基于发布/订阅中间件的时空事件检测研究

禚 伟[1,2] 金蓓弘[1] 陈海彪[1] 张利峰[1,2]
(中国科学院软件研究所 北京 100190)[1] (中国科学院研究生院 北京 100190)[2]

摘 要 许多物联网应用根据带有时空关系约束的事件决定其下一步的动作。为了支持物联网应用检测这些时空事件,构建了发布/订阅中间件OPS4ST。OPS4ST允许用户在订阅中表达事件之间的多种时序、空间和逻辑关系;同时实现了时空事件的分布式检测,其能高效地检测到用户在订阅中所关心的时空事件是否发生。通过模拟实验评估了系统的性能和开销,实验结果表明,OPS4ST具有令人满意的性能和可接受的开销。

关键词 时空事件,事件检测,发布/订阅,匹配算法
中图法分类号 TP311 **文献标识码** A

Spatio-temporal Event Detection Using Pub/Sub Middleware

ZHUO Wei[1,2] JIN Bei-hong[1] CHEN Hai-biao[1] ZHANG Li-feng[1,2]
(Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)[1]
(Graduate University of Chinese Academy of Sciences, Beijing 100190, China)[2]

Abstract Many applications in the internet of things (IoT) decide their next actions according to the occurrences of events with temporal and spatial constraints. In order to help IoT applications to detect spatio-temporal events, the paper built a Pub/Sub middleware OPS4ST. OPS4ST permits users to express in their subscriptions diverse temporal, spatial and logical relationships of events, implements the detection of spatio-temporal events in a distributed way, and can efficiently detect whether the spatio-temporal events to which the users pay attention in their subscriptions occur. The paper also evaluated OPS4ST's matching performance and costs by simulation experiments. The experimental results show that OPS4ST can achieve satisfying performance and acceptable overheads.

Keywords Spatio-temporal event, Event detection, Pub/Sub, Matching algorithm

1 引言

物联网是基于互联网、传感网、电信网等的面向最终用户提供智能服务的网络。物联网的目标是使得人和物品能在任何时间(Anytime)、任何地点(Anyplace),通过任何网络(Any path/network)和任何服务(Any service),与任何物品(Anything)和任何人(Anyone)进行连接。随着物联网的兴起和普及,用户可以很方便地利用各种设备(如GPS接收器、RFID设备等)观察到周围环境中发生的带有时间和位置标记的事件(称为原子事件)。但仅获得这些事件还不足以满足应用的需求,在许多物联网应用中常常需要捕获时空事件,并根据这些时空事件决定其下一步行为。这里的时空事件是指一个由若干事件(包含原子事件)组成的复杂事件,而组成这些复杂事件的成员事件需满足用户事先指定的时空约束。例如,在物流系统中,经常需要观察下列事件:指定车辆是否离开某指定的仓库,指定车辆是否在指定时间段内到达指定区域,车队是否按指定线路依次到达各送货地等。发布/订阅中间件[1]内在的特性正好适合提供此类服务,可以在上述应用的构建中扮演重要的时空事件检测和通知的角色。

在发布/订阅中间件中,事件反映了物体的状态,通常用属性-值对方式描述,订阅描述了订阅者感兴趣的事件。在基于内容的发布/订阅中间件中,原子订阅是一组谓词的合取,用于指定对事件属性的约束;而复合订阅是将多个订阅(也称成员订阅)用操作符连接起来的订阅,反映了多个事件之间要满足的约束。订阅者提交订阅,声明他们感兴趣的事件,而发布/订阅中间件中的服务器(也称代理)接收原子事件,并负责检测原子事件及复杂事件是否与订阅匹配,然后,将满足订阅的事件推送给订阅者。当前已有的发布/订阅中间件对事件的时间和空间关系的支持都比较弱,这主要体现在发布/订阅中间件提供的复合订阅语言的表达能力上。

复合订阅语言是发布/订阅中间件提供给用户的表达事件模式的语言,它的表达能力决定了发布/订阅中间件的表达能力。当前已有的发布/订阅中间件如Hermes[2]等只提供了检测原子事件的功能,Siena[3]只能表达事件的顺序发生关系,PADRES[4]支持对与、或、顺序发生、重复发生4种事件关系上的复合订阅。RCEDA[5]针对RFID生产流水线的实时监控场景提出了一些复合操作符,包括逻辑与、或、非操作符、时序上顺序操作符以及对时间区间的约束。剑桥大学提出的复合订阅语言[6]可以支持弱顺序发生、强顺序发生、同种事件的多次发生,并允许用户订阅相隔时间不超过固定时间段的

到稿日期:2011-12-13 返修日期:2012-06-13 本文受国家自然科学基金(60970027)资助。
禚 伟(1986—),男,硕士,主要研究方向为分布式计算,E-mail:zhuowei2011@yahoo.cn。

两个事件，但它不支持非触发式事件。所谓的非触发式事件是指无法通过某个事件的发生来通知系统自己已经发生的事件。我们早期提出的复合订阅语言[7]重在挖掘事件之间的时序关系，特别是有严格偏序的时序关系，能支持非触发式事件，但未考虑事件的空间关系。文献[8]提出了同地发生和异地发生的空间关系操作符，并给出了同时发生和顺序发生的时序操作符，这些操作符组合起来共形成了4种时空操作，另外，其还给出了逻辑与和或操作符。

