基于乱序修正框架的物联网复杂事件检测模型

许冬冬 袁凌云 李 晶
(云南师范大学信息学院 昆明 650500)

摘 要 针对物联网应用系统中存在的时间戳乱序问题,给出了物联网语义事件定义,对时间戳乱序问题进行了描述,同时基于混合驱动的空间回收机制,构建了基于哈希结构的复杂事件乱序修正框架,并提出了一种基于乱序修正框架的复杂事件检测算法(ORFCED)。该算法提取事件的2个特征参数来计算哈希地址,利用时间戳特性将事件存入循环单链表进行局部排序,从而解决了时间戳乱序问题。仿真结果表明,所提出的ORFCED算法不仅具有较高的处理正确率和可靠性,而且可以对乱序流及时地作出反应,弥补了现有方法存在的不足。最后通过案例研究验证了所提算法的有效性和可行性。

关键词 物联网,复杂事件检测,乱序事件流,乱序修正,空间回收机制

中图法分类号 TN919.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.10.027

IoT Complex Event Detection Model Based on Out-of-order Revise Framework

XU Dong-dong YUAN Ling-yun LI Jing
(College of Information Science and Technology, Yunnan Normal University, Kunming 650500, China)

Abstract There are always events with out-of-order timestamps in the Internet of Things (IoT) application systems. To deal with the problem, a semantic event definition about IoT was presented and the issue of out-of-order timestamps was also described. Meanwhile, according to the mixed driving space reclaim mechanism, an out-of-order revise framework of complex events based on Hash structure was established. What's more, a complex event detection algorithm based on out-of-order revise framework (ORFCED) was proposed. To solve the issue of out-of-order timestamps, the algorithm extracts two characteristic parameters of events to compute the Hash address and stores events into circular linked list in the timestamp order to sort them locally. Simulation results show that the proposed ORFCED algorithm not only can process events with high accuracy and reliability, but also can respond timely to out-of-order streams, which makes up for the deficiency of the existing methods. Finally, a case study was made, which verifies the effectiveness and feasibility of the proposed algorithm.

Keywords IoT, Complex event detection, Out-of-order event streams, Out-of-order revising, Space reclaims mechanism

1 引言

物联网作为新一代网络技术,被誉为是继计算机、互联网之后的第三次信息浪潮,受到各界的广泛推崇,并逐渐被应用于医疗健康、供应链管理、智能交通、物流运输等领域[1,2]。物联网数据具有海量性和异构性等特点,如何从这些海量数据中快速找到所需的数据[3],挖掘出数据之间隐含的内在关系,提取并抽象出有意义的事件,是目前迫切需要研究的一个问题。

复杂事件检测方法的出现为大规模物联网数据流处理带来了极大的便利,它可以大大减少物联网数据处理的时间[4]。作为物联网关键技术之一的复杂事件处理技术,已经被广泛应用于各种领域,国内外的研究者也已经在这些领域取得了一些有价值的成果。数据流中的数据多为简单事件,但单个简单事件对于具有关联性的海量数据来说没有实际意义,且处理单个事件的开销巨大[5]。因此,事件流的有效化是利用事件流的必要工作[5,6]。复杂事件检测是一种常见的事件流有效化技术,即按照给定的规则和模式,将多个基本事件复合成为有着更加复杂语义的复杂事件[6]。

在大多数情境下,物联网数据在传输过程中,要求来自不同设备的事件以事件流的方式聚集,但这些事件常常会由于硬件故障、网络拥塞、存储延迟等原因导致不同数据源产生的事件不能按照时间戳顺序到达处理器,即产生了时间戳乱序现象。如果不对这些乱序事件流进行及时有效的处理,那么这些乱序事件会导致错误的查询结果并影响相关数据的处理。

目前国际上针对物联网数据流时间戳乱序的复杂事件检

到稿日期:2014-10-28 返修日期:2015-02-02 本文受国家自然科学基金资助项目(61262071),教育部人文社会科学研究青年基金资助项目(13YJCZH233),云南省教育厅面上项目(2012Y286),云南省应用基础研究计划面上项目(2013),云南师范大学研究生科研创新基金资助项目资助。

许冬冬(1989—),男,硕士生,主要研究方向为语义物联网技术、复杂事件处理技术,E-mail:xudongdong.ynnuedu@163.com;袁凌云(1980—),女,博士,副教授,硕士生导师,CCF会员,主要研究方向为无线传感器网络、物联网技术;李 晶(1974—),女,硕士,讲师,主要研究方向为人工智能、多Agent。

测方面的研究还比较少，比较典型的系统有加州大学伯克利分校的 Eugene 等人提出的基于查询计划的复杂事件检测技术 SASE[6]，该系统是基于原子事件流时间戳全序关系的假设，而在实际应用中时间戳乱序问题却不可避免；另一个是康奈尔大学提出的 Cayuga 系统[7]，该系统允许接收的事件流为乱序事件流，但事件流在到达处理引擎之前已经通过外部处理保证了事件流时间戳的有序性。这些系统均未考虑时间戳乱序问题，致使检测出的复杂事件为错误事件或无法检测出有效的复杂事件。

为了支持对乱序数据流的处理，一些系统[8,9]基于对乱序事件流进行缓存的思想，提出了在特定时间区间使用缓存区来保存历史事件的方法。文献[8]基于 SASE 的框架，描述了时间戳乱序对 SASE 的影响，提出了基于乱序数据流的复杂事件检测方法，简称为 SASE-Sort 算法。该方法在中间结果堆栈中按时间戳对事件进行排序，来保证事件的有序性。这类方法虽然可以提供精确的处理结果，保证事件处理的正确性，但却不能对乱序流及时地作出反应，产生了高的检测延迟，增加了响应时间，减少了系统吞吐量。

