LCESM:位置敏感的上下文事件订阅机制

刘 韩[1,2] 叶 剑[2] 朱珍民[2] 刘任任[1] 余 畅[1,2]
(湘潭大学信息工程学院 湘潭 411105)[1] (中国科学院计算技术研究所 北京 100090)[2]

摘 要 传统的位置敏感发布/订阅系统的研究集中于事件发布和匹配,不能很好地支持在同一事件源上频繁发生的上下文事件的订阅。给出一种位置敏感的上下文事件订阅机制,该机制包括位置敏感的上下文事件订阅语言LACL和动态绑定方法。基于LACL定义了位置事件和事件源事件,两者分别使系统具有位置感知能力和事件源感知能力;动态绑定包括上下文事件源到订阅的自动映射和agent与订阅的绑定关系,自动映射将事件与订阅的匹配有效地转化为事件源与订阅的匹配,从而减少匹配次数和提高系统性能;agent对订阅的动态操作为传感器减少了不必要的监听成本。在上下文感知中间件CTK上实现该机制,并通过实验验证了其有效性。

关键词 发布/订阅,位置敏感,上下文事件,动态绑定,CTK

LCESM:Location-aware Context Event Subscription Mechanism

LIU Han[1,2] YE Jian[2] ZHU Zhen-min[2] LIU Ren-ren[1] YU Chang[1,2]
(College of Information Engineering, Xiangtan University, Xiangtan 411105, China)[1]
(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100090, China)[2]

Abstract Traditional location-aware publish/subscribe systems mainly limit at event publication and matching, and couldn't support the frequent context events that happen in the same event source. A location-aware context event subscription mechanism that consists of location-aware context event subscription language(LACL) and dynamic binding approach was presented in the paper. Based on LACL, location event and event source event were defined and made the system capable of location awareness and event source awareness separately; dynamic binding consists of the mapping between the available context event source and the subscriptions and the dynamic binding relationship between agents and subscriptions. The former converts the matching between events and subscriptions to the matching between event sources and subscriptions, and reduces the matching and improves the system performance; agents' dynamic operations to subscriptions avoid redundant monitoring cost. The mechanism was implemented in context-aware middleware CTK. Experiments show effectiveness of the mechanism.

Keywords Publish/Subscribe, Location-aware, Context event, Dynamic binding, CTK

1 引言

当前,移动计算技术和无线网络技术得到了很大发展,许多移动计算设备的便携性、存储能力、计算能力都得到了突破。如数字助理、掌上电脑、智能手机以各种方式、在各种场合灵活地访问和使用着信息和服务资源,给人们的日常工作、生活带来了很大的方便。然而,相对于传统的分布式系统,移动应用有动态性和适应性的要求[1]。移动应用要有上下文感知能力,并且对客户端位置的改变要有自适应能力,以保证为移动客户提供最好的上下文服务。无线传感器网络为移动应用获取各类上下文信息提供了新的传递方式,传感器网络提供上下文事件服务,上下文事件发布/订阅方式具有异步、松散耦合和多对多通信的特点,在分布式环境中得到了广泛应用[2]。

近年来,很多人开始研究将发布/订阅系统扩展到移动环境中,以支持移动环境下的应用。对应用移动性的支持分为对物理移动性和逻辑移动性的支持:在移动环境下,常见的应用就是对移动客户端的物理移动性的支持[3],这种情况下的应用不需要对客户端的移动具有感知能力。即订阅系统的各事件代理都位于固定的有线网络中,而其客户端可以是手机等移动计算设备。事件代理相当于移动客户端对订阅系统的接入点,移动客户端可以从某个事件代理处断开连接,经过一段时间以后在新的事件代理处重新连接到订阅系统。所谓逻辑移动性是指在应用保持与事件代理连接的情况下,在客户端漫游的过程中,应用为客户提供依赖于位置的订阅服务,即位置敏感的订阅。如一辆行驶在街道上的汽车想找一个有免费停车位的停车场,为了能快速到达停车场,汽车应订阅汽车所在区域的停车位。

到稿日期:2010-08-10 返修日期:2011-12-10 本文受国家863重点项目课题(2009AA011906)资助。

刘 韩(1987—),女,硕士生,主要研究方向为普适计算、情境感知;叶 剑(1974—),男,博士,主要研究方向为普适计算、智能感知技术;朱珍民(1962—),男,正研级高工,主要研究方向为普适计算、嵌入式系统;刘任任(1959—),博士,教授,主要研究方向为多值逻辑理论、计算机算法设计与分析。