在事件的空间关系表达上，文献[9]为基于位置的服务设计、实现了两个订阅谓词 within 和 distance。在空间事件处理方面，文献[10,11]讨论了空间位置提醒(Spatial Alarm)需求，即用户在他提交的空间警报(类似订阅)中指定他所关心的空间区域，当有移动目标例如用户自己进入该空间区域时，给出报警通知。空间位置提醒提供对一类空间事件的检测，所以，可以将其看成是发布/订阅机制的变种。

本文针对物联网下时空事件检测的需求，设计了新的复合订阅语言，该语言较完整地支持了事件间的各类关系，包括逻辑关系、时序关系、空间关系等的表达。同时，针对该订阅语言，提出了一套时空事件检测策略，包括时序关系判定方法、空间关系判定方法、订阅变量检查机制、事件失效机制等，从而能够正确、高效地完成复杂时空事件的检测。本文第2节介绍 OPS4ST 提供的复合订阅语言；第3节详细描述了时空事件的检测策略；第4节给出了对 OPS4ST 的性能评估；最后是全文总结。

2 复合订阅语言规约

OPS4ST 采用时间区间表示事件发生的持续时间，其中属性 *stm* 表示事件的起始时间，属性 *etm* 表示事件的结束时间。同时，对于事件发生的位置，OPS4ST 采用该事件所覆盖的地理范围边界的坐标点所构成的凸包来表示该事件的位置属性。

为了表述的方便，下文用大写字母(除特别说明的之外)表示订阅，用相应的小写字母表示满足该订阅的事件，并用数字区分同类事件的前后关系，用串联的小写字母表示一个由单个小写字母表示的事件组成的复杂事件。例如，A、B 表示订阅，a、b 表示满足订阅 A、B 的事件，ab 表示由事件 a 和 b 组成的复杂事件，ai 表示满足订阅 A 的第 $i(i>0)$ 个事件。

2.1 复合操作符和订阅变量

OPS4ST 的订阅用复合订阅语言 OPSL 描述。与许多已有的发布/订阅中间件(如 Siena)一样，OPSL 采用基于内容的发布/订阅模式，其中原子订阅是一组谓词的合取。这里的谓词采用如下格式："属性类型:属性名字 操作符 值"，而复合订阅是若干成员订阅(成员订阅可以是原子订阅，也可以是复合订阅)通过复合操作符连结而成的订阅。

OPS4ST 借鉴 Allen 的时序区间代数[12]，在 OPSL 中定义了下列时序操作符：

(1)否定操作符，基本形式为 $!(T;t)$，变体为 $!(A;t)$ 和 $!(A;B)$；

(2)之间(happen-between)操作符，基本形式为 $=|(T;t)$，变体为 $=|(A;t)$ 和 $=|(A;B)$；

(3)之后(happen-after)操作符，基本形式为 $|=(T;t)$，变体为 $|=(A;t)$ 和 $|=(A;B)$；

(4)并发操作符，形式为 $|(A)$。

这里，T 代表一个实际的时间点；t 是一个整数，表示一个时间区间；A、B 表示订阅，上述操作符用来传递发生满足 A 或 B 的事件的时间信息。

基于以上时序操作符，用户可以提交下列格式的订阅：

(1) $!(A;t)B$，该订阅表示如果在 $(a.etm, a.etm+t]$ 之间没有发生事件 b，那么通知提交此订阅的用户，类似格式的订阅可以用于观察非触发式事件；

(2) $=|(A;t)B$，该订阅用于表示用户对事件 b 感兴趣，并且是发生在某个事件 a 发生后的 t 时间段内，即发生在 $(ai.etm, ai.etm+t]$ 内的事件 b；

(3) $|=(A;B)C$，该订阅表示用户感兴趣发生在事件 a 和事件 b 之间的事件 c；

(4) $|(A)B$，该订阅用于观察事件 a 和事件 b 同时出现。

另一方面，通过分析事件的各种空间关系，包括拓扑关系、方向关系和度量关系[13]，在 OPSL 定义了下列空间操作符：

(1)之内(happen-in)操作符，基本形式为 $@(R)$，变体为 $IN(A)$；

(2)覆盖操作符，基本形式为 $OVLP(R)$，变体为 $OVLP(A)$；

(3)相同地点操作符，基本形式为 $SPL(R)$，变体为 $SPL(A)$；

(4)距离操作符，基本形式为 $DIST(R;<;l)$、$DIST(R;=;l)$ 和 $DIST(R;>;l)$，变体为 $DIST(A;<;l)$、$DIST(A;=;l)$ 和 $DIST(A;>;l)$。

这里，R 表示一个用凸多边形表示的区域；l 是整数，表示距离；A 表示订阅，上述操作符用来传递发生满足 A 的事件的位置信息。

基于以上空间操作符，用户可以提交下列格式的订阅：

(1) $@(R)A$，该订阅用于观察发生在区域 R 内的事件 a；

(2) $OVLP(A)B$，该订阅用于表示用户需要在事件 a、事件 b 的发生地点有重叠的时候被通知；

(3) $SPL(A)B$，该订阅表示用户关注在相同地点发生的事件 a 和事件 b；

(4) $DIST(R;<;l)A$、$DIST(R;=;l)A$、$DIST(R;>;l)A$，这些订阅分别用于观察事件发生地与区域 R 距离小于 l 的事件 a、与区域 R 距离为 l 处的事件 a 和与区域 R 距离大于 l 的事件 a。