针对上述问题，激进型方法[10]提供了新的思路，这类方法可以对任何新到达的事件进行处理，并对所有可能的结果进行输出。文献[10]构建了一种高性能的乱序事件处理机制，该机制可以对任何新接收的事件进行匹配，而不需要将所有事件进行缓存直到所有事件到达。尽管这类方法缩短了响应时间，但处理的精度成为其主要问题，因为立即应答很难保证结果的正确性。

本文将时间区间、激进型方法与流修正方法相结合，提出了基于混合驱动的空间回收机制，建立了基于哈希结构的复杂事件乱序修正框架，并在此基础上提出了一种基于乱序修正框架的复杂事件检测算法，对复杂事件处理过程中涉及的乱序事件流进行处理，重点解决激进型方法处理过程中存在的处理精度问题。通过仿真实验研究了乱序事件的概率、产生的基本事件总数等对算法处理过程中内存的占用量、算法的执行时间、成功匹配率等性能的影响。最后以智能交通应用场景为例，说明如何基于物联网数据流，利用乱序修正框架对事件进行检测，结果验证了所提算法模型的有效性和可行性。

2 基于乱序修正框架的复杂事件检测模型

2.1 物联网语义事件定义

定义 1(基本事件) 基本事件表示具有一定语义的目标对象或它们之间某种具有时空关联的行为的发生[11]。通常，基本事件具有应用相关性，在不同的应用系统中，事件的语义也不尽相同。本文从物联网数据处理的实际需求出发，在最小化和完备性的基础上，提出了 3 类基本事件，如表 1 所列。

表 1 基本事件模型

基本事件的形式化描述	数据模型
Sensing (Sensor_id, Object_id, ts)	感知器 Sensor_id 在 ts 时刻感知/接收了目标对象 Object_id 的信息
Storing (Sensor_id, Object_id, ts)	感知器 Sensor_id 将在 ts 时刻感知到的目标对象 Object_id 的信息写入存储器
Routing (Sensor1_id, Sensor2_id, Object_id, ts)	某感知器 Seneor1_id 在 ts 时刻将感知的信息/接受的目标对象 Object_id 的信息转发给下一感知器 Sensor2_id

定义 2(复杂事件) 复杂事件是由基本事件或复杂事件与一个或多个事件操作符复合形成的事件，用于表达上层应用的语义对象之间某关系的发生，其数据形式可定义为 CE(Attr$_1$,…,Attr$_n$,Rule,ts)。其中，Attr$_i$ 为复杂事件的属性或构成元素；Rule 为构成规则，主要由事件操作符组成；ts 为复杂事件发生的时间。

定义 3(子事件) 把在一个复杂事件中定义的、被用来参与复合的事件称为复杂事件的子事件[12]。

上文给出的物联网感知过程中最常用的 3 种类型的基本事件，可简写为感知(S)、存储(M)和转发(R)。从复杂事件处理的角度来看，一个感知器完整的处理过程需要依次执行这 3 类事件，可表示为复杂事件 $SEQ(S,M,R)$。

定义 4(事件间隔) 事件间隔表示从一个事件的发生到另一个事件的发生所经历的事件区间，用 $\tau(\tau>0)$表示。

定义 5(时间戳乱序) 假设在物联网系统中需要处理复杂事件 $SEQ(S,M,R)$，如果其中的任意两个子事件 X 和 Y 满足 $ts_X<ts_Y$，且其到处理器的传输时间 T_X、T_Y 以及它们之间的事件间隔 τ 之间满足 $T_X>T_Y+\tau$，则在该系统的处理器处发生了时间戳乱序。简单的说，先发生的事件先到达事件处理器才不会产生时间戳乱序现象。

如果这种乱序现象发生频繁，可能导致数据的异常跳变[13]，检测出的复杂事件的正确率会非常低，基于此所做出的决策也是不可信的。本文针对以上问题，从复杂事件处理的角度研究有效的解决方案。

2.2 基于哈希结构的复杂事件乱序修正框架

2.2.1 复杂事件乱序修正框架描述

本文引入 Hash 结构[5,14]来保存中间结果，Hash 结构由数据域和指向主链结点的指针域构成。为了可以使用链地址法解决哈希冲突，主链结点选择基于树的孩子兄弟表示法表示，两个指针域分别指向该结点的第一个孩子结点和下一个兄弟结点，孩子结点则使用循环单链表表示。乱序修正框架如图 1 所示。

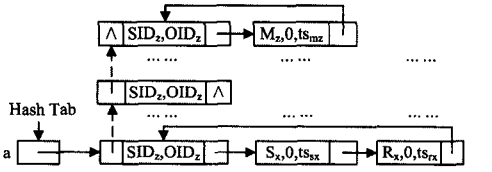


图 1 乱序修正框架

通过提取事件的两个特征，即事件感知器的 ID(Sensor_id,SID)和目标对象的 ID(Object_id,OID)，将其作为 Hash 表的键，设定哈希函数 H，使每对关键字和存储位置一一对应，构建 Hash 表。主链结点的 Data 域中只存储 Hash 表的键，即 SID 和 OID，若被接收事件的这两个属性值与主链结点中的相匹配，则将该事件按照产生时间从小到大的顺序存储于该主链结点的子链表中。为了避免信息的重复存储，同时保证信息的完整性，子链结点的 Data 域存储事件名 Name、目标感知器的 ID(Object_Sensor_id,OSID)和时间戳(Timestamp,ts)。所以，主链结点与子链结点的 Data 域中存储的数据可分别表示为〈SID,OIS〉和〈Name,OSID,ts〉，当不涉及目标感知器时，可将 OSID 的值设为 0 或做其他处理。若 Hash 地址相同，但被接收事件的 SID 和 OID 属性值与当前主链结点不匹配，则通过主链结点的兄弟域指针进行新建或继续查找。

