基于 CEP 的消防物联网火警误报监测①

祝锡永, 韩会平

(浙江理工大学 管理科学与工程研究所, 杭州 310018)

摘 要: 针对消防设施管理缺位、消防设施完好率低等引起的火警误报漏报等问题, 研究消防物联网火警实时监测与分析系统; 将复杂事件处理技术引入到消防安全远程监控系统, 采用事件处理语言 EPL 和复杂事件处理引擎 Esper, 开发一个面向消防物联网火警误报监测的应用实例, 实现从消防物联网数据流处理、复杂事件规则验证到消防物联网火警误报实时甄别的整个监测过程.

关键词: 消防物联网; 复杂事件处理; 实时监控; 火警误报

Discrimination of False Fire Reporting Based on Internet of Things and Complex Event Processing

ZHU Xi-Yong, HAN Hui-Ping

(Institute of Management Science and Engineering, Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: To solve the problems of false fire reporting caused by inefficient management or insufficient maintenance of firefighting facilities, research on internet-based real time fire monitoring and on-line data analysis is presented. The concept of Complex Event Processing is introduced into a remote firefighting security monitoring system. In the system, an event processing language, together with its engine (Esper), is used to illustrate the events of false fire reporting. As a result, the entire process including data stream processing from IOT, events and rules modeling, and on-line discrimination of false fire reporting, are demonstrated in the system.

Key words: internet of things of firefighting facilities; complex event processing; real-time monitoring; false fire reporting

1 引言

“智慧城市”是目前研究的热点课题之一, 我国许多城市相继推出了“智慧城市”发展战略.“智慧消防”是“智慧城市”的重要组成部分, 它以移动计算、智能识别、智能处理、虚拟仿真等现代信息通信技术为支撑, 以数字地理信息为基础, 结合移动定位系统、数字通信技术和计算机软件平台, 为城市消防火灾防控、应急指挥、部队管理等工作建立智能化消防数字平台, 提供消防装备、应急预案、消防水源、建筑固定消防设施等信息的智能采集、汇总、分析、发布及辅助决策等手段, 实现对城市消防安全的监测、预警、处置、指挥调度等管理功能[1].因此从技术上说, “智慧消防”是物联网技术在消防领域的具体应用.

目前国外关于物联网技术在消防领域主要应用在火灾联网监测上.德国、日本、美国等采用计算机与用户终端信号采集器相连, 进行对火灾自动报警设备的实时监测以及故障远程传输[2].我国对此项技术应用的研究始于上世纪 90 年代中期, 近年来得到快速发展.国内关于“智慧消防”的应用研究主要集中在消防安全远程监控管理系统上[3], 目前消防远程监控系统可以实时监控消防联网单位消防设施的运行状态, 在一定程度上实现了火灾预警, 但仍面临在分布式系统中海量报警数据的及时分析处理以及数据误报和漏报的甄别等难题.

① 基金项目:国家自然科学基金(71271192)

收稿时间:2014-06-11;收到修改稿时间:2014-07-14

© 中国科学院软件研究所 http://www.c-s-a.org.cn

针对以上问题，本文提出基于复杂事件处理(Complex Event Processing, CEP)的消防物联网火警误报监测系统，通过将 CEP 技术引入到消防物联网监控系统中，实现火警误报信号的实时甄别，对于前移防灭火关口、提高社会单位建筑消防设施完好率和降低火警误报率具有很大应用价值.

2　消防物联网监测系统

消防物联网监测系统在互联网上分省级和地市级两级部署，省级监测中心分数据库服务器、综合服务应用服务器和数据交换服务器，安装综合服务子系统和数据交换子系统软件，省级监测中心还需配备相应的存储设备对数据进行备份保存.市级监测中心分数据库服务器、信息收发服务器、综合服务应用服务器、数据交换服务器和报警受理客户端电脑、支队及各大队指挥中心的火警信息客户端电脑，在信息收发服务器上部署信息收发子系统软件，在综合服务应用服务器上部署综合服务子系统和数据交换子系统软件，在报警受理工作站电脑上安装报警受理客户端软件，在火警信息工作站电脑上安装火警信息客户端软件.市级监测中心还需配备相应的存储设备对数据进行备份保存.