OPSL 还定义了两个逻辑操作符 &&、||，它们分别表示事件之间的逻辑与和逻辑或关系。

为了在一个复合订阅中表达成员订阅之间的相关关系，以反映成员事件之间的空间和时间关系，引入了订阅变量的概念。订阅变量是一个以"$"开头、以";"结尾的字符数字串，用于指代一个订阅。在复合订阅中，一个订阅变量能出现在任何允许订阅出现的位置。若在订阅 A 中出现了两个订阅变量"$var1;"和"$var2;"，那么，它们的使用需要遵循下面的格式：

$A, \$var1;=(sub1), \$var2;=(sub2)$

式中，"$var1; =(sub1)"和"$var2;=(sub2)"表示订阅变量"$var1;"、"$var2;"分别绑定到订阅 *sub*1 和 *sub*2，同时，这两个订阅变量可以出现在 A 中可以出现订阅的地方。

2.2 订阅举例

下面通过物流应用场景来说明订阅语言的使用。作为物流管理业务之一，物流公司的人员需要监测载货车辆是否离开源仓库、是否到达目的地仓库。假设源仓库位于区域 *Region1*，目的地仓库位于区域 *Region2*，订阅 A 关注指定车辆的出现事件，那么，下面的订阅(1)可以用于观察原来位于区域 *Region1* 的车辆、在 $t1$ 时段后与该区域 *Region1* 的距离大于 $x1$ 的事件。如果检测到满足该订阅的事件，那么意味着该车辆已离开源仓库。订阅(2)用于观察原来与区域 *Region2* 的距离大于 $x2$ 的车辆在 $t2$ 时间内到达 *Region2*，也就是说，订阅(2)可以观察到车辆到达目的地仓库。

$$=|((@(Region1)(A);t1)(\mathrm{DIST}(Region1;>;x1)(A)) \quad (1)$$

$$=|(\mathrm{DIST}(Region2;>;x2)(A);t2)(@(Region2)(A)) \quad (2)$$

在货物运输过程中，通常带队车辆与其他车辆 V 的距离不能超过 x，如果距离超过 x，就意味着车辆 V 远离了车队，物流公司的相关人员应该收到相应的通知。所以，物流公司的相关人员可以发送如下订阅：

$$=|(\$e1;;t)(\$e2;),$$

$$\$e1;=(|(\$A;)(\$B;)\ \&\&\ \mathrm{DIST}(\$A;;<;x)(\$B;)),$$

$$\$e2;=(|(\$C;)(\$D;)\ \&\&\ \mathrm{DIST}(\$C;;>;x)(\$D;)),$$

$$\$A;=E\text{-}A,\ \$B;=E\text{-}B,\ \$C;=E\text{-}A,\ \$D;=E-B \quad (3)$$

这里，订阅 *E-A* 表示关注带队车辆之外的其它车辆出现的事件，订阅 *E-B* 表示关注带队车辆出现的事件。订阅(3)用于观察在 t 时段内带队车辆和其他某个车辆之间的距离从小于 x 到大于 x，这意味着，那辆车脱离了车队。

3 时空事件检测

时空事件检测的基本流程是：当订阅者提交的复合订阅从客户端到达 OPS4ST 的代理 X 时，会在代理 X 上建立相应的匹配结构，同时以代理 X 为订阅者，向其他代理提交组成该复合订阅的各原子订阅。当原子事件到达某代理后，若与该代理上所保存的以 X 为订阅者的订阅相匹配，则将该事件转发给 X，由 X 负责进一步对复杂事件的匹配。本节首先介绍匹配中用到的数据结构和匹配流程，然后描述复杂事件匹配中的 4 项关键技术。

3.1 匹配结构和匹配流程

OPS4ST 利用一个有向无环图作为匹配结构记录一个复合订阅包含的信息。在该匹配结构中有 3 类节点：(1)订阅节点，用于记录原子订阅；(2)操作符节点，记录操作符约束；(3)时间节点，记录时间点。

操作符节点需要维护以下信息：(1)订阅变量的索引。若订阅不含订阅变量，则该索引为空。具体而言，索引记录了(a)在该节点定义的或者由其他节点传来的订阅变量，(b)若该变量是由其他节点传来的，需要记录该前驱节点；(2)一个或多个保存已到达事件的队列；(3)指向后继节点的指针；(4)指向前驱节点的指针；(5)订阅者列表。

图 1 给出了订阅"$|(\$x;)\$y;\ \&\&\ \mathrm{OVLP}(\$x;)\$y;,\ \$x;=A\&\&B,\ \$y;=D\&\&E$"的匹配结构，同时在节点旁标注了订阅变量，其中 L、R 表示订阅变量来自前驱节点中的左节点和右节点。图中标有 t 的节点为时间节点，出度为 0 的"&&"操作符节点是该订阅的根节点。