2.2.2 复杂事件乱序修正框架优化

在复杂事件的检测过程中，事件通过链表长期存储在内存中等待与之匹配的事件，这些大量被存储的中间结果可能

导致系统的不稳定，且超过一定时间未能成功参与复杂事件匹配的子事件，没有必要进行长期保存。为了使这些无用数据不占用磁盘空间，就必须采用空间回收技术将这些不满足条件的事件从存储区删除，使空间得以释放，以供系统或其他用户使用。

定义 6（脉动） 根据上文对乱序修正框架的描述可以推测，只有在极少的几种情况（如接收的事件不匹配、复杂事件发生等）发生时才需要对内存空间进行清理，所以本文把这些情况作为影响因素，每一种情况的发生称为一个“脉动”，脉动的产生与许多因素有关。

在事件相关的应用中，通常会考虑使用两种方法对内存空间进行清理，即基于时间和基于事件的驱动机制[15]。时间驱动和事件驱动各有优缺点，在步长较大时，时间驱动可以有效地提高系统的效率，而事件驱动则表现出良好的延迟性能，准确度更高。因此，系统的效率和准确度是一个相互对立的矛盾[15]。本文结合这两种更新机制的优点及脉动事件的发生，提出了混合驱动机制（Mixed Driving Mechanism，MDM）。MDM 以时间驱动为基础，对固定时间步长内发生的事件进行评估，如果有新事件被插入到子链表或脉动发生则采用事件驱动进行更新，否则只在固定时间步长时刻对事件进行清除，同时回收内存空间。假设 t 为系统时间，t' 为固定步长，e_j（$j=1,2$）为成功插入子链表的新事件，p 为脉动事件，则基于混合驱动的空间回收机制如图 2 所示。

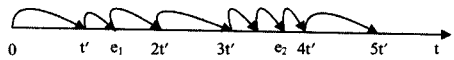


图 2 基于混合驱动的空间回收机制

为了解决中间结果可能导致的系统不稳定问题，基于上述混合驱动的空间回收机制，对本文提出的复杂事件乱序修正框架做进一步优化，具体如下：

（1）当子链表的第一个结点事件被插入时，在其对应主链结点创建并启动一个定时器 Timer，用于给所有子链结点限定一个生存期。若某一时刻复杂事件发生，则将子链表的复杂事件输出并将子链表删除，同时将主链结点的定时器删除；否则，如果定时器到期，而复杂事件尚未发生，则中断主程序的执行，调用子链表的删除程序将该定时器所在主链结点的子链表中的所有结点删除。

（2）在主链结点中提供一个用于统计该主链结点对应的子链结点数目的计数器 i，当有事件插入时，将计数器的值加 1，当该计数器与查询模式的长度相等时，表明复杂事件发生，则将复杂事件输出，同时删除该子链表。

因此，改进后的主链结点可用一个四维向量 E〈SID，OID，i，Timer〉表示，具体的乱序修正框架如图 3 所示。

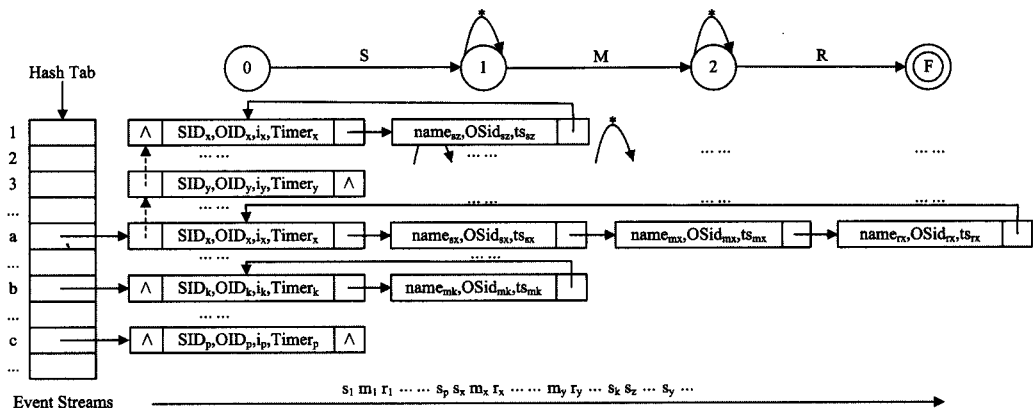


图 3 基于乱序修正框架的复杂事件检测结果

2.3 基于乱序修正框架的复杂事件检测算法

2.3.1 算法描述

针对事件之间存在的乱序问题对顺序操作符的影响，采用非确定有限自动机（NFA）结构，并结合 Hash 技术，在检测过程中修正乱序问题。

事件处理引擎接收到一个事件时，通过哈希函数 H 将该事件的关键字对〈SID，OID〉映射到相应的哈希地址，如果哈希地址对应的向量分量为空（即主链结点不存在，该类事件第一次出现），则在该地址的指针域中插入一个主链结点 E，并将相关信息〈SID，OID，i=0〉存入其中，同时在该主链结点的孩子域中插入一个子链结点，并将相关信息〈Name，OSID，ts〉存入 Data 域；主链结点中计数器 i 的值加 1，同时创建并启动定时器，判断此时 E.i 和 NFA 可识别的序列长度 Length（NFA）是否相等。如果哈希地址对应的向量分量不为空，则查找与该结点匹配的主链结点是否存在，若不存在，则新建一个主链结点；若存在，则按时间顺序插入相应的子链结点。如果某事件所在的主链结点的定时器到期，则根据混合驱动的空间回收机制，立即调用子链表的删除程序将该事件所在子链表的所有结点从对应主链结点删除。

