本交互

C. 交互时间

物联网作为一个实时系统，时间因素在其交互过程中应该得以体现。通过对物联网交互过程中时间因素的描述，与时间相关的性质诸如实时性能够进行验证。在实时 UML 顺序图中，通常用时间戳来表示消息执行的时刻。如图 5 所示，时间戳 t 表示物联网交互中消息开始执行的时刻，时间戳 t_next 表示该消息再次执行的时刻。

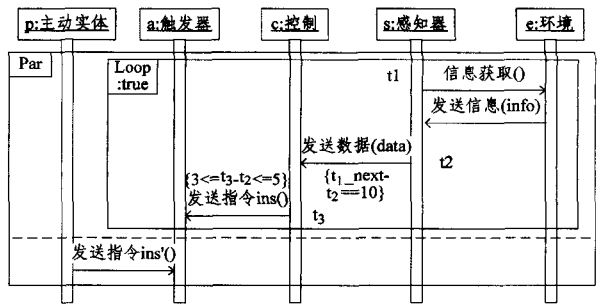


图 5　带有时间因素的交互

在物联网交互过程中，时间因素可分为两类：确定时间和不确定时间。确定时间表示时间的取值是确定的，如图 5 中 $\{t_1\_next - t_2 == 10\}$ 表示每隔 10 个单位时间，感知器获取环境信息并且将信息发送给控制器。不确定时间表示时间的取值在一个范围内，是不确定的。如图 5 中 $\{3 \leqslant t_3 - t_2 \leqslant 5\}$ 表示控制器收到数据后，其处理数据并生成指令的时间至少需要 3 个单位时间，至多不超过 5 个单位时间。以图 5 为例，带有时间因素的物联网交互模型描述如下：

$$TimedInteract = par(F_1, F_2)$$

$$F_1 = loop(true, (e:A_{11}, s:A_{12}, c:A_{13}, a:A_{14}, p:nil))$$

$$A_{11} = recv(s, obtainInfo); send(s, val(info)); nil$$

$$A_{12} = send(e, obtainInfo); recv(e, val(info)); send(c, val(data)); reset(x); delay(10); nil$$

$$A_{13} = recv(s, val(data)); reset(x); delay(d); send(a, ins); nil;$$

$$3 \leqslant d \leqslant 5$$

$$A_{14} = recv(c, ins); nil$$

$$F_2 = (e:nil, s:nil, c:nil, a:A_{24}, p:A_{25})$$

$$A_{24} = recv(p, ins'); nil$$

$$A_{25} = send(a, ins'); nil$$

D. 交互地点

在物联网中，交互对象往往处于不同的地点。地点因素在物联网交互中主要体现在两个方面：(1)物联网交互对象地点的变化。(2)交互对象处于不同地点关系时对物联网交互的影响。方面(1)将在 3.2.E 节中进行阐述，本节主要对方面(2)进行说明。

定义 5(地点关系)　在物联网中不同对象之间的地点关系可定义为 $R(L_S, L_D, n) = \{r(L_S, L_D, n) \mid r(L_S, L_D, n) \in Local \cup Network \cup OutofConnect\}$。如图 6 所示，$L_S$，$L_D$ 分别表示源对象 S 和目的对象 D 的地点；n 表示除源对象和目的对象外交互所需要的中间物理节点的个数，这些节点负责交互信息的传递；$Local$ 表示本地交互，其交互对象处于同一地点，不需要其他物理节点进行信息传递；$Network$ 表示位于不同地点的交互对象通过网络连接的方式进行交互，除交互源对象和交互目的对象外，其中进行信息传输的物理节点有 n 个；N 为最大连接中间节点数，其定义如定义 6 所述。

$$\text{Local} = \{r(L_S, L_D, n) \mid L_S = L_D, n = 0\}$$
$$\text{Network} = \{r(L_S, L_D, n) \mid L_S \neq L_D, 0 \leq n \leq N\}$$
$$\text{OutofConnect} = \{r(L_S, L_D, n) \mid L_S \neq L_D, N<n<+\infty\}$$

图 6　交互地点关系

定义 6(最大连接中间节点数 N)　存在源对象 S 和目的对象 D,在其处于所有能够网络连接的地点关系中,N 表示表示除源对象和目的对象外交互所需要的中间物理节点个数的最大值。即 $N=\max\{n \mid \exists L_S, L_D, \forall r(L_S, L_D, n) \in Network\}$。

OutofConnect 表示位于不同地点的交互对象之间没有足够的物理节点进行网络连接,无法交互,不等式 $N<n<+\infty$ 表示交互对象所处地点间的距离超过了网络连接所能覆盖的最大范围。