位置是最重要的上下文信息，在上下文事件订阅过程中发挥着越来越重要的作用。如移动客户端在不同的位置具有不同的订阅上下文服务的需求，同一订阅请求在不同位置对应不同的上下文事件源等。上下文事件的特性要求移动环境中的上下文事件发布/订阅系统具有如下特点：1)位置敏感性。上下文事件接收者和事件产生者能灵活地定义特定区域的事件产生者和接收者。2)多次发布、一次匹配。上下文的动态性使一段时间内对应于同一上下文事件源的上下文事件频繁发生，系统应当避免不必要的事件发布与订阅者的匹配，即在订阅者与事件源的匹配成功，同一事件源事件发布后，不再与订阅者匹配。3)低的监听成本和网络通信量。能灵活取消和启动上下文事件的订阅，减少不必要的传感器监听成本和由上下文事件传送引起的通信量消耗。

本文给出一种位置敏感的上下文事件订阅机制(LCESM)，该机制包括位置敏感订阅语言 LACL 和动态绑定方法。基于 LACL 定义了位置事件和事件源事件，两者分别使系统具有位置感知能力和事件源感知能力；为适应普适计算的动态性，动态绑定方法完成上下文事件源到订阅的自动映射，将事件与订阅的匹配有效地转化为事件源与订阅的匹配，从而减少匹配次数和提高系统性能；此外，绑定 agent 与订阅的动态绑定使系统具有良好的可扩展性和性能，绑定 agent 对订阅的动态操作使系统避免了多余的上下文事件的监听成本和网络通信量，更好地支持了事件源与订阅的动态绑定，为应用提供了满意的位置敏感的上下文信息服务。

本文第 2 节是对国内外相关工作的总结；第 3 节提出了位置敏感订阅语言 LACL，在此基础上定义了位置事件和事件源事件；第 4 节分别分析了事件源与订阅的动态绑定关系和绑定 agent 与订阅的动态绑定关系；第 5 节给出了支持分布式事件源发现的基础拓扑结构和发现算法；第 6 节是应用实例分析和实验结果；最后是对本文主要工作的总结和对未来工作的展望。

2 相关工作

目前，对位置敏感订阅的研究主要集中于事件发送路由协议和位置敏感的事件匹配[4-10]。REBECA[4]中客户端在整个订阅生命周期里是静止的，它只支持静态订阅；文献[5]是对 REBECA 的改进，其以分布式事件代理的方式支持客户端的物理移动性，客户端可以断开与当前事件代理的连接，也可以重新连接到客户感兴趣的新的事件代理；此外，客户端与同一事件代理绑定期间，客户端的应用具有位置感知能力，支持位置敏感的订阅。但是，该系统中事件生产者不能为事件指定特定区域的事件接受者，不能完成订阅方和事件的准确匹配。此外，匹配的双方是订阅者和事件而不是订阅者和事件源，造成了多余的匹配次数。文献[8,9]支持客户端的物理移动性，前者支持客户应用在不同的网络连接节点之间的迁移，以实现客户应用移动的透明性；后者采用预迁移和预恢复的方法支持订阅者移动后的事件迁移。文献[8,9]侧重于支持物理上的客户移动性，而没有考虑逻辑上的位置敏感的订阅服务。文献[6]支持发布/订阅系统中位置敏感的消息传递并给出了“位置导向”路由协议，即发布者指定事件只发送给特定区域的接受者，订阅者只接收发生在特定区域的事件。该系统虽然对分布式事件的发布/订阅提供了很好的支持，但是缺乏对客户端逻辑移动性的支持，即不能随着移动应用位置的改变而提供不同的事件订阅服务。文献[7]在移动客户端上安装位置监听设备，服务器完成位置敏感事件的匹配，即实现了移动应用和事件生产者对位置敏感的事件订阅和发布的双向选择。但是，同一事件源上每次事件发布时都对应一次匹配，在事件发布频繁时，系统性能不高。此外，基于服务的架构存在性能瓶颈，不能有效处理大量订阅事件。文献[10]支持由多个交互移动组件构成的应用的移动性，利用移动组件之间的相关性，减轻了网络通信负担并提高了事件传送效率，同时也从整体上减少了匹配次数，而单个事件与订阅者的匹配次数并没有减少。

上下文的动态性使得上下文事件的发生具有频繁性，同时，在订阅位置敏感的上下文事件时，移动客户端更关注由某个位置的上下文事件源产生的上下文事件。然而，在这些订阅系统中，任何情况下的每次事件发布都引起一次事件匹配，缺乏一种支持事件匹配重用的机制，因此对于在一段时间内订阅同一事件源的移动客户端来说，造成了不必要的事件匹配和系统资源的浪费。同时，在传感器没有订阅者时，不能及时停止监听数据，从而增加了传感器监听成本。因此，采用将上下文事件源与上下文事件分离匹配的方式能减少匹配次数和提高系统性能。采用 agent 技术动态控制整个订阅过程，能有效地减少传感器监听成本。