针对一组发生时间戳乱序的原子事件流，算法 1 描述了基于乱序修正框架的复杂事件 $SEQ(S,M,R)$ 的检测方法，ORFCED 算法具体描述如下。

算法 1 基于乱序修正框架的复杂事件检测

Input：有效事件集合 S，变量 n 记录集合中有效事件的个数，复杂事件 SEQ，Hash 函数 H，定时器 Timer

1. 从 S 中抽取一个有效事件 e，并根据 e 的关键字计算 Hash 地址。
2. 若 Hash 地址对应向量分量为空，则执行步骤 3 和步骤 4。
3. 增加主链结点 E 和子链结点 N，计数器增 1 并开启定时器。
4. 若计时器的值等于 SEQ 的长度，则将该复杂事件输出，同时删除子链表。
5. 若向量分量不为空，且事件 e 对应的主链结点不存在，则执行步骤 3 和步骤 4。
6. 若向量分量不为空，且事件 e 对应的主链结点存在，则按顺序插入子链结点 N，计数器增 1。
7. 若未有定时器触发，则执行步骤 8；若某时刻某主链结点的定时器触发，则将该主链结点的子链表删除，执行步骤 8。
8. n=n−1；若 n≠0，则执行步骤 1；若 n==0，则程序停止。

Output：输出步骤 4 中成功检测的复杂事件构成的结果集

2.3.2 算法复杂度分析

上述算法的步骤5中，需要对事件e对应的主链结点进行寻找与匹配。若事件e为第$i(i\leqslant n)$个事件，则最差情况下，需要查找$i-1$次才能确定主链结点是否存在。步骤8是对算法循环的控制，则整个算法的时间复杂度为$O(n^2)$。若主链结点所占空间为K，子链结点所占空间为k，经过哈希函数H产生的哈希地址映射为区间$[0,m-1](m\leqslant n)$，且每个地址所占的空间为h，则在最坏情况下，该算法的空间复杂度为$O(m*h+n*(K+k))$。

3 仿真实验及分析

3.1 仿真设置

本节通过模拟实验对所提算法进行分析，实验通过C语言实现了算法模型，对其事件处理的性能进行了测试并对结果进行了分析。使用到的软硬件环境为Intel Core双核3.10GHz，内存2GB，Windows 7操作系统，Visual C++ 6.0编程环境。

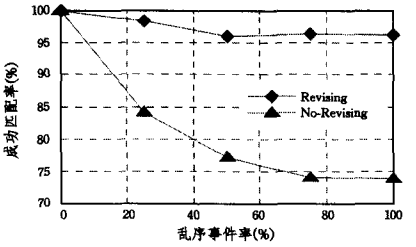
由于目前国内外尚没有通用的物联网相关的数据集，且获取海量的物联网应用数据比较困难，因此，本文实现了一个可以在不同系统下仿真的事件序列产生器，用于获取物联网乱序事件流，产生的事件存储在文档中，可以根据乱序率随机对事件进行乱序模拟。根据2.2节中对时间戳乱序问题的描述，假设查询的复杂事件为$SEQ(S,M,R)$，实验过程中使用的乱序率为0%、25%、50%、75%和100%，基本事件数量为1×10^4、2×10^4至6×10^4的数据集，通过与SASE-Sort算法进行性能对比和分析，来测试所提出的ORFCED算法的处理效率。

3.2 算法正确性验证

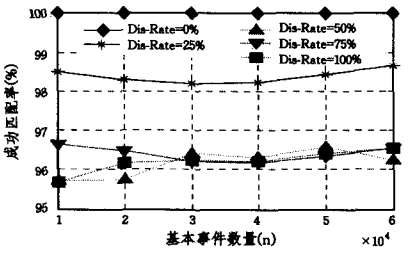
本实验在不同乱序率条件下，将未经修正及使用其他算法处理的结果的正确率与使用所提出的乱序修正算法进行对比，以验证ORFCED算法的正确性。

首先，通过6×10^4个基本事件在不同乱序率条件下，乱序修正框架使用前后复杂事件检测成功匹配率(即处理正确率)的对比，研究基于乱序修正框架的复杂事件检测算法的处理效果。由图4(a)可以看出，随着乱序事件率的增加，乱序修正框架使用前后复杂事件检测的成功匹配率总体均呈下降趋势，但经过乱序修正框架处理的复杂事件检测的成功匹配率明显高于未经处理的情况，且前者保持在较高水平。这表明，针对乱序事件流，乱序修正框架的处理效果很明显，可以大大提高复杂事件检测的正确率，提供精确的处理结果。