定义 7(地点关系函数)　地点关系函数 $Loc(L_S, L_D, r(L_S, L_D, n))=Mid$ 可用来具体量化不同的地点关系,其中 L_S, L_D 分别表示源对象 S 和目的对象 D 的地点,n 表示除 S 和 D 外交互所需要的中间节点的个数,$r(L_S, L_D, n) \in R(L_S, L_D, n)$ 表示 S 和 D 之间的地点关系,Mid 表示除 S, D 外进行信息传输的中间节点集合,Mid 可为空。并且根据定义 5,地点关系函数可将地点关系表述为:

$$Loc(L_S, L_D, r(L_S, L_D, n)) = \begin{cases} \varnothing, & \text{if } r(L_S, L_D, n) \in Local \text{ or if } r(L_S, L_D, n) \in Network \wedge n == 0 \\ \{mid_1\}, & \text{if } r(L_S, L_D, n) \in Network \wedge n == 1 \\ \{mid_1, \cdots, mid_n\}, & \text{if } r(L_S, L_D, n) \in Network \wedge 1<n \leqslant N, N \geqslant 2 \\ \{mid_1, \cdots, mid_n \mid N<n<+\infty, n \neq 1\}, & \text{if } r(L_S, L_D, n) \in OutofConnect \end{cases}$$

不同的地点关系对交互对象带来的影响各不相同,根据定义 7,图 7 显示了对象 S 和对象 D 分别处于不同的地点关系时的交互,其中 m 表示需要传递的消息。其模型描述如下:

$LocInteract = alt(g_1, F_1, g_2, F_2, g_3, F_3, g_4, F_4)$

$g_1 := Loc(L_S, L_D, r(L_S, L_D, n)) == \varnothing$

$F_1 = (S:A_{1S}, D:A_{1D}, mid_1:nil, \cdots, mid_n:nil)$

$A_{1S} = send(D, m); nil$

$A_{1D} = recv(S, m); nil$

$g_2 := Loc(L_S, L_D, r(L_S, L_D, n)) == \{mid_1, \cdots, mid_n\} \wedge 1<n \leqslant N, N \geqslant 2$

$F_2 = (S:A_{2S}, D:A_{2D}, mid_1:A_{21}, \cdots, mid_n:A_{2n})$

$A_{2S} = send(mid_1, m); nil$

$A_{2D} = recv(mid_n, m); nil$

$A_{21} = recv(S, m); send(mid_2, m); nil$

$A_{2k} = recv(mid_{k-1}, m); send(mid_{k+1}, m); nil$

$k \in \{2, \cdots, n-1\}$

$A_{2n} = recv(mid_{n-1}, m); send(D, m); nil$

$g_3 := Loc(L_S, L_D, r(L_S, L_D, n)) == \{mid_1\}$

$F_3 = (S:A_{3S}, D:A_{3D}, mid_1:A_{31}, mid_2:nil, \cdots, mid_n:nil)$

$A_{3S} = send(mid_1, m); nil$

$A_{3D} = recv(mid_1, m); nil$

$A_{31} = recv(S, m); send(D, m); nil$

$g_4 := Loc(L_S, L_D, r(L_S, L_D, n)) == \{mid_1, \cdots, mid_n \mid N<n<+\infty, n \neq 1\}$

$F_4 = (S:nil, D:nil, mid_1:nil, \cdots, mid_n:nil)$

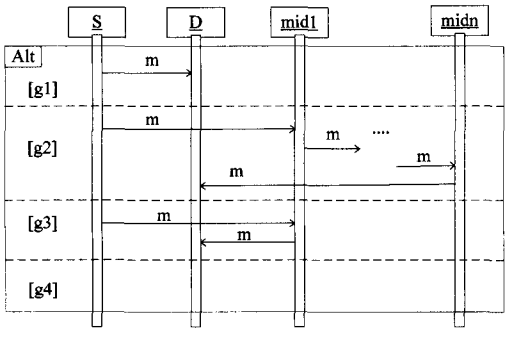


图 7　不同地点关系交互

E. 交互对象状态

在物联网交互过程中,交互对象的状态可能发生变化。这些变化包括两种:(1)交互对象的状态不受交互影响发生的内部变化,比如交互对象的地点随时间发生变化。(2)交互对象在交互过程中执行行为后所发生的状态变化,比如交互对象接收指令后发生的状态变化。由于这些变化会对交互造成影响或者受到交互影响,因此需要使用形式化语言对交互对象的状态变化进行描述。第 2 种状态变化由于受到交互影响,其描述隐含在实时 UML 顺序图所描述的交互模型中。对于第 1 种状态变化,由于实时 UML 顺序图不适合描述交互对象的内部状态变化,可使用时间自动机对其进行建模。