各联网单位的消防控制室图形显示装置或用户信息传输装置等数据采集终端通过 VPN 将采集到的消防设施运行状态、报警信息、记录信息等数据信息发送到监控中心.

监控中心收发服务器设计容量为 5000 家联网单位连接，若超过 5000 家，可部署多个信息收发服务器来扩充系统容量.消防物联网监测系统网络拓扑结构如图 1 所示.

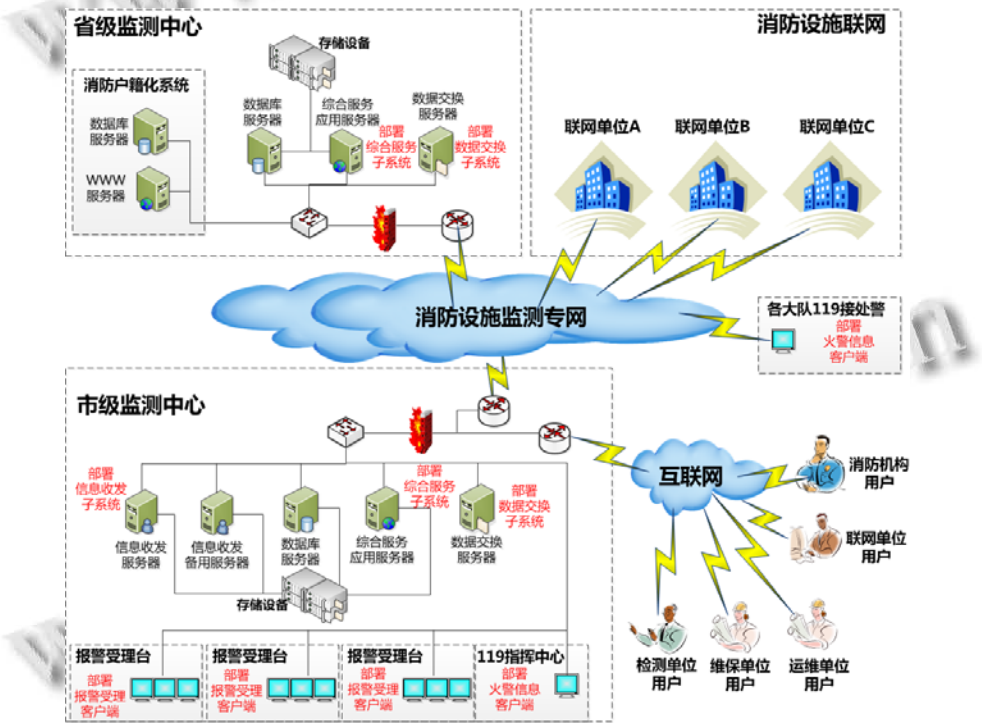

图 1　消防物联网监控系统网络拓扑结构

消防物联网监测系统是一个任务繁重的综合服务系统，自身产生数据量大.消防物联网中普遍存在的问题是火警误报、漏报，而很多报警信息大多是由于人为因素或设备故障引起的火警误报.目前对于消防物联网中海量异构传感器数据的存储与查询处理、大量传感器的智能分析与协同工作、复杂事件的自动探测与有效应对技术的研究还比较有限，因此如何合理有效地利用这些数据、对火警误报信息进行监测是研究的难点.国内学者在研究火警误报时以历史数据进行火警信号甄别的为多，主要方法包括模糊逻辑算法、BP 人工神经网络技术以及多传感器技术等.钟新跃[4]用模糊逻辑算法处理采集到的电压值，能够比较准确

© 中国科学院软件研究所 http://www.c-s-a.org.cn

判断是否发生了火灾.李文杰[5]提出了采用 ZigBee 技术与多传感器相结合的方式, 实现了对可能产生火灾的区域进行实时监控的目的, 以提高火警报警的准确性.事实上, 基于历史数据的分析技术在火警信息处理的实时性等方面依然存在很大挑战.