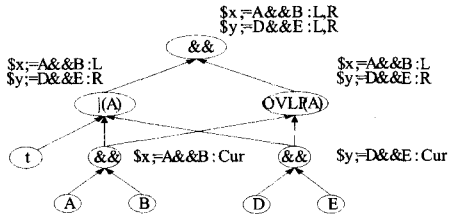


图 1 订阅"$|(\$x;)\$y;\ \&\&\ \mathrm{OVLP}(\$x;)\$y;,\ \$x;=A\&\&B,\ \$y;=D\&\&E$"的匹配结构示意图

时空事件的匹配流程是：当一个普通事件到达匹配结构中的某个订阅节点时，表明它与该节点所代表的原子订阅相匹配，所以，将该事件从订阅节点传递到所有的后继节点，后继节点会根据自己所代表的复合操作符执行相应的约束(包括时序关系约束、空间关系约束、订阅变量一致性约束)检测。若该事件使得后继节点上的操作符约束也被满足，那么会形成一个新的复杂事件，这个新的复杂事件将继续向后继节点传递并重复进行前述的处理过程，直到到达出度为 0 的节点。若该节点的订阅者列表非空，则对应的复杂事件就会发送给订阅者。

如果到达的不是普通事件而是时间事件，那么该时间事件会触发相应的时间节点，后面的处理流程同上。

3.2 时序关系的判定

为了能够支持时序关系的判定，OPS4ST 引入了时间事件和时间事件发生器。时间事件是指系统内部生成的那些发生在预定时间点的事件。为叙述方便，以下把除时间事件之外的事件称为普通事件。通常，时间事件会由时间节点向时间事件发生器提前注册，该步骤称之为"注册时间事件"，当到达时间事件注册的时刻时，时间事件产生器会产生相应的时间事件并发送给对应的时间节点。

注册时间事件的时刻因订阅中所含的复合操作符而异，可以分为两大类：(1)时间事件是在建立匹配结构的操作符节点时注册。以订阅$=|(T;t)A$ 为例，由于只有匹配订阅 A 并且发生在$(T,T+t]$时间段的事件才会满足该订阅，因此，在构造该订阅匹配结构时会注册"T"与"$T+t$"这两个时间事件；(2)时间事件是在时间节点的后继节点收到相关的普通事件时注册。比如，在订阅$!(A;t)B$的匹配结构中，$!(A;t)$操作符节点在收到满足订阅 A 的事件 a 时，会由时间节点 t 向时间事件发生器中注册时间事件"$a.etm+t$"。

匹配结构中的时间节点能够接收时间事件并触发后继节点开始或者不再处理普通事件，还能使事件队列中的事件生效或者失效。比如，$|=(T;t)$操作符节点在收到时间事件"$T+t$"时，开始对到达的普通事件进行处理；但是，$=|(T;t)$操作符节点在收到该时间事件后就不再处理到达的普通事件；又如，$|=(A;t)$操作符节点在收到 a 事件对应的时间事件"$a.etm+t$"后使 a 事件生效，但当$=|(A;t)$操作符节点收到时间事件"$a.etm+t$"后，将使 a 事件失效。

3.3 空间关系的判定

在 OPS4ST 系统中，空间关系判断涉及以下几种计算。(1)求点集的凸包时采用 Graham 扫描算法[14]求解。在两个

或多个事件形成一个复杂事件时，将根据构成各成员事件的点集构造出的最小凸多边形作为该复杂事件的位置属性。(2)判断点与凸多边形的拓扑关系。通过叉积计算二维空间上一个点和凸多边形的关系，即点在凸多边形内、点在凸多边形上、点在凸多边形外。(3)判断凸多边形之间的拓扑关系，即两个凸多边形的“内含(in)”、“重叠(overlap)”、“分离(disjoint)”时，通过逐个计算点与凸多边形关系来得出凸多边形间的拓扑关系，从而获得事件之间的拓扑关系。(4)求解两分离的凸多边形的距离即最近点对的距离，并把它作为两个事件之间及事件与区域之间的距离。这里采用旋转卡壳寻找对踵点算法[15]计算。(5)计算凸多边形的中心点。将包含 n 个顶点的凸多边形分解为 $n-2$ 个三角形(它们以凸多边形的某个顶点为其公共顶点)，分别求这 $n-2$ 个三角形的重心和面积，得到总面积的同时按照三角形的面积加权相加得到凸多边形的重心，并将该重心作为凸多边形的中心点。通过求两凸多边形中心点之间的距离是否小于设定阈值来判断两事件是否发生于同一个地点。

3.4 订阅变量检查机制

如果匹配结构中的一个操作符节点含有订阅变量(假设为“$x;”)的定义，那么满足该节点的操作符约束的事件 *Event* 将会以“*Event*,⟨$x;,*Event*⟩”的形式传递给它的后继节点。后继节点将根据订阅变量的到达情况进行订阅变量取值的一致性检查。接下来，以订阅“|($x;)$y; &&OVLP($x;)$y;,$x;=A&&B,$y;=D&&E”的处理过程为例来分析说明订阅变量检查流程。