F. 建模步骤

通过以上对物联网交互的分析,可对物联网交互模型建模过程进行细化,如图 8 所示,建模过程包括以下步骤:

(1)分析参与物联网交互的物体,建模为交互对象。

(2)根据交互对象的类别和定义 4,使用实时 UML 顺序图对物联网的基本交互进行建模。

(3)分析交互模型中交互对象间的地点关系,根据定义 5、定义 7,对交互模型进行修改。

(4)根据物联网系统的时间约束,在交互模型中添加相应的时间戳描述时间约束。

(5)使用时间自动机对交互对象的内部状态变化进行建模。

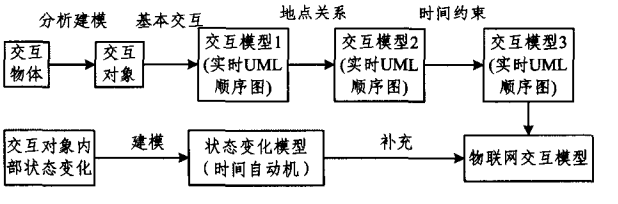


图 8　交互建模过程

3.3　物联网交互验证

根据转换规则将实时 UML 顺序图描述的交互模型转换形式为时间自动机网络 NTA_1,并且根据定义 2,结合描述交互对象内部状态的时间自动机网络 NTA_2,可得到时间自动机网络 $NTA = TA_{11} \mid \cdots \mid TA_{1i} \mid TA_{21} \mid \cdots \mid TA_{2j} (1<i \leqslant n, 1<j \leqslant m)$。NTA 可描述完整的物联网交互模型,其中 $TA_{11} \mid \cdots \mid TA_{1i} = NTA_1$ 用以描述物联网交互,$TA_{21} \mid \cdots \mid TA_{2j} = NTA_2$

用以描述物联网交互对象的内部状态变化。

得到时间自动机网络 NTA 后，可以使用 UPPAAL 对其进行验证。UPPAAL 作为一个集成的工具环境，允许用户根据系统状态和状态间的变迁对系统行为进行建模、模拟和验证[8,9]。文中使用 UPPAAL 协助完成以下工作：(1)系统运行过程的模拟：通过对系统运行过程的模拟，可以观察系统交互过程中各交互对象时钟变量和数据变量的变化。(2)系统性质的验证：通过 UPPAAL 中的时序逻辑公式表述系统的性质，进而使用 UPPAAL 提供的验证器对系统的性质进行验证。可验证的系统性质主要包括 3 类：

(1)系统可靠性：表示系统中不期望的事件永不发生，包括系统不会进入错误的状态，系统不出现死锁等性质。

(2)系统可达性：表示系统能否到达某个状态，即存在从初始状态开始到该状态的一条运行轨迹。

(3)系统时间性：表示系统是否满足某些时间约束，这类性质的描述中都包含有时钟变量。

4 案例分析

本节以智能家居温度控制系统为例，来具体说明如何应用交互模型对物联网进行建模，并且使用 UPPAAL 对模型进行验证。如图 9 所示，在房间内系统有一个感知器用于探测房间的温度，一台计算机用于计算生成控制指令，以及一台空调接收指令并调节房间的温度。房间外有两个位于不同地点的无线传输器负责进行信息的接收和发送，圆弧范围表示圆弧内的区域为无线传输器的网络连接所能覆盖的最大范围，比如范围表示无线传输器网络连接所能覆盖的最大范围。与此同时，住户通过其携带的手机对空调进行控制。

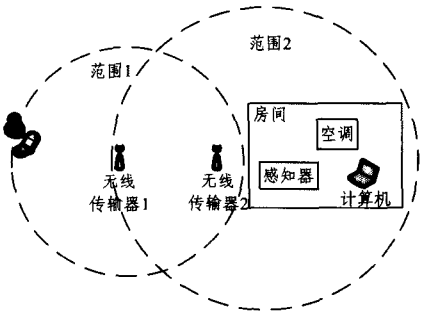


图 9 智能家居温度控制系统

根据 3.2 中 A 节将系统中所有参与交互的物体建模为交互对象，并且它们的行为描述如下：

住户 p：住户定期(每隔 5 个单位时间)通过手机向空调发送指令以打开空调，如手机反馈为指令发送成功，住户停止发送指令，否则继续发送指令。另一方面，住户不断向房间移动并且在使用手机发送指令的同时被动向手机提供当前所处的位置信息。假设住户移动到无线传输器 1 范围内需要 10 个单位时间，然后需要 20 个单位时间移动到无线传输器 2 范围内，最后需要 15 个单位时间移动到房间内。住户属于混合实体类。