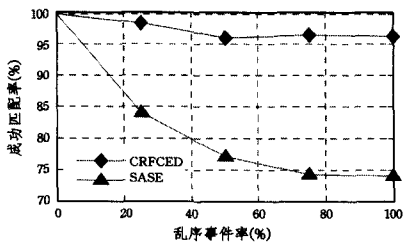
接着，通过不同乱序率条件下ORFCED算法与SASE-Sort算法的对比，研究基于乱序修正框架的复杂事件检测算法的处理效果。由图4(b)可以看出，在相同乱序事件率情况下，基本事件数量的增加对成功匹配率没有产生较大的影响，且保持在较高的水平，表明本算法具有很高的处理效率，故本实验把相同乱序率情况下的平均成功匹配率作为该乱序率对应的成功匹配率。由图4(c)可以看出，随着乱序率的增加，两种算法的成功匹配率总体均呈下降趋势，但本文提出的ORFCED算法明显优于SASE-Sort算法。这是因为ORFCED算法通过链表结构可以很好地处理结点事件的增加、删除等操作，而SASE-Sort算法基于AIS处理，在RIP的设置和右邻接栈的RIP修改过程中可能会由于不完整的事件检索，导致数据丢失，最终无法成功匹配。



(a) 修正前后复杂事件处理正确率对比



(b) 基本事件数量对成功匹配率的影响



(c) 乱序率对成功匹配率的影响

图4 修正前后复杂事件处理正确率对比

3.3 算法时延效果验证

本实验通过乱序率为100%和全序两种情况下不同算法的对比，来研究基本事件数量对算法执行时间的影响。由图5可以看出，随着基本事件数量的增加，两种算法的执行时间总体均呈增长趋势，但在基本事件数量相同的情况下，本算法处理不同乱序率事件的执行时间大致相同，而SASE-Sort算法则随着乱序率的增加而增加。这是因为在乱序事件的处理过程中，本算法无需遍历每个事件，只需查看主链表中有无与其相匹配的主链结点存在，同时利用空间回收机制对过期数据进行及时处理，减少了不必要的查询，且随着主链结点的构建，每个事件处理的平均时间会相应减少；而SASE-Sort算法不仅未对过期中间结果提出相应的处理方法，还需要在序列扫描和构建(SSC)过程中对AIS中的所有事件进行遍历，事件数量越多，遍历的时间就越长。因此，本文提出的乱序修正算法不仅可以大大提高复杂事件检测的正确率，还能对乱序事件流及时作出处理，减少不必要的延迟，缩短了系统响应时间，且当数据集增大时，本算法的优势也就更加明显。

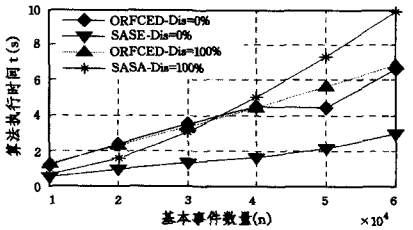
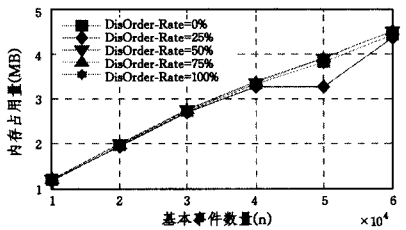


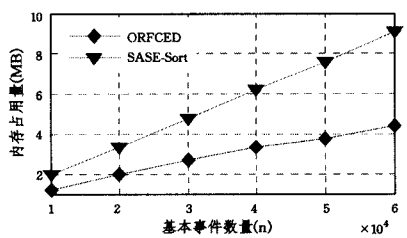
图5 基本事件数量对算法执行时间的影响

3.4 算法内存效率验证

本实验通过对不同数据集下两种算法的内存占用量进行分析，研究空间回收机制的应用对处理性能的影响。由图 6(a)可以看出，在基本事件数量相同、乱序事件率不同的情况下，事件处理所需的内存量大致相同，故本实验把该情况下事件处理占用的平均内存作为处理该数据集所需的内存占用量。由图 6(b)可以看出，随着基本事件数量的增加，两种算法的内存占用量均呈现出增长的趋势，且 SASE-Sort 算法的增长速度更快。这是因为 SASE-Sort 算法中并未提出相应的处理过期数据问题的解决方案，致使大量过期的中间结果长期存储在内存中，影响了系统的处理效率。相反，本算法可以对乱序事件流及时作出处理，并使用空间回收机制对过期数据进行及时清理，减少中间结果对内存的占用，且数据量越多效果就越明显。



(a)基本事件数量对内存占用量的影响



(b)空间回收对处理效果的影响

图 6 基本事件数量对内存占用量的影响

上述几组实验结果表明，对于乱序事件流，ORFCED 算法具有很好的处理效果，不仅可以大大提高复杂事件检测的正确率，提供精确的处理结果，还能对乱序事件流及时作出处理，减少不必要的延迟，缩短系统响应时间，同时减少内存的占用。当参与检测的事件规模足够大时，乱序事件率的增加对复杂事件处理的准确性没有显著影响，且基本事件规模越大，本算法的处理效果就越明显。

4 实例研究

本节以智能交通应用场景为例，说明如何基于物联网数据流，利用乱序修正框架对基本事件或复杂事件进行物联网复杂事件检测。

假设一个智能交通场景为某智能交通灯对应的十字交叉路口，该路口由一条东西方向的主干道和一条南北方向的支道构成。在十字路口设置有左转的交通灯，并在直行道和左转道口安装环形线圈感应该车道是否有车辆通过。所有机动车辆应遵守国家道路交通安全法规定，按照交通信号通行：红灯表示禁止通行，绿灯表示准许通行，黄灯表示警示。下面主要介绍其建模过程。