复杂事件处理技术(Complex event processing, CEP)能较好地解决数据量大以及数据实时性等问题, 由于 CEP 具有实时处理来自物联网底层的海量 RFID 标识与传感器感知原子事件, 以及根据既定的规则对海量的数据流进行智能产生响应的能力, 使得它可以成为物联网体系中的一个关键技术[6].本文将 CEP 概念引入消防物联网监测系统, 提出基于 CEP 的消防物联网火警误报信号甄别方法, 以更好地体现火警监控的实时性和火警报告的准确性.

3 复杂事件处理与火警误报处理

3.1 CEP 概述

3.1.1 CEP 概念

CEP 是一种新兴的数据处理技术, 其理论雏形起源 90 年代的主动数据库研究.复杂事件处理是将传统的先存储静态数据再进行数据的挖掘与处理的思路, 转变为即时处理无限的动态事件流, 能够高效、即时地在数据流中过滤出相对少量的具有应用意义的信息, 其核心是流式数据处理技术[7].复杂事件处理设定时间窗口限制, 通过事件层次结构、事件之间的关系, 使用事件处理语言进行事件关联和抽象, 得出有助于决策的数据或警信息等.

3.1.2 复杂事件描述语言及相关定义

复杂事件描述语言(Event Processing Language, EPL)是 CEP 的重要组成部分, 它是一种类 SQL 语言, 沿用 SQL 语句的描述方式, 把传统数据库的处理对象由数据表换为发生的事件.EPL 主要有事件代数表达式、数据流查询语言和产生式规则等三种类型[8].事件代数方法提供原子事件组合成复合事件的代数操作, 通过使用组合操作构造复合事件描述.

本文首先采用事件代数方法表示原子事件如何关联构成复杂事件, 然后采用数据流查询语言将输入流中的事件转换成数据库中的关系, 并在这些关系上执行查询, 最后将查询结果转换成输出数据流.

输入数据流中的事件称为简单事件或原子事件, 而原子事件关联形成的事件流就是复杂事件.复杂事件是原子事件的抽象与整合, 是根据用户预定义的规则将无序、分散的原子事件相互关联起来形成的事件流.这里对原子事件、复杂事件分别作如下定义:

定义 1. 设 E 为复杂事件处理引擎可检测到的原子事件的有限集合.对于, $\forall e \in E$ 有 $e=(i,a,t_0,t_1)$, 其中 i 是事件的标识符, 即对于任意两个$\forall e_i$与$\forall e_j$, 有 $i \neq j$; a 表示事件 e 的属性的有限集合, 即 $a=\{a_1, a_2,\ldots, a_n\}$,$n \geq 0$; t_0、t_1 表示事件的开始时间和结束时间.若事件 e 为间隔时间戳, 则有 $t_0 \geq t_1$; 若事件 e 为点时间戳, 则有 $t_0=t_1$.

定义2. 设复杂事件C是一系列满足特定关系模式 $R(r_1, r_2,\ldots\ldots, r_m)$,$m \geq 0$的原子事件组的有限集合C=(H, begin, end), 其中H表示构成此复杂事件的原子事件的有限集合, 即$H=\{e_1, e_2,\ldots\ldots, e_n\}$, $n \geq 0$; begin、end分别表示复杂事件起始时间戳和结束时间戳, $begin=min(e_1, t_0)$, $end=max(e_i, t_1)$, $e_i \in H$.

3.1.3 复杂事件处理引擎

复杂事件处理的核心是其事件处理引擎.在复杂事件处理系统中, 将生成事件流的实体称为事件源, 根据一定的规则对输入事件流进行处理并产生输出流的实体称为复杂事件处理引擎, 输出流的接受实体为事件槽.目前比较成熟的复杂事件处理引擎包括微软的 Microsoft StreamInsight 引擎和 Esper 引擎等.