手机 cp：手机接收住户指令，获取住户当前位置信息并向空调发送指令，然后将结果反馈给住户。手机属于设备类。

无线传输器 tr1，tr2：无线传输器功能为信息的接收与发送。无线传输器属于设备类。

空调 a：空调接收指令以执行动作。空调有关闭、打开、运行和休眠 4 种状态。其中运行和休眠两种状态为打开状态的后续状态。空调的初始状态为关闭或者打开。此外，空调接收到打开指令后将当前状态信息发送给感知器。空调属于触发器类。

计算机 c：计算机接收房间温度信息，根据预设的处理逻辑进行分析以生成指令，并且将指令发送给空调。处理逻辑为：如房间温度为高，计算机生成空调运行指令以降温；如房间温度为低，计算机生成空调休眠指令以节能。计算机处理信息的时间为 2 到 3 个单位时间。计算机属于控制类。

感知器 s：若空调被打开，感知器开始定期(每隔 5 个单位时间)获取房间的温度信息，并且将信息发送给计算机。感知器属于感知器类。

房间 r：房间被动向感知器提供其温度信息。房间的温度有高和低两种状态，假设每隔 30 个单位时间，房间温度依次在高和低之间变化，并且房间初始温度为高。房间属于被动实体类。

根据定义 4 和交互对象，对系统基本交互进行建模为 $((p,cp),a,c,s,r)$。由于住户 p 必须通过手机 cp 才能与空调 a 交互，因此可将住户和手机联合为 (p,cp) 以起到基本交互中主动实体类对象的作用。

对交互模型进行分析后，得到交互地点关系 $R(L_S,L_D,n)=\{r(L_s,L_c,n),r(L_a,L_s,n),r(L_c,L_a,n),r(L_s,L_r,n),r(L_p,L_{cp},n),r(L_{cp},L_a,n)\}$。根据定义 5，在 $R(L_S,L_D,n)$ 中分别有 $Network=\{r(L_s,L_c,0),r(L_a,L_s,0),r(L_c,L_a,0),r(L_{cp},L_a,0),r(L_{cp},L_a,1),r(L_{cp},L_a,2)\}$，$Local=\{r(L_p,L_{cp},0),r(L_s,L_r,0)\}$，$OutofConnect=\{r(L_{cp},L_a,n)\}$ $(n>2)$。文中使用 $Outtr$，$Intr1$，$Intr2$，Inr 分别标识地点关系 $r(L_{cp},L_a,n)$，$r(L_{cp},L_a,2)$，$r(L_{cp},L_a,1)$ 和 $r(L_{cp},L_a,0)$。标识 $Outtr$，$Intr1$，$Intr2$，Inr 分别表示住户位置在网络连接以外、无线传输器 1 范围内无线传输器 2 范围外、无线传输器 2 范围内房间外，以及房间内。

根据定义 7 使用地点关系函数描述地点关系：

$Loc(L_S,L_D,r(L_S,L_D,n))=\emptyset,r(L_S,L_D,n)\in\{r(L_p,L_{cp},0),r(L_s,L_r,0),r(L_s,L_c,0),r(L_a,L_s,0),r(L_c,L_a,0)\}$，

$Loc(L_{cp},L_a,Intr1)=\{tr_1,tr_2\}$，

$Loc(L_{cp},L_a,Intr2)=\{tr_2\}$，

$Loc(L_{cp},L_a,Inr)=\emptyset$，

$Loc(L_{cp},L_a,Outtr)=\{tr_1,\cdots,tr_n\mid 2<n<+\infty\}$

由此可将交互模型修改为 $((p,cp),tr_1,tr_2,a,c,s,r)$。然后根据系统交互对象行为描述中的时间约束，在交互模型中添加相应的时间戳。最终得到基于实时 UML 顺序图的交互模型如图 10 所示，描述如下：

$SmartHomeInteract=par(F_1,F_2)$

$F_1=loop(ACState==\text{true},(p:nil,cp:nil,tr_1:nil,tr_2:nil,a:nil,c:A_{11},s:A_{12},r:A_{13});alt(tem==hot,F_{11},tem==cool,F_{12}))$