3 位置敏感的上下文事件订阅语言 LACL

LACL 是一种简单、易操作的类 SQL 语言，描述了上下文事件源的位置约束和订阅者感兴趣的上下文事件。为使系统有效地支持位置敏感的上下文订阅，基于 LACL 定义了位置事件和事件源事件，其中位置事件在某个预定的位置产生，系统通过接收这种位置事件而具有了位置感知的能力；事件源事件在当前事件源不满足订阅需求时发生，系统通过接收这种事件源事件而使订阅过程具有了自适应能力。

定义 1(LACL) LACL 是一种基于数据查询语言 SQL 的扩展语言：

LACL={@}U{&}U{#}U{on}U{constraint-loc}U{func-loc}U{life-control}U{SQL}

其中：

@，& 和 # 分别是上下文变量、传感器属性标识和订阅者属性标识；

on 是约束上下文变量的约束子句；

constraint-loc∈{rang:loc(r),nearest}(r 为实数)，rang:loc(r)和 nearest 分别指定离订阅者距离 r 的范围内和离订阅者位置最近的上下文事件源；

func-loc 是位置选择函数，func-loc∈{anyof(),nearof(),remoteof()}，其中 anyof()表示在位置满足条件 rang:loc(r)的事件源中，任选一个事件源；nearof()表示在位置满足条件 rang:loc(r)的事件源中，选择离订阅者位置最近的一个事件源；remoteof()表示在位置满足条件 rang:loc(r)的事件源中，选择离订阅者位置最远的一个事件源；

constraint-loc 与 func-loc 具有这样的函数关系：f:constraint-loc→func-loc。如果对于任意的 $a\in$ func-loc，$c\in$ constraint-loc，满足 $f(c)=a$，则有

$$a=\begin{cases} b(b\in \text{func-loc}), & c=\text{rang:loc}(r) \\ \phi, & c=\text{nearest} \end{cases}$$

life-control 是管理订阅生命周期的子句，如 life-control：#location in room A，表示订阅者位置在 room A 时，该订阅执行；

SQL 是数据库查询语言标记。

图 1 为一个简单的 LACL 的例子。在此例子中，我们使用了传感器属性标识 & 对传感器属性 sensorid、context 和 location 分别加以约束；on 子句表明了订阅者感兴趣的上下文事件；标识符 # 用来修饰订阅者本身的属性；life-control 子句对该订阅的生命周期进行了约束，表明只有当订阅者处于位置 roomA 时，该订阅才能启动。

```
Select anyof() Where &context=light and &location
in rang:loc(30)
On @>50
Life-control #location in room A
```

图 1　LACL 示例

从图 1 可知，只有属性同时满足条件 a(&context = light)和条件 b(&location in rang:loc(30))的传感器才能作为该订阅的事件源。条件 b 称为事件源位置表达式与订阅者的位置相关，表明订阅事件源可能会因订阅者和事件源本身位置的改变而不同。为了适应订阅者和事件源位置的变化，我们引入了位置事件和事件源事件。此外，life-control 控制着订阅的生命周期，指定了订阅当且仅当订阅者处在某个特定的位置时才启动，位置事件可以有效控制订阅生命周期，减少上下文事件监听成本。

定义 2(事件源事件)　在订阅 p 启动过程中，设 p 的事件源为 o，当 o. location 不满足条件 constraint-loc 时，称 p 的事件源事件发生。图 2 为事件源事件监听接口的实现代码。

```
public class EveSourceListener implements EventListener
{
public void SourceEvent(SourceEvent event) {
 if(event.getSourceState()!=null&&event.getSourceState().equals("false"))
                {
                   notify(subscriber);
                 }
        }
}
```

图 2　事件源事件监听接口

定义 3(位置事件)　位置事件定义为五元组 locationEvent=〈manager，locations，special-loc，state，$F_{\text{loca-state}}$〉。

manager 是唯一管理位置事件注册和监听位置事件发生的 agent，位置事件以〈registerid，实体 id，location〉的形式注册，表示注册者(registerid)需要 manager 监听对象(实体 id)的位置 location。这里，registerid 与 location 有函数关系 $F_{\text{register-loc}}$：registerid→location，如果 $F_{\text{register-loc}}$(registerid)=location，则有：registerid 是订阅 id 时，location=a($a\neq *$)；registerid 是事件源表达式 id 时，location=*。它表示如果是订阅注册，则指定一个特定的位置 location，在实体到达位置 a 时，位置事件发生；否则，用符号 * 表示指定了任意位置，在实体位置变化时，位置事件发生。

令 locations 为$[l_0,\infty)$，是集合 L 的子集，即 locations=$\{l \mid l\in L, l\geqslant l_0\}$，其中 $L=R\times R$，$l_0\in L$，称 locations 为位置集合；special-loc 是当前系统注册的位置集合，special-loc$\subseteq$locations。

state={true，false}；

$F_{\text{loca-state}}$：locations→state，如果对任意 $l\in$ locations 和任意 $s\in$state 满足 $F_{\text{loca-state}}(l)=s$，则有