假设在某一时刻分别满足订阅 A、B、D、E 的 a、b、d、e 四个原子事件同时到达 OPS4ST 的某一代理。那么，首先如图 2(a)所示，会形成两个复杂事件 ab 与 de。接着，如图 2(b)所示，由于表示 $A\&\&B$ 的操作符节点是订阅变量“$x;”的定义节点，那么其向后继节点传递事件 ab 的同时还会附加订阅变量信息⟨$x;,$ab$⟩；类似地，复杂事件 de 也会附加⟨$y;,de⟩并传递给其后继节点。然后，如图 2(c)所示，由于在|(A)与 OVLP(A)两个操作符节点中，订阅变量“$x;”和“$y;”都只出现了一次，因此无需进行订阅变量的检测。假设 ab 和 de 都满足这两个节点上的时间、空间约束，那么这两个事件组成新的复杂事件即“$abde$,⟨$x;,$ab$⟩,⟨$y;,de⟩”继续向后继节点传递。最后，如图 2(d)所示，当前述复杂事件到达订阅“|($x;)$y; && OVLP($x;)$y;”的操作符节点后，由于该节点上每个订阅变量都出现了两次，因此需要进行订阅变量的一致性检测。就上述订阅而言，可以判断出从左右前驱节点到达的复杂事件 ab 是一致的，因此订阅变量“$x;”的取值是一致的，同理可得订阅变量“$y;”的取值也是一致的。因此可以组装成新的复杂事件“$abde$,⟨$x;,$ab$⟩,⟨$y;,de⟩”发送给订阅者。

看另一个例子，若所发生的事件有下列特征：(1)4 个事件 $a1$、$b1$、$d1$、$e1$ 同时到达，随后事件 $d2$、$e2$ 到达；(2)$a1$、$b1$ 形成的复杂事件 $a1b1$ 与 $d1$、$e1$ 形成的复杂事件 $d1e1$ 不能满足操作符 OVLP(A)中的空间约束条件，而与 $d2$、$e2$ 形成的复杂事件 $d2e2$ 能满足操作符 OVLP(A)中的空间约束。那么，根据订阅变量检查机制，订阅“|($x;)$y; && OVLP($x;)$y;”对应的操作符节点得到的事件分别为 $a1b1d1e1$ 组成的复杂事件(来自左分支)和 $a1b1d2e2$ 组成的复杂事件(来自右分支)，两个复杂事件中的订阅变量“$y;”其赋值不一致，从而表示订阅“|($x;)$y; && OVLP($x;)$y;”的操作符节点不会被触发。

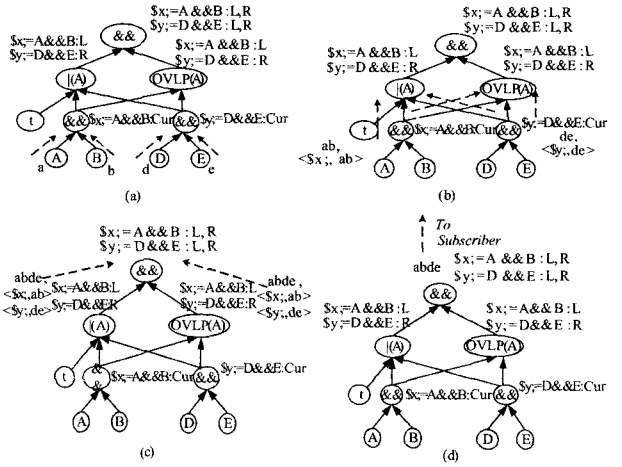


图 2 订阅“|($x;)$y; && OVLP($x;)$y;,$x;= A&&B,$y;= D&&E”的处理过程

3.5 事件失效机制

引入订阅变量机制能够增强复合订阅语言的表达能力，但是订阅变量的处理也会给事件检测带来副作用。如图 3 所示，在订阅“|($x;)$y; && OVLP($x;)$y;,$x;=A&&B,$y;= D&&E”的匹配过程中，在 $t1$ 时刻到来事件 $a1$、$b1$、$t2$ 时刻到来事件 $d1$、$e1$，它们分别满足订阅 A、B、D、E。这时与订阅节点相连的操作符节点会组装复杂事件“$a1b1$,⟨$x;,$a1b1$⟩”与“$d1e1$,⟨$y;,$d1e1$⟩”沿出边传递。由于“|(A)”操作符节点上的时序约束未被满足，那么，在该节点的左事件队列添加事件“$a1b1$,⟨$x;,$a1b1$⟩”，右事件队列添加事件“$d1e1$,⟨$y;,$d1e1$⟩”；假设“OVLP(A)”操作符节点上的空间约束未被满足，那么，在该节点的左事件队列添加事件“$a1b1$,⟨$x;,$a1b1$⟩”，右事件队列添加事件“$d1e1$,⟨$y;,$d1e1$⟩”。在 $t3$ 时刻前，“|(A)”操作符节点注册表示并发约束超时的时间事件到达，使得事件失效，则该节点的左、右事件队列为空。在 $t3$ 时刻，到来了事件 $a2$、$b2$、$d2$、$e2$，它们分别满足订阅 A、B、D、E，并假设它们能够满足整个订阅。在具体匹配中，在“&&”操作符节点上会组装复杂事件“$a2b2$,⟨$x;,$a2b2$⟩”与“$d2e2$,⟨$y;,$d2e2$⟩”并传递给后继节点。它们满足“|(A)”操作符节点上的时序约束后形成复杂事件“$a2b2d2e2$,⟨$x;,$a2b2$⟩,⟨$y;,$d2e2$⟩”继续传递；而在“OVLP(A)”操作符节点上，假设“$a2b2$,⟨$x;,$a2b2$⟩”与右事件队列的事件“$d1e1$,⟨$y;,$d1e1$⟩”满足“OVLP(A)”节点的空间约束，而“$d2e2$,⟨$y;,$d2e2$⟩”与左事件队列的事件“$a1b1$,⟨$x;,$a1b1$⟩”满足“OVLP(A)”节点的空间约束，那么“OVLP(A)”节点将传递复杂事件“$a1b1d2e2$,⟨$x;,$a1b1$⟩,⟨$y;,$d2e2$⟩”与“$a2b2d1e1$,⟨$x;,$a2b2$⟩,⟨$y;,$d1e1$⟩”。由于这些事件未能满足根节点的订阅变量约束，因此这个订阅将无法被触发。订阅未能形成复杂事件“$a2b2d2e2$”而通知给订阅者。