4.1 智能交通中时间戳乱序问题的描述

(1)智能交通应用场景中的基本事件

通常，智能交通应用场景中的基本事件包含交通灯的切换以及感应线圈对车辆的感知等，具体如表 2 所列。

表 2 智能交通应用场景中的基本事件类型

基本事件的形式化描述	数据模型
Toggle (Crossroad_num, Object_id, Toggle_state, Time_slot, ts)	编号为 Crossroad_num 的交通路口的编号为 Object_id 的交通灯，在 ts 时刻切换成 Toggle_state 状态，该状态持续的时间为 Time_slot
Induction (Crossroad_num, Sensor_id, Light_state, ts)	编号为 Crossroad_num 的交通路口的编号为 Sensor_id 的感应线圈在 ts 时刻，感知到有车辆通过，此时交通灯的状态为 Ligth_state

(2)智能交通应用场景中的复杂事件

基于智能交通应用场景中基本事件的定义，可将复杂事件定义为左转闯红灯，则参与左转闯红灯事件的基本事件为该复杂事件的子事件。该左转闯红灯事件由 3 个子事件构成，分别为左转交通灯由绿灯切换为红灯(G 事件)、机动车后车轮越过左转车道的停车线使感应线圈感知到车辆(A 事件)以及机动车经过马路对面时感应线圈再一次感知到车辆(B 事件)。也就是说，一个完整的左转闯红灯事件应该依次执行 G 事件、A 事件和 B 事件，即该复杂事件可表示为 $SEQ(G, A, B)$。

若某一时刻，南北路上左转的交通灯由绿色跳转为红色，此时一辆机动车由南向北行驶进入左转车道，越过停止线并继续行驶。该过程中，机动车闯红灯事件符合智能交通应用场景中复杂事件的定义，即复杂事件 $SEQ(G, A, B)$ 发生，具体的基本事件如表 3 所列。

表 3 智能交通应用场景中的基本事件

基本事件的形式化描述	含义
G (001, 001, R, 30s, 2014/10/12 9:00)	编号为 001 的交通路口的编号为 001 的交通灯，在 2014/10/12 9:00 切换成红色(R)状态，该状态持续的时间为 30s
A (001, 100, R, 2014/10/12 9:05)	编号为 001 的交通路口的编号为 100 的感应线圈在 2014/10/12 9:05 感知到有车辆通过，此时交通灯的状态为红灯(R)
B (001, 101, R, 2014/10/12 9:10)	编号为 001 的交通路口的编号为 101 的感应线圈在 2014/10/12 9:10 感知到有车辆通过，此时交通灯的状态为红灯(R)

上述过程中，任何事件发生后都将会在第一时间被送往终端处理器进行处理。但传输过程中的硬件故障、网络拥塞等原因会使先产生的事件后到达处理器，即满足了时间戳乱序的定义。

4.2 基于乱序修正框架的复杂事件检测

4.2.1 乱序修正框架设计

由上述基本事件的定义可知，同一个闯红灯事件的 3 个子事件都包含两个共同的属性 Crossroad_num 和 Ligth_state，且这两个属性均有相同的取值，则可将这两个属性作为 Hash 表的键。同时，主链结点的 Data 域中存储的这两个属性、计数器 i、定时器 Timer 以及第一个感应事件发生的时间(用来区别同一交通灯、同一状态下不同的复杂事件)，可表示为 E〈Crossroad_num, Ligth_state, ts_i, i, Timer〉；子链结点存放各自的信息，即 Toggle 事件存放的信息为〈Object_id, Time_slot, ts〉，Induction 事件存放的信息为〈Sensor_id, ts_i〉。

4.2.2 基于乱序修正框架的复杂事件检测

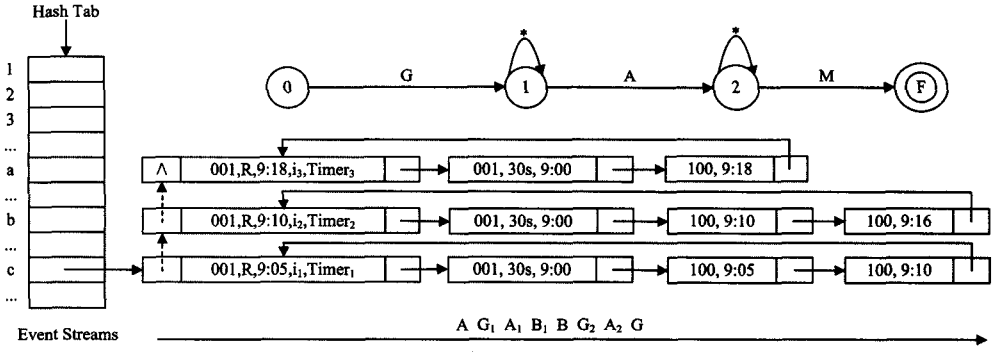
假设到达处理器的事件的顺序依次为 $A, G_1, A_1, B_1, B, G_1, A_2, G_2, G$，其中，$G, A, B$ 为闯红灯事件 P_1 的子事件，G_1，

A_1,B_1 为闯红灯事件 P_2 的子事件,G_2,A_2 为闯红灯事件 P_3 的子事件,且 P_1、P_2、P_3 为同一个红灯下连续的 3 个闯红灯事件,即 $P_1.ts<P_2.ts<P_3.ts$,$G=G_1=G_2$,具体如表 4 所列。