(((l∈special-loc)&&(s=true)=true))||((l∈$C_{\text{locations}}$special-loc)&&(s=false)=true))=true。当 s 的值由 false 变为 true 的瞬间，位置事件发生。

4　动态绑定

动态绑定可以认为是订阅者的代理，也可以认为是整个环境的代理。主要功能是在订阅者位置变化的情况下，屏蔽位置的变化，自动为订阅者提供可利用事件源的信息，不需要订阅者的干预和改变。订阅者只需向它提出订阅上下文事件请求，而不需要了解下层的映射和位置的变化。

4.1　事件源与订阅的动态绑定

动态绑定的主要功能由接口、需求管理 agent、绑定管理 agent、位置监听 agent、绑定 agent 实现，如图 3 所示。

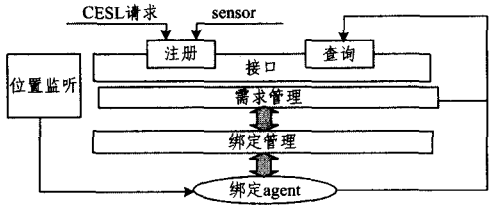

图 3　事件源与订阅的动态绑定

· 接口

订阅者通过注册接口向动态绑定库注册 LACL 请求，传感器通过注册接口注册事件源信息；绑定 agent 利用查询接口获取满足该 LACL 请求的事件源引用。

· 需求管理

需求管理 agent 接收位置敏感的上下文事件订阅，为绑定管理 agent 提供事件源需求信息，向位置监听 agent 注册位置事件，需求管理 agent 处理事件源事件和位置事件，执行订阅的启动或取消操作。主要涉及的数据结构包括一张 Map Table，其中包含了订阅和订阅状态之间的对应关系。每个表项包括订阅 ID、生命周期管理表达式的位置信息和订阅状态。

· 绑定管理

绑定管理 agent 动态、透明地确定(或调整)订阅与所需事件源之间的对应关系。主要涉及的数据结构包括一张 Map Table，其中包含了订阅与事件源信息之间的对应关系。每个表项包括事件源表达式 ID、订阅 ID 和满足该事件源表达式的事件源引用。

· 位置监听

为使系统能够产生位置事件，在位置监听模块中引入了位置事件发生器。位置事件发生器记录着已经注册的位置，即未来的某个位置。当位置到达该位置时，事件发生器就会生成一个相应的位置事件给对应的绑定 agent 或需求管理模块。根据用户订阅中位置的指定类型，系统有两种产生位置事件的方式：①如果位置是相对于订阅的，如在位置 A 启动订阅 x，则在位置事件发生器中注册一个位置事件〈x，订阅者，A〉，并在该事件发生时，通知需求管理 agent，需求管理 agent 查看订阅 x 的状态来决定是否启动订阅 x；②如果位置是相对于事件源的，如订阅 y 的事件源 s 满足事件源表达式 c，则在位置事件发生器中注册两个位置事件：〈c，s，*〉和〈c，

订阅者，*），并在该事件发生时，通知 c 对应的绑定 agent。

· 绑定 agent

为了支持位置敏感订阅，绑定管理 agent 为每个订阅分配一个绑定 agent，绑定 agent 动态查询满足需求的事件源，将查询到的事件源引用通知给绑定管理 agent，绑定管理 agent 动态更新 Map table。绑定 agent 除了查找事件源外，监听事件源表达式值，并在事件源事件发生时通知需求管理 agent。

4.2 绑定 agent 与订阅的动态绑定

不仅订阅与其事件源具有动态绑定关系，我们认为绑定 agent 与订阅之间也有动态绑定关系。绑定 agent 在其生命周期内可以动态地绑定或者释放某个订阅，绑定 agent 所绑定的订阅可以处于运行(active)状态也可以处于非运行(inactive)状态，一个绑定 agent 在某一个时刻可以绑定多个订阅。

在本节中，设 BA 是系统中所有绑定 agent 的集合，P 是系统中所有订阅 p 的集合，S_p 是订阅 p 的所有状态集合，$S_p=\{\text{start},\text{active},\text{inactive},\text{stop}\}$，$H$ 是事件源表达式的集合。$p.h$ 表示订阅 p 的事件源表达式，$p.a$ 表示订阅 p 对应的绑定 agent。设布尔函数 $find(p,a)(p\in P,a\in BA)$ 为真表示绑定 agent a 为订阅 p 找到了符合要求的事件源，否则为假。我们使用 $p_1,p_2,\cdots$ 表示具体的订阅，以 $a_1,a_2,\cdots,a_n$ 表示具体的绑定 agent。