本文采用 Esper 引擎进行复杂事件处理, 它是一个事件流处理和复杂事件处理框架, 可以监测事件流并在特定事件发生时触发既定动作, 属于一种基于规则的事件处理引擎[9].Esper 提供事件模式匹配和事件流查询这两种机制处理事件, 其中事件流查询将实时数据流转成实时事件流, 并以 Object、Map 和 Node 形式的事件发送到 Esper 引擎中[10].在 Esper 引擎中事先使用 EPL 语言创建系统需要的各类查询(Statements), 并对这些查询进行注册, 然后根据定义的 EPL 检测事件流, 输出数据对象 POLO, 并通过输出适配器将其转换成实时数据流, 最后流出引擎.

3.2 火警误报复杂事件描述

火警误报信息的实时甄别是一个复杂事件, 本文从下列三种情境构造相应的复杂事件以甄别火警误报情况: ①同一消防设施短时间内连续报警次数过多; ②消防故障信号与火警信号间隙式地关联发生; ③同一楼层范围内消防设施联动报警.这里将消防物联网中联网单位发送的消防设施的运行状态信息数据流作为输入事件流, 对火警误报监测系统中的相关变量、

© 中国科学院软件研究所 http://www.c-s-a.org.cn

原子事件以及复杂事件进行定义.

(1) 变量定义

FacID: 消防设施设备标识符;

FacName: 消防设施设备名称;

FacSpec: 消防设施设备型号;

FacLoc: 消防设施设备位置;

FacSts: 消防设施运行状态(0-正常, 1-火警; 2-故障);

ComID: 消防设施所属联网单位;

BuilID: 消防设施所在建筑物;

FlrID: 消防设施所在楼层;

TimeStamp: 状态变化时间.

(2) 火警原子事件定义

设 E 为消防物联网火警误报监测系统中事件集合:

$$E=\{e_n \mid n=1,2...n\},\ e_n=\{i, a, t_0, t_1\}$$

FE 为消防设施运行状态事件:

$FE=\{e_n \mid n=1,2...n\}$, $e_n=\{i, a, t_0, t_1\}$,

a={ FacID, FacName, FacSpec, FacLoc, FacSts, ComID, BuilID, FlrID, TimeStamp}

(3) 火警复杂事件定义

事件 1. 在 T 时间段内, 消防设施 ID 为 m 的消防设施, 连续发出火警信号且次数超过数量阀值 v, 其代数表示如下:

$$C_1=\sigma_\theta(FE)_T$$
$$\theta=(FacID=m)\wedge(FacSts=1)\wedge(SUM(FacID)\geq v)$$

事件 2. 在 t_0 时刻内, 某设备标识符为 m、所属单位为 i、建筑物为 j、楼层为 k 的消防设施发出故障信号; 继后在 t_1 时刻, 该消防设施发出火警信号, 且两者发出信号的时间间隔小于阀值 t, 其代数表示如下:

$$C_2\rightarrow C_3$$
$$C_2=\sigma_{\theta_1}(FE)t_0$$
$$C_3=\sigma_{\theta_2}(FE)t_1$$
$$\theta_1=(ComID=i)\wedge(BuliID=j)$$
$$\wedge(FlrID=k)\wedge(FacID=m)\wedge(FacSts=2);$$
$$\theta_2=(ComID=i)\wedge(BuliID=j)$$
$$\wedge(FlrID=k)\wedge(FacID=m)\wedge(FacSts=1);$$
$$t_1-t_0\leq t$$

这里, 操作符→表示事件 C_2 先于 C_3 发生, 事件不可重叠, 即 C_2 结束之后 C_3 发生.