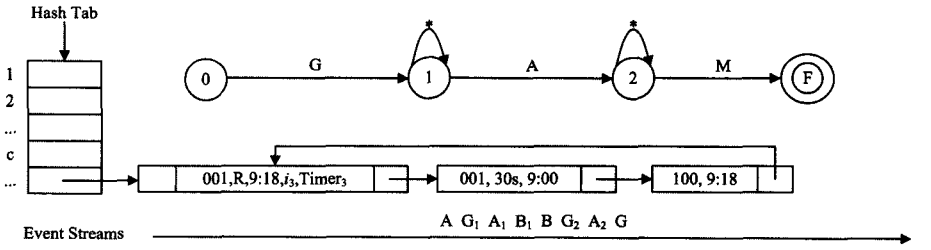
经过基于乱序修正框架的复杂事件检测模型的修正,事件流 A,G_1,A_1,B_1,B,G_1,A_2,G_2,G 中的事件 G,A,B 可以被可以被检测为复杂事件 $SEQ(G,A,B)$ 的子事件;同样,事件 G_1,A_1,B_1 可以被检测为复杂事件 $SEQ(G_1,A_1,B_1)$ 的子事件,具体检测过程如图 7(a)所示,这些子事件在复合形成复杂事件后均被删除,检测的结果如图 7(b)所示。

表 4 智能交通应用场景中的基本事件实例

复杂事件	G 事件	A 事件	B 事件
P_1	G(001,001,R,30s,14/10/12 9:00)	A(001,100,R,14/10/12 9:05)	B(001,101,R,14/10/12 9:10)
P_2	G_1(001,001,R,30s,14/10/12 9:00)	A_1(001,100,R,14/10/12 9:10)	B_1(001,101,R,14/10/12 9:16)
P_3	G_2(001,001,R,30s,14/10/12 9:00)	A_2(001,100,R,14/10/12 9:18)	



(a)基于乱序修正框架的复杂事件检测过程



(b)基于乱序修正框架的复杂事件检测结果

图 7 基于乱序修正框架的复杂事件检测

4.3 有效性分析

上述智能交通应用场景中,在没有使用基于乱序修正框架的复杂事件检测模型对事件流 A,G_1,A_1,B_1,B,G_1,A_2,G_2,G 进行修正的情况下,这些事件中仅有事件 G_1,A_1,B_1 作为复杂事件 $SEQ(G_1,A_1,B_1)$ 的子事件被检测出来,即无法检测出复杂事件 $SEQ(G,A,B)$,这意味着 ORFCED 算法较原有的方法可以提高检测的正确率,保证检测的准确性。当参与检测的事件规模逐渐增大时,使用 ORFCED 算法对海量的物联网事件进行修正,仍可以保证复杂事件处理的正确率及复杂事件检测的成功匹配率,且不会对准确性产生较大的影响。从而说明,本文提出的 ORFCED 算法应用于智能交通应用领域时,满足准确性、有效性和可行性。

同样,对于智能交通应用场景中的其他数据以及其他领域的相关数据,也可以根据本文提出的物联网语义事件定义方法进行事件化,并利用复杂事件处理技术对事件流进行有效化。对于存在时间戳乱序的事件流,还可以借助基于乱序修正框架的复杂事件检测模型进行修正处理。

结束语 随着物联网技术的应用和发展,作为关键技术之一的复杂事件处理技术已经被广泛应用于各种领域。本文针对物联网应用系统中存在的时间戳乱序问题,提出了一种可对海量实时的物联网乱序事件流进行及时高效处理的复杂事件检测算法,该算法在混合驱动的空间回收机制的基础上建立了基于哈希结构的复杂事件乱序修正框架,在对高速事件流进行实时处理的同时,可以实现对内存空间的及时清理。仿真结果表明,本文提出的 ORFCED 算法较 SASE-Sort 算法有更高的处理效率,可以减少内存占用量、执行时间,提高成功事件的匹配率等,尤其适用于大规模的数据流处理。此外,本文还以智能交通应用场景为例,展示了物联网语义事件定义以及基于乱序修正框架的复杂事件检测的完整过程,该过程验证了本文提出的算法模型的准确性与可行性。未来将进一步考虑相同子事件的共享问题并给出相关的代价评估模型及优化方法。

参 考 文 献

[1] Yuan L Y, Wang X C, Gan J H. A semantic-based spatio-temporal data model for Internet of Things [J]. Journal of Convergence Information Technology, 2013, 8(6): 1159-1168

[2] Yuan L Y, Wang X C. Study on IOT Spatio-temporal Data Description Model Based on Semantics [C]// Proceedings of the 2013 International Conference on Control Engineering and Communication Technology. Nanning, 2013: 759-764

[3] 史喜阳,孙棣华,宋潇潇. 基于 CEP 的 RFID 数据处理模型研究[J]. 自动化技术与应用,2008,27(4):74-76
Shi X Y, Sun D H, Song X X. A CEP-Based RFID Data Processing Model [J]. Techniques of Automation and Applications, 2008, 27(4): 74-76

(下转第 153 页)

户位置确定度较低，平均伪装区域面积较小，能够更好地适用于现实环境。

参 考 文 献

[1] 周傲英，杨彬，金澈清，等. 基于位置的服务：架构与进展[J]. 计算机学报，2011，34(7)：1155-1171
Zhou A Y，Yang B，Jin C Q，et al. Location-based service：Architecture and Progress [J]. Chinese Journal of Computers，2011，34(7)：1155-1171

[2] Xu J，Tang X，Hu H，et al. Privacy-conscious location-based queries in mobile environments [J]. IEEE Transactions on Parallel and Distributed Systems，2010，21(3)：313-326

[3] Pan X，Xu J，Meng X. Protecting location privacy against location-dependent attacks in mobile services[J]. IEEE Transactions on Knowledge and Data Engineering，2012，24(8)：1506-1519