形成订阅变量副作用的原因是空间操作符节点没有时间感知能力，保存在时序操作符上的事件会随着时间事件到达而失效，而空间操作符上保存的对应的事件未能够被正确地清除，从而导致在各个操作符中记录的事件出现不一致的情况，造成无法正确地进行事件匹配。为此，引入了事件失效机制，当与订阅变量定义匹配的事件在某个操作符处理中失效时，能够保证事件在系统中以一致的方式被记录。

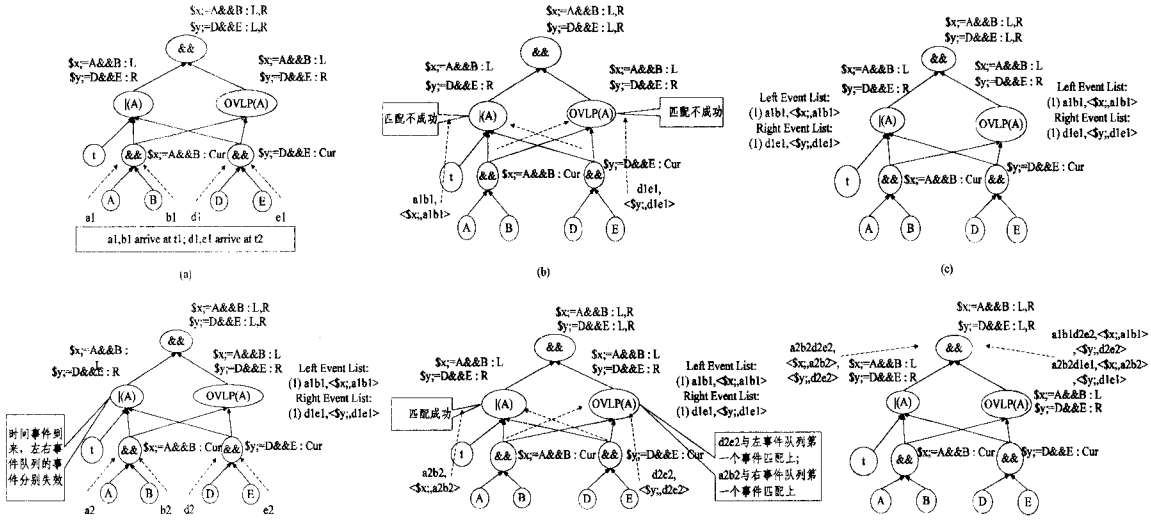


图 3　订阅变量机制的副作用的例子

下面分析事件失效机制的触发条件。当一个事件到达订阅匹配结构进行匹配时会发生这样 4 个行为：(1)与已经到达的事件形成新的复杂事件；(2)未能匹配成功而主动失效；(3)对应的时间事件到达而被清理；(4)使得某节点或者某节点上已有的一些事件永远不能再形成新的复杂事件，自己也完成使命而失效。当一个事件出现后 3 种行为时，事件即在系统中消失，应该触发事件失效机制。而事件失效机制的内涵总结为以下 3 个步骤：

(1)判断事件是否是本节点上与某个订阅变量定义匹配的事件，如果是，则转(2)。如果不是，则停止处理过程。

(2)通过节点记录的后继节点信息，找到订阅的根节点。

(3)从根节点沿入边逆向遍历其他节点，删除这些节点上的事件队列中的相应事件。

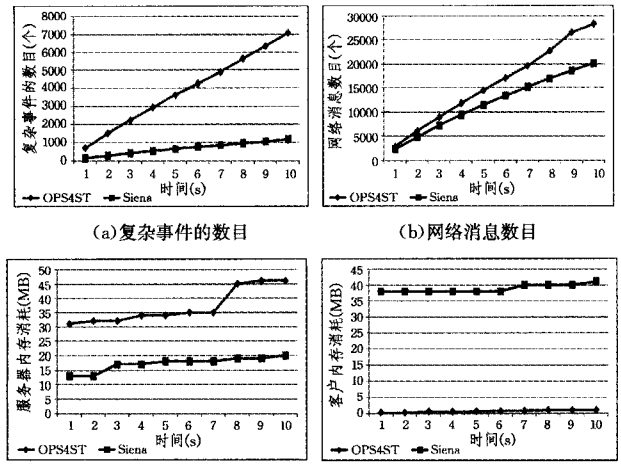
4　实验评估

为了评估 OPS4ST 的性能和开销，以物流应用场景为背景设计实验，比较了 OPS4ST 与 SIENA 在匹配时空事件方面的性能和开销。考虑到公平性，为 SIENA 编写了胖客户端，使得该系统能完成与 OPS4ST 一样的复合订阅匹配的功能。