$A_{11}=recv(s,val(tem));reset(x);delay(d);nil$ $2\leqslant d\leqslant 3$

$A_{12}=send(r,obtainTem);recv(r,val(tem));send(c,val(tem));reset(x);delay(5);nil$

$A_{13}=recv(s,obtainTem);send(s,val(tem));nil$

$F_{11}=(p:nil,cp:nil,tr_1:nil,tr_2:nil,a:recv(c,run);nil,c:send(a,run);nil,s:nil,r:nil)$

$F_{12}=(p:nil,cp:nil,tr_1:nil,tr_2:nil,a:recv(c,sleep);nil,c:send(a,sleep);nil,s:nil,r:nil)$

$F_2=loop(result==\text{false},(p:A_{21},cp:A_{22},tr_1:nil,tr_2:$

$nil, a: nil, c: nil, s: nil, r: nil); alt(loc == Intr1, F_{21}, loc == Intr2, F_{22}, loc == Outtr, F_{23}, loc == Inr, F_{24}))$

$A_{21} = send(cp, open); recv(cp, obtainLoc); send(cp, val(loc)); nil$

$A_{22} = recv(p, open); send(p, obtainLoc); recv(p, val(loc)); nil$

$F_{21} = (p: recv(cp, val(result)); reset(x); delay(5); nil, cp: send(tr1, open); recv(tr1, val(result)); send(p, val(result)); nil, tr_1: recv(cp, open); send(tr2, open); recv(tr2, val(result)); send(cp, val(result)); nil, tr_2: recv(tr1, open); send(a, open); result := true; send(tr1, val(result)); nil, a: recv(tr2, open); send(s, acon); nil, c: nil, s: recv(a, acon); ACState := 1; nil, r: nil)$

$F_{22} = (p: recv(cp, val(result)); reset(x); delay(5); nil, cp: send(tr2, open); recv(tr2, val(result)); send(p, val(result)); nil, tr_1: nil, tr_2: recv(cp, open); send(a, open); result := true; send(cp, val(result)); nil, a: recv(tr2, open); send(s, acon); nil, c: nil, s: recv(a, acon); ACState := 1; nil, r: nil)$

$F_{23} = (p: recv(cp, val(result)); reset(x); delay(5); nil, cp: result := false; send(p, val(result)); nil, tr_1: nil, tr_2: nil, a: nil, c: nil, s: nil, r: nil)$

$F_{24} = (p: recv(cp, val(result)); reset(x); delay(5); nil, cp: send(a, open); result := true; send(p, val(result)); nil, tr_1: nil, tr_2: nil, a: recv(cp, open); send(s, acon); nil, c: nil, s: recv(a, acon); ACState := 1; nil, r: nil)$

其中，tem 表示房间温度；loc 表示住户位置；ACState 表示空调状态(打开或关闭)；result 表示指令发送结果反馈；hot 和 cool 分别表示房间温度为高和低。