[4] Mokbel M F，Chow C Y，Aref W G. The new Casper：query processing for location services without compromising privacy[C]// Proceedings of the 32nd International Conference on Very Large Data Bases. 2006：763-774

[5] Gedik B，Liu L. Protecting location privacy with personalized k-anonymity：Architecture and algorithms [J]. IEEE Transactions on Mobile Computing，2008，7(1)：1-18

[6] Chow C，Mokbel M F，Liu X. A peer-to-peer spatial cloaking algorithm for anonymous location-based services [C] // Proceedings of the Annual ACM International Symposium on Advances in Geographic Information Systems(GIS'06). Virginia，USA，2006：171-178

[7] 黄毅，霍峥，孟小峰. Coprivacy：一种用户协作无匿名区域的位置隐私保护方法[J]. 计算机学报，2011，34(10)：1976-1985
Huang Y，Huo Z，Meng X F. CoPrivacy：A Collaborative Location Privacy-Preserving Method without Cloaking Region[J]. Chinese Journal of Computers，2011，34(10)：1976-1985

[8] Solanas A，Martínez-Ballesté A. A TTP-free protocol for location privacy in location-based services[J]. Computer Communications，2008，31(6)：1181-1191

[9] Ghinita G，Kalnis P，Skiadopoulos S. PRIVE'：anonymous location based queries in distributed mobile systems[C]// Proceedings of the Sixteenth International World Wide Web Conference(May 2007). Alberta，Canada，2007：371-380

[10] Yiu M L，Jensen C S，Huang X，et al. SpaceTwist：Managing the trade-offs among location privacy，query performance，and query accuracy in mobile services[C]// Proceedings of the IEEE International Conference on Data Engineering(ICDE'08). Cancun，Mexico，2008：366-375

[11] Bettstetter C，Hartenstein H，Pérez-Costa X. Stochastic properties of the random waypoint mobility model[J]. Wireless Networks，2004，10(5)：555-567

[12] Campos C A V，de Moraes L F M. A markovian model representation of individual mobility scenarios in ad hoc networks and its evaluation[J]. EURASIP Journal on Wireless Communications and Networking，2007，1(3)

[13] Camp T，Boleng J，Davies V. A survey of mobility models for ad hoc network research[J]. Wireless Communications and Mobile Computing，2002，2(5)：483-502

(上接第 131 页)

[4] Wang Fu-sheng，Liu Shao-rong，Liu Pei-ya. Complex RFID Event Processing[J]. The VLDB Journal，2009，18(4)：913-931

[5] 徐传飞，林树宽，乔建忠，等. 高密度 RFID 事件流上的复杂事件检测[J]. 东北大学学报(自然科学版)，2012，33(5)：627-631
Xu C F，Lin S K，Qiao J Z，et al. Complex event detection on high-density RFID event streams [J]. Journal of Northeastern University (Natural Science)，2012，33(5)：627-631

[6] Wu E，Diao Y，Rizvi S. High-performance complex event processing over streams [C]// Proceedings of ACM Conference on Management of Data. Chicago，2006：407-418

[7] Demers A J，Gehrke J，Panda B，et al. Cayuga：A general purpose event monitoring system [C]// Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research. California，2007：412-422

[8] Li M，Liu M，Ding L P，et al. Event stream processing with out-of-order data arrival [C]// Proceeding of the 27th International Conference on Distributed Computing Systems Workshops. Toronto，2007：67-74

[9] Brito A，Fetzer C，Sturzrehm H，et al. Speculative out-of-order event processing with software transaction memory[C]// DEBS 2008. New York：ACM，2008

[10] Li C W，Gu Y，Yu G，et al. Aggressive Complex Event Processing with Confidence over Out-of-Order Streams [J]. Journal of Computer Science and Technology，2011，26(4)：685-696

[11] 王中强. RFID 复杂事件实时查询处理及其优化策略[D]. 武汉：华中科技大学，2011
Wang Z Q. Real-time query processing and optimizing strategy for RFID complex events [D]. Wuhan：Huazhong University of Science and Technology，2011

[12] 叶蔚，黄雨，赵文，等. 基于 Petri 网的 RFID 中间件中复合事件检测研究[J]. 电子学报，2008，36(12A)：1-8
Ye Wei，Huang Yu，Zhao Wei，et al. Research on Composite Event Detection in RFID Middleware Based on Colored Petri Net [J]. Acta Electronica Sinica，2008，36(12A)：1-8

[13] 曹原，刘英博，肖利，等. 状态监测数据流时间乱序问题建模与研究[J]. 计算机集成制造系统，2013，19(12)：2960-2967
Cao Yuan，Liu Ying-bo，Xiao Li，et al. Modeling on time out-of-order problem of condition monitoring data stream [J]. Computer Integrated Manufacturing Systems，2013，19(12)：2960-2967

[14] 刘海龙，李战怀. 基于 ENFA 的乱序 RFID 复杂事件检测算法[J]. 华中科技大学学报(自然科学版)，2010，38(1)：25-30
Liu Hai-long，Li Zhan-huai. Out-of-order RFID complex event detecting algorithm based on ENFA [J]. Journal of Huazhong University of Science and Technology (Nature Science Edition)，2010，38(1)：25-30

[15] 马宝林，孙济洲，于策. 基于混合时间-事件驱动的信任值更新机制[J]. 计算机应用，2006，26(10)：2289-2291
Ma Bao-lin，Sun Ji-zhou，Yu Ce. Trust value updating mechanism based on mixed time-event [J]. Journal of Computer Applications，2006，26(10)：2289-2291