图 4 描述了绑定 agent 与 p 之间的动态绑定关系，其中圆圈表示绑定 agent 所绑定的 p 的状态，有向箭头表示对 p 的操作。绑定 agent 所绑定的 p 具有以下两种不同的状态：

active 状态：绑定 agent 绑定某个 p 期间的一种状态，绑定 agent 可以对该 p 的状态进行改变，监听 p 的事件源表达式 $p.h$ 的值，当 $p.h$ 值变化时通知需求管理 agent；

inactive 状态：绑定 agent 绑定某个 p 期间的一种状态，绑定 agent 为 p 寻找新的事件源，但是绑定 agent 无法改变该 p 的状态，也不监听该 $p.h$ 的值。

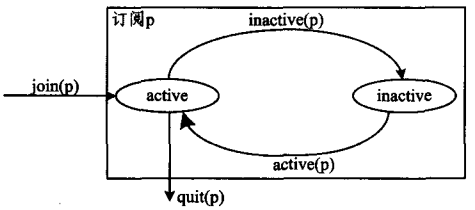

图 4 绑定 agent 与订阅的动态绑定关系

定义 4 令 T 为 $[t_0,\infty)$ 是实数集的子集，即 $T=\{t\mid t\in R,t\geqslant t_0\}$，其中 t_0 可以为任意实数，称 T 为时刻集合。

定义 5 在时刻 $t\in T$，对于任意绑定 agent a，令 $P(a,t)=P^A(a,t)\cup P^I(a,t)$，表示绑定 agent a 在时刻 t 绑定的 p 集合，其中 $P^A(a,t)$ 表示绑定 agent a 在时刻 t 所绑定的并处于 active 状态下的所有 p，$P^I(a,t)$ 表示绑定 agent a 在时刻 t 所绑定的并处于 inactive 状态下的所有 p，则有 $P^A(a,t)\cap P^I(a,t)=\phi$。

定义 6 对于任意的 $p_1,p_2\in P$，如果 $p_1.h=p_2.h$，则有 $p_1.a=p_2.a$。

由定义 5 和定义 6 可知，绑定 agent 与事件源表达式是一一对应关系，但是一个绑定 agent 可以同时绑定多个订阅。

为了支持绑定 agent 动态绑定 p 以及 p 状态的变迁，我们引入了 4 个关于 p 的操作：

1) join：对于任意订阅 $p\in P$，绑定 agent $a\in BA$，则操作 join 对应于函数 $F_{join}:B^A\times S_p\rightarrow S_p$。如果 $F_{join}(a,s)=s'$，则有 $(find(p,a)=\text{true})\&\&(s=\text{start})=\text{true}$ 成立，$s'=\text{active}$。

绑定 agent 为订阅查找到事件源后，通知需求管理 agent，需求管理 agent 使得该 p 处于 active 状态。

2) quit：对于任意订阅 $p\in P$，绑定 agent $a\in BA$，则操作 quit 对应于函数 $F_{quit}:BA\times S_p\rightarrow S_p$。如果 $F_{quit}(a,s)=s'$，则有 $s'=\text{stop}$。

绑定 agent 解除与某个 p 的绑定后，p 停止订阅。

3) active：对于任意订阅 $p\in P$，绑定 agent $a\in BA$，则操作 active 对应于函数 $F_{active}:BA\times S_p\rightarrow S_p$。如果 $F_{active}(a,s)=s'$，则有 $(find(p,a)=\text{true})\&\&(s=inactive)=\text{true}$，$s'=\text{active}$。

绑定 agent 为订阅查找到新的事件源后，通知需求管理 agent 将订阅的状态由 inactive 状态变成 active 状态。

4) inactive：对于任意订阅 $p\in P$，绑定 agent $a\in BA$，则操作 inactive 对应于函数 $F_{inactive}:BA\times S_p\rightarrow S_p$。如果 $F_{inactive}(a,s)=s'$，则有

(constraint-loc(p)=false)&&(s=active)=*true* 成立，$s'=$inactive。

当 p 的事件源事件发生时，绑定 agent 通知需求管理 agent 取消对 p 的订阅。

5 上下文事件源的发现

CTK 系统采用集中型上下文事件源发现模式。集中型的发现范围可以很广，其缺点是健壮性较差，且随着范围和请求的增加效率会降低。分散型模式的优点是简单、健壮性好，易于在计算能力较弱的设备上实现，但其缺点是受范围限制较大，且随着范围和请求的增加效率会降低。普适计算环境需要在较大范围内发现事件源，客户和传感器数目众多，分散型模式显然无法适应；而集中型模式虽然也能工作，但其效率非常低，因此必须寻找新的解决方案。一种有效途径就是在 CTK 发现协议的基础上建立各个注册中心之间的协作机制及相应的基础设施，从而拓展事件源发现的范围。