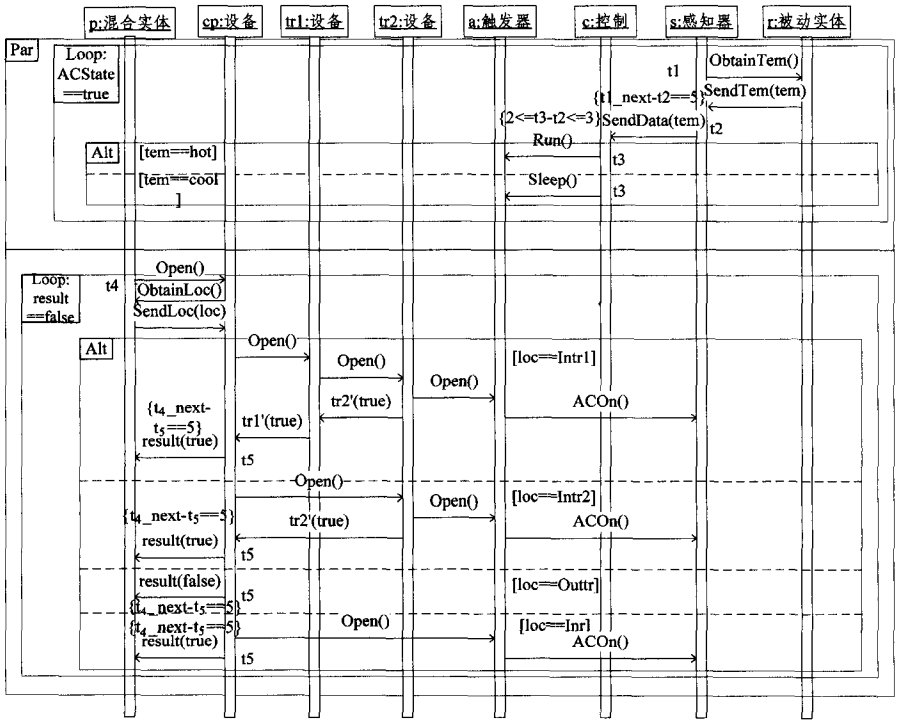


图 10　智能家居温度控制系统交互模型

通过时间自动机对系统中交互对象状态的内部变化建模，可对交互模型进行补充。系统中交互对象状态的内部变化包括：住户 p 的地点变化以及房间 r 的温度变化。使用时间自动机对这 2 种状态变化分别进行建模后，结果如图 11、图 12 所示。

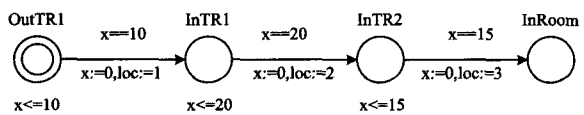


图 11　autP1 描述住户 p 的地点变化

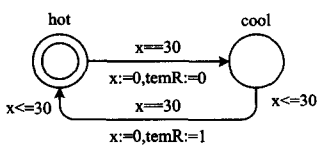
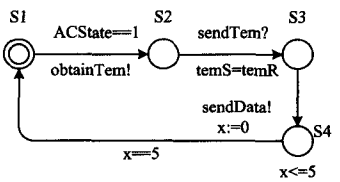
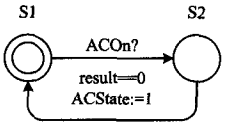


图 12　autR1 描述房间 r 的温度变化

然后根据 2.3 节所描述的转换规则将交互模型转换为时间自动机形式，如图 13－图 20 所示。



(a)autS1 描述感知器 s



(b)autS2 描述感知器 s

图 13

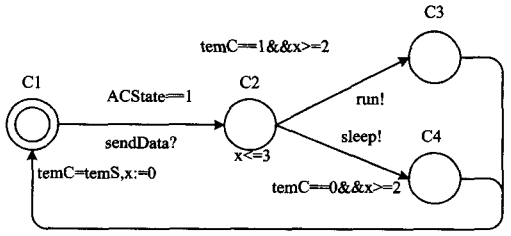
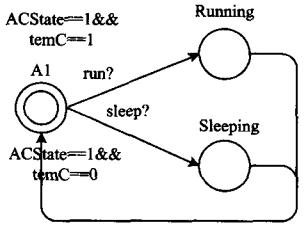
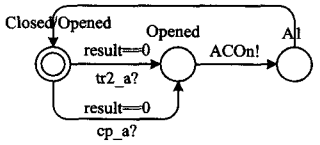