实验中将第 2.2 节中的订阅(1)、订阅(2)和订阅(3)作为输入系统的 3 类订阅。上述 3 个订阅中的订阅 *A*、*E-A*、*E-B* 都具有如下格式：*ARITH*：*id*＝*v*1，*STRING*：*type*＝*v*2，其中，*id* 表示车辆的标识符，*type* 表示车辆的类型，比如卡车、小货车等，*v*1 是正整数，*v*2 是字符常量。同时，实验中为每一类订阅各生成 5000 个具体的复合订阅。另外，事件按如下格式生成：*ARITH*：*id*＝*v*1，*STRING*：*type*＝*v*2，*timestamp*，*x*，*y*，其中，*v*1、*v*2 的含义及取值范围与上述订阅相同，*timestamp* 表示事件发生时间，(*x*，*y*)表示事件发生的位置。

我们针对 3 类事件检测进行了 3 组实验。在每组实验中，首先注入属于同一类的 5000 个复合订阅，然后以 2000 个/秒的速率发布事件，总共发布的事件量为 20000 个。针对订阅(1)、(2)，保证事件流中两个连续的事件能匹配上一个已存在的订阅；针对订阅(3)，保证事件流中 4 个连续的事件能匹配上一个已存在的订阅。

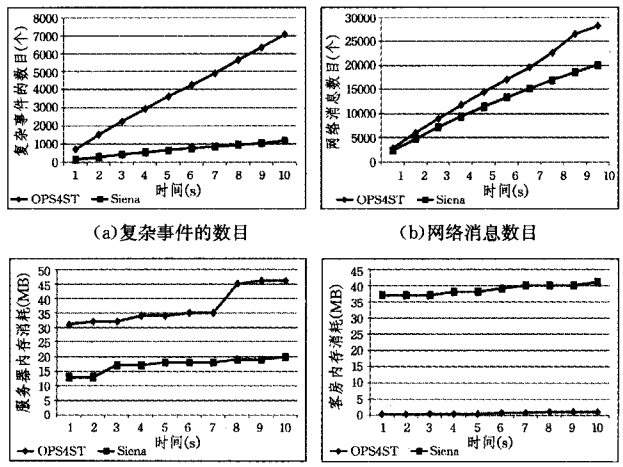
在 3 组实验中，对每组实验，分别记录随时间的流逝匹配成功的复杂事件的总数目、所消耗的网络消息数目、服务器端内存消耗以及客户内存消耗。实验数据如图 4—图 6 所示。



(a)复杂事件的数目　(b)网络消息数目

(c)服务器端内存消耗　(d)客户内存消耗

图 4　检测车辆离开仓库事件



(a)复杂事件的数目　(b)网络消息数目

(c)服务器内存消耗　(d)客户内存消耗

图 5　检测车辆到达仓库事件

从图 4—图 6 可以看出，每一组实验的实验结果类似，所以这里只给出对“检测车辆离开仓库”的实验结果分析。从图 4 所示的实验结果可以看出，在实验过程中，OPS4ST 在 10 秒时间内总共匹配出了 7068 个复杂事件，Siena 改造版总共匹

(下转第 135 页)

[4] Bartolini C, Bertolino A, Marchetti E, et al. WS-TAXI: A WSDL-based Testing Tool for Web Services[C]// Proceedings of the International Conference on Software Testing Verification and Validation (ICST '09). Denver, Colorado, USA, 2009: 326-335

[5] Li Z J, Zhu J, Zhang L-J, et al. Towards a practical and effective method for Web Services test case generation[C]// Proceedings of the ICSE Workshop on Automation of Software Test (AST '09). May 2009: 106-114

[6] Wang Y, Bai X, Li J, et al. Ontology-based Test Case Generation for Testing Web Services[C]// Proceedings of the Eighth International Symposium on Autonomous Decentralized Systems (ISADS '07). Sedona, AZ, USA, Mar. 2007: 43-50

[7] Yu Y, Huang N, Luo Q. OWL-S Based Interaction Testing of Web Service-based System[C]// 3rd International Conference on Next Generation Web Services Practices (NWeSP 2007). Seoul, South Korea, Oct. 2007: 31-34

[8] Li Bi-xin, Ji Shun-hui, Qiu Dong, et al. Generating Test Cases of Composite Services Based on OWL-S and EH-CPN[J]. International Journal of Software Engineering and Knowledge Engineering, 2010, 20(7): 921-941

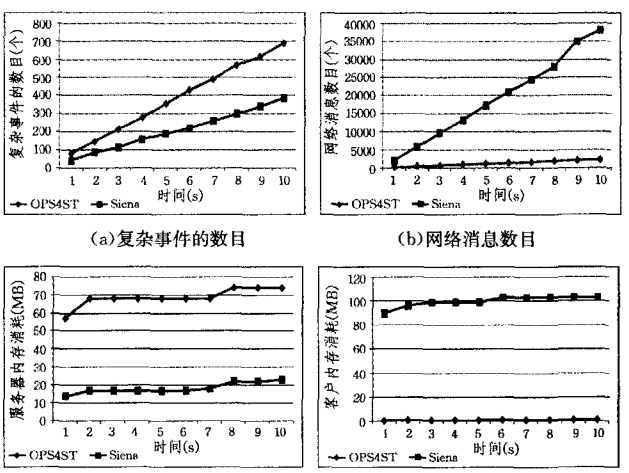
[9] Martin D, Burstein M, Hobbs J. OWL-S: Semantic Markup for Web Services[OL]. http://www.w3.org/Submission/OWL-S/, 2004-11-22