5.1 拓扑结构

如图 5 所示，结构中的节点分为两类：主干节点 GN 和普通节点 BN。

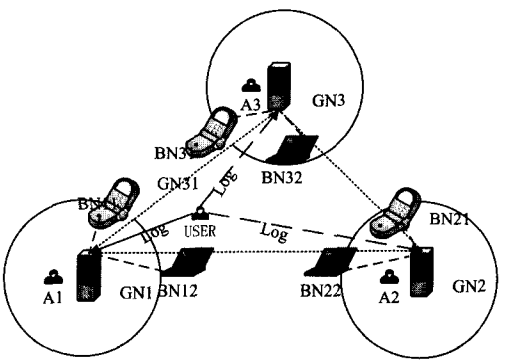

图 5 事件源发现的拓扑结构

主干节点(GN)：所有主干节点之间以一种完全图拓扑形式的分布式 P2P 结构连接，任意节点可以自由加入成为主干节点。主干节点是连接本地和远程组件的通信纽带，同时维护数据传送的路径信息。

普通节点(BN):普通节点包括应用组件和传感器组件,物理位置相近的组件注册在同一 GN 上,GN 存储和维护着注册组件的信息。BN 可以在自身所在位置寻找适合的 GN 注册。

5.2 事件源发现算法

绑定 agent 查找符合订阅需求的上下文事件源的算法描述如下。

Step1 本地事件源查询。订阅者发送订阅消息给本地注册中心 D,绑定 agent 在 D 上查找事件源信息。若查找成功,转 Step4。否则,转 Step5。

Step2 远程事件源查询。绑定 agent 并发查找消息给所有远程注册中心。设其中任意一个为 D_x,D_x 查找事件源信息,若查找成功,回复查询结果(事件源引用)给绑定 agent。否则,转 Step6。

Step3 选择传感组件。绑定 agent 将远程查询到的传感组件信息进行比较,根据事件源选择函数选择符合要求的传感器。如没有符合要求的事件源信息,则等待本地和远程的请求订阅回复消息,转 Step7。

Step4 上下文事件订阅。绑定 agent 发送订阅上下文事件的请求给目标传感器组件完成订阅,转 Step7。

Step5 本地请求订阅。绑定 agent 在 D 上订阅目标事件源信息,直至得到 D 的通知(通知某个传感器符合该事件源信息),同时执行 Step2,转 Step3。

Step6 远程请求订阅。在 D_x 上订阅目标事件源信息。当 D_x 通知绑定 agent 目标传感组件已找到时,取消订阅。该通知消息包括传感组件的连接和位置信息,转 Step3。

Step7 结束。

6 实例分析和实验

6.1 实例分析

为了验证机制对位置敏感订阅的支持,我们在如下会议室场景中实现了一个位置敏感的订阅应用。

如图 6 所示,该会议室分布着多个噪声传感器、3 台 PC(PC1,PC2 和 PC3)和会议参与者随身携带的 PDA。其中噪声传感器 B1、B2 和 B3 注册在 PC1 上,B4 和 A 注册在 PC2 上,B5、B6 和 B7 注册在 PC3 上,会议报告人的订阅请求通过 PDA 注册在 PC1 上;每个 PDA 和噪声传感器上安装有室内定位系统 active bat;会议报告人按照轨迹 2 以 1m/s 的速度行走,并发出订阅请求 A:(a) 当会议报告人在台上时(线 1 的右侧),订阅中央噪音传感器 A,根据噪音量调节麦克风音量;(b) 当报告人在下面(线 1 的左侧)跟会议参与人员一起讨论时,订阅离所处位置最近的噪音传感器 B,根据噪音量调节麦克风音量。

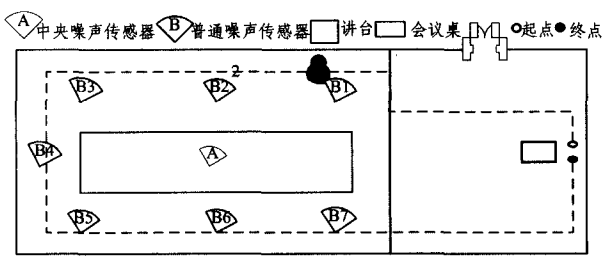

图 6 会议室场景图

订阅的执行与会议报告人的位置有关,在台上和台下分别执行订阅 a 和 b。此外,在执行订阅 b 期间,事件源会随着会议报告人位置的变化而改变。因此,如图 7 所示,LCESM 的运行过程如下:

Step1 需求管理 agent 解析需求,得到标识会议报告人的 id:s、标识订阅 p($p=a$ 或 $p=b$)的 id:$p.id$、事件源的上下文变量约束 $p.c$、订阅的 life-control 约束 $p.l$ 和事件源表达式 $p.d$。

Step2 需求管理 agent 执行操作:1) 在位置产生器中注册〈$p.id$,s,$p.l$.location〉和〈$p.id$,s,null〉,分别注册位置台上(台下)和会议报告人 s 的位置。同时在本地建立一张 Map Table 表,其中包括 $p.id$,$p.state$ 和 $p.l$ 的对应关系,初始时 p.state=start;2) 发送〈$p.id$,$p.d$,$p.c$〉给绑定管理 agent。

Step3 绑定管理 agent 为当前订阅 $p.id$ 分配一个绑定 agent。分配过程如下:绑定管理 agent 在 Map Table 中查找与 $p.d$ 相同的事件源表达式,如果存在事件源表达式 $d'=p.d$,则 $p.a=d.a=d'.a$,绑定 agent($p.a$)向事件源 d.sensor 订阅上下文事件 $p.c$;否则,$p.a=d.a=$new(agent),即产生一个新的 agent。

Step4 $p.a$ 为 p 查找符合事件源表达式 $p.d$ 的事件源 sensor,查找成功后回复〈$p.id$,sensor〉给绑定管理 agent,绑定管理 agent 完成订阅 p 与事件源 sensor 的绑定。同时,绑定管理 agent 发送〈$p.id$,sensor,null〉给位置监听 agent,位置监听 agent 在位置产生器中注册对应的位置事件。

Step5 位置监听 agent 监听 $p.l$.location 时,只有当订阅者位置达到 $p.l$.location 或离开 $p.l$.location 时通知需求管理 agent;在监听订阅者或传感器位置时,当它们的位置变化时通知对应的绑定 agent。绑定 agent 监听 $p.d$ 值。当 $p.d$ 值为 true 时,发送订阅消息给相应传感器完成订阅,通知需求管理 agent 设置 p.state=active。当 $p.d$ 值变为 false 时,$p.a$ 通知需求管理 agent 取消订阅,设置 p.state=inactive。转 Step4。

Step6 需求管理 agent 监听 Map Table 中的 $p.l$ 和订阅启动状态 p.state,在逻辑表达式($p.l$&&p.state)=false 的情况下,当 state=active 时,通知管理 agent 取消订阅 $p.id$;否则,通知需求管理 agent 启动订阅 $p.id$,转 Step3。即当位置监听 agent 监听到会议报告人到达台下时,通知需求管理 agent 停止订阅 a,设置 p.state=stop,启动订阅 b。

Step7 结束。

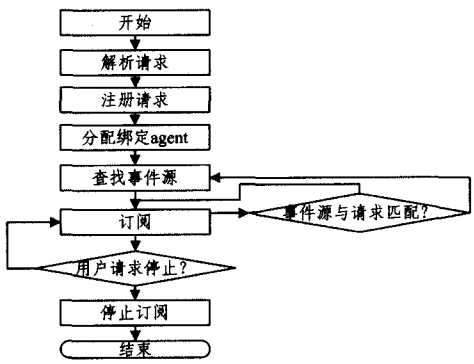

图 7 LCESM 运行流程

6.2 实验

为了验证 LCESM 的有效性,我们在改进的 CTK 平台

(下转第 92 页)

con-nection-Systems management overview[S]

[4] Yang J, Wu J, You Y. A Web-based, Event-driven Network Management Architecture[C]// 4th Asian-Pacific Conference on Communications. 1999:1214

[5] 李克清. 一种动态自适应的网格平衡调度算法[J]. 武汉大学学报,2006(2):69

[6] Gavalas D, Greenwood D, Ghanbarim, et al. Hierarchical network management: a scalable and dynamic mobile agent-based approach [J]. Computer Network, 2002, 38(8):693-711

[7] Zhu Y, Chen T, Liu S. Models and analysis of trade-offs in distributed network management approaches [A] // Integrated Network Man-agement Proceedings [C]. Seattle, WA, USA, 2001:391-404

[8] Yoshihara K, Isomura M, Horiuchi H. Dynamic load balancing for distributed network management [A]// Integrated Network Management[C]. 2003:277-290

[9] Giladi R, Gat M. Meta-management of dynamic distri-buted network managers [A] // IFIP/IEEED SOM [C]. Texas, USA, 2000:119-131

[10] Sinnen O. Task scheduling for parallel systems [M]. Hoboken, New Jersey: John Wiley & Sons, 2007:89

[11] 高隽. 人工神经网络原理及仿真实例[M]. 北京:机械工业出版社,2003:112

(上接第 84 页)