图 14　autC 描述计算机 c



(a)autA1 描述空调 a



(b)autA2 描述空调 a

图 15

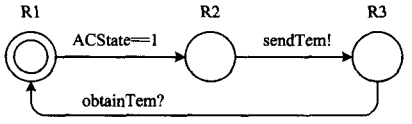


图 16　autR2 描述房间 r

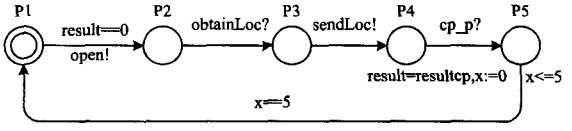


图 17　autP2 描述住户 p

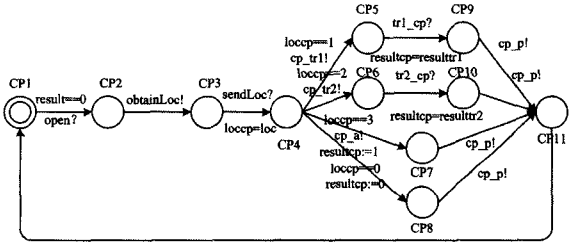


图 18　autCP 描述手机 CP

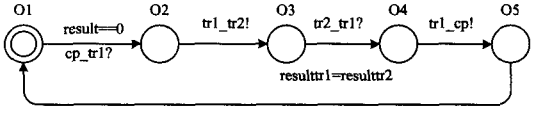


图 19　autTR1 描述无线传输器 tr1

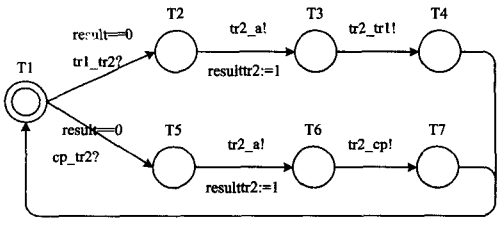


图 20　autTR2 描述无线传输器 tr2

按照 3.3 节中的描述得到完整的时间自动机网络后，可使用 UPPAAL 对物联网交互模型进行验证。首先验证一条重要的系统可靠性质：系统在运行过程中不出现死锁。将公式‘A[] not deadlock’输入到验证器进行检测，得到的结果为此性质满足。然后可对系统的可达性进行验证，譬如将公式‘E<> autA1. Sleeping’输入到验证器，可对性质：空调能够进入休眠状态进行验证，得到的结果为此性质满足。最后验证系统的时间性，性质描述为：房间温度变低后，空调在 2 个单位时间内进入休眠状态。但将公式‘E<> (autR1. cool && autR1. x<2 && autA1. Sleeping)’输入到验证器进行检测后，得到的结果为此性质不满足。因此进而验证性质：房间温度变低后，空调在 3 个单位时间内进入休眠状态，其公式表述为‘E<> (autR1. cool && autR1. x<3 && autA1. Sleeping)’。通过验证得到的结果为此性质满足，这说明房间温度变低后，空调进入休眠状态至少需要 2 个单位时间。

结束语　近年来，物联网成为了研究的热点[10]，关于物联网建模和验证方面有许多工作。文献[11]中提出了基于标识混合 petri 网(Labeled Hybrid Petri Net)对物联网进行建模和验证的方法。文献[12]使用有限状态进程 FSP(finite state processes)以及线形混合自动机(linear hybrid automata)对物联网行为进行建模，并且分别使用工具 LTSA(labeled transition system analyzer)和 PHAVer(polyhedral hybrid automaton verifier)对系统进行验证。

本文提出一种基于实时 UML 顺序图的物联网交互模型，通过实时 UML 顺序图对物联网中各物体之间的交互进行建模，以及补充使用时间自动机对物联网物体的内部状态变化建模，最终可以得到一个完整的物联网模型。与一些侧重于刻画物联网状态变化的建模方法[11,12]相比，文中提出的方法通过对物联网交互的直接描述，能够更直观地反映物联网中各物体之间的静态关系和交互过程中的动态行为，更细致地体现不同因素，如时间、地点等对物联网系统的影响。对物联网进行建模和验证是物联网研究的一个重要方面。随着当今社会物联网应用的逐渐普及，更多工作亟待解决。

参 考 文 献

[1] Sarma S, Brock D, David L, et al. The Networked Physical World [R]. Technical Report MIT-AUTOID-WH-001. Cambirdge, MIT, 1999

[2] Lee E A. Cyber Physical Systems: Design Challenges [C]// Object-Oriented Real-Time Distributed Computing. Orlando: IEEE, 2008: 363-369

[3] Sha Lui, Gopalakrishnan S, Liu Xue, et al. Cyber-physical systems: A new frontier [C]// Sensor Networks, Ubiquitous, and Trustworthy Computing. Taichung: IEEE, 2008: 1-9

[4] OMG UML. 2.0 Superstructure Specification [R/OL]. 2005-07-04 [2013-07-01]. http://www.uml.org

[5] Thomas Firely, Michaela Huhn, Karsen Diethers, et al. Timed Sequence Diagrams and Tool-Based Analysis [C]// The Unified Modeling Language. Berlin: Springer, 1999: 645-650

[6] Alur R, Dill D. A theory of timed automata [J]. Theoretical Computer Science, 1994, 126(2): 183-235

[7] Cambronero M-E, Valero V, Diaz G, et al. Verification of real-time systems design [J]. Software Testing, Verification & Reliability, 2010, 20(1): 3-37

[8] Frits Vaandrager. A First Introduction to UPPAAL [R/OL]. [2013-07-01]. http://www.mbsd.cs.ru.nl/publications/papers/fvaan/uppaaltutorial.pdf

[9] Behrmann G, David A, Larson K G, et al. A Tutorial on UPPAAL 4.0 [R/OL]. [2013-07-01]. http://www.it.uu.se/research/group/darts/papers/texts/new-tutorial.pdf

[10] Ohn H, Marburger J H, Kvamme E F, et al. Leadership Under Challenge: Information Technology R&D in a Competitive World [R/OL]. [2013-07-01]. http://ostp.gov/pdf/nitrd_review.pdf

[11] Thacker R A, Jones K R, Myers C J, et al. Automatic Abstraction for Verification of Cyber-Physical Systems [C]// Proceedings of the 1st IEEE International Conference on Cyber-Physical Systems. New York: ACM, 2010: 12-21