[10] Bertolino A, Gao J, Marchetti E, et al. Automatic Test Data Generation for XML Schema-based Partition Testing[C]// AST '07: Proceedings of the 2nd International Workshop on Automation of Software Test. Minneapolis, Minnesota, USA, May 2007: 11-17

[11] Maryland Information and Network Dynamics Lab Semantic Web Agents Project[OL]. http://www.mindswap.org/2004/owl-s/1.1/FindCheaperBook.owl, 2010-10-12

(上接第 103 页)

配出了 1180 个复杂事件，OPS4ST 匹配能力为 Siena 改造版的 6.0 倍；10 秒内，OPS4ST 所消耗的网络消息总数为 28272 个，Siena 改造版为 20056 个，这意味着 OPS4ST 每匹配出一个复杂事件需要网络消息传送数目约为 Siena 改造版的 1/4。另外，OPS4ST 服务器内存消耗约为 Siena 改造版的 2.5 倍，但是客户端内存消耗远远小于 Siena 改造版。



(a)复杂事件的数目　(b)网络消息数目

(c)服务器内存消耗　(d)客户内存消耗

图 6　检测车辆离开车队事件

通过以上实验可以看出，OPS4ST 的匹配性能高于 Siena 改造版。胖客户端策略是在客户端完成时空检测的计算，所以除了吞吐量低于 OPS4ST 外，还会导致系统中事件传输数量的增加，从而增加了网络传输的开销；而且如果同一个订阅被不同的客户订阅，那么，该订阅会在不同的客户同样地匹配计算，这增加了系统总体计算资源的开销。

结束语　本文关注利用发布/订阅系统进行时空事件检测，为检测时空事件提供了一个简洁而功能强大的复合订阅语言，该语言提供 4 类 10 种时序操作符、4 类 11 种空间操作符，利用它们可以描述物联网中大多数时空事件。针对提出的复合订阅语言，本文提出了一套时空事件检测策略，从而能够正确、高效地完成复杂时空事件的检测。上述机制已在我们开发的发布订阅中间件 OPS4ST 中实现。实验结果表明，相对于 SIENA 改造版，OPS4ST 的匹配效率更高、开销更低。

参 考 文 献

[1] Eugster P T, Felber P A, Guerraoui R, et al. The Many Faces of Publish/Subscribe[J]. ACM Computing Surveys, 2003, 35(2)

[2] Pietzuch P R, Bacon J. Hermes: A distributed event-based middleware architecture[C]// Proceedings of the 22nd International Conference on Distributed Computing Systems. Washington, IEEE Computer Society Press, 2002: 611-618

[3] Carzaniga A, Rosenblum D, Wolfal. Design and Evaluation of a Wide-Area Event Notification Service[J]. ACM Transaction on Computer Systems, 2001: 19(3): 332-383

[4] Li G L, Jacobsen H A. Composite Subscriptions in Content-based Publish/Subscribe Systems[C]// The 6th ACM/IFIP/USENIX International Middleware Conference. 2005: 249-269

[5] Wang Fus-heng, Liu Shao-rong, Liu Pei-ya, et al. Bridging Physical and Virtual Worlds[C]// Complex Event Processing for RFID Data Streams. EDBT, 2006: 588-607

[6] Pietzuch P R, Shand B, Bacon J. A Framework for Event Composition in Distributed Systems[C]// The 4th ACM/IFIP/USENIX International Conference on Middleware. 2003: 62-82

[7] Jin B, Zhao X, Long Z, et al. Effective and Efficient Event Dissemination for RFID Applications[J]. Computer Journal, 2009, 52(8): 988-1005

[8] Schwiderski-Grosche S, Moody K. The SpaTeC Composite Event Language for Spatio-Temporal Reasoning in Mobile System [C]// DEBS. 2009

[9] Chen X, Chen Y, Rao F. An Efficient Spatial Publish/Subscribe System for Intelligent Location-Based Services[C]// DEBS. 2003

[10] Bamba B, Liu L, Yu P S, et al. Scalable Processing of Spatial Alarms[C]// Annual IEEE International Conference on High Performance Computing. 2008

[11] Bamba B, Liu L, Iyengar A, et al. Distributed Processing of Spatial Alarms: A Safe Region-based Approach[C]// International Conference on Distributed Computing Systems-ICDCS. 2009

[12] Allen J F. Maintaining Knowledge about Temporal Intervals [J]. Communications of the ACM, 1983, 26(11)

[13] Guting R H. An Introduction to Spatial Database Systems[J]. The VLDB Journal, 1994, 3(4)

[14] Cormen T H, Leiserson C E, Rivest R L, et al. Introduction to Algorithms (Second Edition)[M]. MIT press, 2001

[15] Toussaint G. Solving Geometric Problems with the Rotating Calipers[C]// IEEE Melecon. 1983: 10