上[11]对 LCESM 与 LPS[7]在传感器监听成本和匹配次数上进行了实验比较。

· 实验环境

会议室均匀分布着 20 个温度传感器,每个温度传感器以 5 次/s 的速度产生数据。10 个移动应用在时间段 $T=60$s 内发出订阅请求:订阅最近的温度传感器。且在 T 内,每个移动应用随机地沿着不同曲线移动,应用每隔 2s 需要更换事件源。

· 实验结果

如图 8 所示,LPS 中的所有温度传感器在时间 T 内不断产生数据。而在 LCESM 下,在时间 T 内有些温度传感器被多个应用共享,平均只有 6~7 个温度传感器产生数据,监听成本比 LPS 低很多。如图 9 所示,在 LCESM 下,只有在当前订阅需求和事件源匹配失效的情况下才进行重新匹配,与 LPS 中一次发布对应一次匹配的情况相比,匹配次数大幅度减少。随着时间的推移,总匹配次数增加缓慢。而 LPS 中总匹配次数随着时间的增加沿呈斜率为 100 的直线增加,增加速度快。

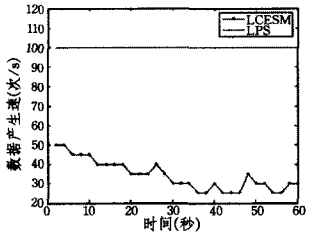

图 8 上下文数据产生速率比较

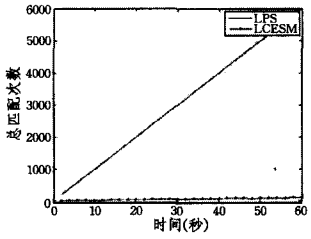

图 9 总匹配次数比较

结束语 本文给出一种位置敏感的上下文事件订阅机制(LCESM),该机制包括位置敏感订阅语言 LACL 和动态绑定方法。基于 LACL 定义了位置事件和事件源事件,两者分别使系统具有位置感知能力和事件源感知能力;为适应普适计算的动态性,动态绑定方法完成上下文事件源到订阅的自动映射,将事件与订阅的匹配有效地转化为事件源与订阅的匹配,从而减少了匹配次数和提高了系统性能。此外,绑定 agent 与订阅的动态绑定使系统具有良好的可扩展性和性能,绑定 agent 对订阅的动态操作使系统避免了多余的上下文事件的监听成本和网络通信量,更好地支持了事件源与订阅的动态绑定,为应用提供了满意的、位置更敏感的上下文信息服务。

下一步需要从以下几方面改进:1) 改进 LACL 语言,使之具有更强的表达能力,以支持更复杂、位置敏感的复合上下文订阅。2) 需要针对不同计算环境的特点,解决如何处理订阅、事件等语言的扩展问题,解决适应不同应用程序需求的服务模块的管理问题,解决不同计算环境下的异构协议问题。

参 考 文 献

[1] Unhelkar B, Murugesan S. The Enterprise Mobile Applications Development Framework[C]// IT Pro. May/June 2010

[2] 马建刚,黄涛,汪锦岭,等. 面向大规模分布式计算发布订阅系统核心技术[J]. Journal of Software, 2006, 17(1):136-147

[3] Huang Yong-qiang, Hector G-M. Publish/Subscribe in a Mobile Environment[J]. Wireless Networks, 2004, 10:643-652

[4] Fiege L, Mühl G, Felix C, et al. Modular event-based systems [J]. The Knowledge Engineering Review, 2002, 17(4):359-388

[5] Fiege L, Gartner F C, Kasten O, et al. Supporting Mobility in Content-based Publish/Subscribe Middleware[C]// Endler M, Schmidt D, eds. LNCS 2672. 2003:103-122

[6] Cugola G, de Cote J E M. On Introducing Location Awareness in Publish-Subscribe Middleware [C] // Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops(ICDCSW'05). 2005

[7] Eugster P T, Garbinato B, Holzer A. Location-based Publish/Subscribe[C]// Proceedings of the 2005 Fourth IEEE International Symposium on Network Computing and Applications (NCA'05). 2005

[8] 闫新庆,尹周平,熊有伦. 支持移动客户应用的分布式订阅/通知框架[J]. 计算机科学,2008,35(6)

[9] 彭禹,苑洪亮,吴泉源. 基于内容的发布订阅中支持订阅者移动的事件迁移算法研究[J]. 计算机研究与发展,2007,44(Suppl.):67-72

[10] Meier R, Cahill V. On Event-based Middleware for Location-aware Mobile Applications[J]. IEEE Transactions on Software Engineering, 2010, 36(3)

[11] Liu Han, Ye Jian, Zhu Zhen-min, et al. Research of Mobile Applications-oriented Peer-to-Peer Context-aware Mechanism[C]// International Conference on Computer Application and System Modeling(ICCASM 2010). Shanxi, Taiyuan, October 2010:121-127