[12] Akshay R, Shangwen C, Schmerl B R, et al. An Architectural Approach to the Design and Analysis of Cyber-Physical Systems [J]. Electronic Communications of the EASST, 2009, 21

(上接第 78 页)

5.3 其他实验结果分析

实验结果中，出现页面白底白字的情况。通过分析页面源代码，发现原页面使用了不适合在移动设备上显示的 DOM 元素，而新页面将元素替换成适合在移动设备上显示的 DOM 元素。这导致原来对于该元素的 CSS 选择器失效。这一问题可以通过修改 CSS 文件解决。

实验结果中，也出现了某些相对路径未被替换为绝对路径的情况。通过分析页面源代码，发现这些元素是由 JavaScript 进行加载的，即当页面加载完成后才由 JavaScript 引入，所以工具未对这些相对路径进行转化。这一问题可以通过修改 JavaScript 文件中的相对路径解决。

5.4 实验结果分析

根据实验结果，MobiTran 的优势主要体现在以下方面：

(1)可以将原网页自动转化为适合移动设备宽度的新网页，清晰而无溢出。

(2)可以较大限度地维持新页面与原页面风格一致。

(3)在自动转化过程中可以较少地丢失页面信息。

(4)转化后的新页面相比原页面消耗数据流量较小。

根据实验结果，仍须完善的部分主要体现在以下方面：

(1)转化后的页面会出现元素与背景重叠。

(2)部分类型的 DOM 元素转换效果不好。

结束语 本文的主要工作包括：

(1)介绍应用背景及相关技术工具的选择，比较面向最终用户和面向开发者的 Web 应用移动版本转化工具的特点，分析了本文选择实现面向开发者的 Web 应用移动版本转化工具的原因。

(2)实现了面向开发者的 Web 应用移动版本转化工具并将其部署为在线应用转化服务，Web 应用开发者只需上传应用的 URL，该服务即可将其半自动化地转化为移动版本，并发布新的 URL 供移动设备访问。

(3)使用该服务对一些流行网站进行实例验证，并与现有的工具进行比较。

本文实现了面向开发者的移动版 Web 应用转化工具，未来工作主要集中于完善本工具：

当前工具仅支持最基本的自动转换。开发者进行编辑后，当原始 PC 版应用出现内容上的变化时，系统应可以将新内容映射到移动版 Web 应用对应位置；当原始 PC 版应用出现结构上的变化时，系统应可以提醒开发者重新手工编辑移动版应用。未来工作在自动转化和手工编辑阶段并不仅限于修改当前页面，而是记录一种映射规则，以此应对页面变化。

当前工具仅支持最基本的自动转换。未来工作可以对某种特定类型的网站给出一种自动转换模板，并支持内容自动匹配。

当前工具仅支持最基本的编辑。对于某一 Web 应用，会有很多内容页结构相同，内容不同。此时编辑某一页面，仅会对当前页面进行修改。未来工作可以给开发者更多选项，支持开发者选择仅修改当前页面，或修改本类页面。

参 考 文 献

[1] UC 浏览器功能介绍[OL]. UC 浏览器官方网站. http://www.uc.cn/browser/introduce.shtml

[2] 杨雪涛. 五大特色解析 UC 浏览器 8.0 安卓版评测[OL]. 手机中国. [2011-9-15]. http://www.cnmo.com/soft/110022.html

[3] 百度 SiteApp[OL]. 百度. http://siteapp.baidu.com/static/document.html

[4] Huang G, Wang D. Adapting user interface of service-oriented rich client to mobile phones[C]// 2011 IEEE 6th International Symposium on Service Oriented System Engineering (SOSE). IEEE, 2011: 140-145

[5] Chen Y, Xie X, Ma W Y, et al. Adapting web pages for small-screen devices[J]. Internet Computing, IEEE, 2005, 9(1): 50-56

[6] Nishiura K, Maezawa Y, Ishikawa F, et al. Supporting view transition design of smartphone applications using Web templates [C]//Proceedings of the 12th International Conference on Web Engineering. Springer Berlin Heidelberg, 2012: 323-331

[7] JavaScript[OL]. WiKiPedia. http://zh.wikipedia.org/wiki/Java Script

[8] Node.js[OL]. WiKiPedia. http://zh.wikipedia.org/wiki/Node.js

[9] BYvoid. Node.js 开发指南[M]. 北京：人民邮电出版社，2012

[10] jQuery UI. 百度百科. http://baike.baidu.com/view/2998196.htm

[11] GoMo[OL]. http://gomo.dudamobile.com/