事件 3. 某联网单位为 i、建筑为 j、楼层为 k 的同一楼层中有三个消防设施 A、B、C, 在 T 时间段内, 联动发出火警信号, 且报警次数超过阀值 v, 其代数表示如下:

$$C_3=\sigma_\theta(FE)_T$$
$$\theta=(ComID=i)\wedge(BuilID=j)\wedge(FlrID=k)$$
$$\wedge(FacSts=2)\wedge((FacID=A)\vee(FacID=B)$$
$$\vee(FacID=C))\wedge(SUM(FlrID)\geq v);$$

在系统实现时, 可根据上述事件制定相应的规则, 并由 EPL 语句描述.

4 系统实现

本节构造火警误报监测 CEP 应用程序并在物联网监测系统中进行运行验证. 场景设置: 设某联网单位 comA, 其建筑 buildA, 楼层 FlrA, 其中设有消防设施 A_0、A_1、A_2, 其中消防设施运行状态 facSts 取值范围[0-2](0 为正常; 1 为火警报警; 2 为故障报警). 这里采集 10 万条消防设施运行状态数据进行分析. 系统开发环境为: Windows Server2008、MyEclipse 6.5、Java(JDK 版本 1.6)、Esper 引擎 5.0 版本等.具体实现过程如下:

(1)EPL 规则定义

根据火警误报事件的相关代数定义和事件描述, 使用 EPL 语句给出相应规则的定义.

规则 1. 对应 3.2 中事件 1, 监测短时间内(本文设置为 300 秒)消防物联网监控系统中同一消防设施发生火警的次数, 对应的 EPL 语句为:

```
SELECT COUNT(*) FROM FE
(FacID:A1,FacSts:1).win: time(300 sec);
```

规则 2. 对应 3.2 中事件 2, 监测消防设施某一时刻发出故障报警并在短时间内(本文设为 10 秒)又发出火警报警, 可以判断该设施发出的火警信息为误报, 对应的 EPL 语句为:

```
SELECT a.facID as facID FROM
pattern [every a=FE(facSts='2')->(timer:
interval(10 sec) and
b=FE(facSts='1',b.facID=a.facID))];
```

规则 3. 对应 3.2 中事件 3, 监测短时间内, 同一楼层内消防设施 A_0、A_1、A_2 的报警次数, 当消防设施联动报警且火警报警概率较高的可判断为真实火警, 而概率较低的则可判断为火警误报, 对应的 EPL 语句为:

```
SELECT COUNT(FacID), FacID FROM
FE.win:time(300 sec) WHERE (FacID=A0 or
FacID=A1or FacID =A2) and FacSts=1 GROUP BY
FacID.
```

(2)程序实现

© 中国科学院软件研究所 http://www.c-s-a.org.cn

Esper 引擎处理程序主要包括三个部分: ①EPServiceProvier: 用于创建引擎、线程和事件流.这里利用随机产生的联网单位消防设施运行状态数据封装为对象作为输入事件流; ②EPStatement: 事件查询语言, 用于定义事件规则, 通过引擎注册该 statement.这里根据事先定义的相关事件描述, 编写判断火警误报的 EPL 语句; ③UpateListener: 事件处理器, 是 Esper 提供的接口, 用于监听 EPL 在引擎中的运行情况, 即事件进入并产生结果时会触发 UpateListener.本程序核心代码如下, 运行结果如图 2、图 3 所示.

```
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A2 facName:facName2 facSpec:facSpec2 facLoc:facLoc2 facSts:0 timeStamp:2014-06-29 22:47:50
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A0 facName:facName0 facSpec:facSpec0 facLoc:facLoc0 facSts:2 timeStamp:2014-06-29 22:47:51
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A1 facName:facName1 facSpec:facSpec1 facLoc:facLoc1 facSts:2 timeStamp:2014-06-29 22:47:52
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A0 facName:facName0 facSpec:facSpec0 facLoc:facLoc0 facSts:1 timeStamp:2014-06-29 22:47:53
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A2 facName:facName2 facSpec:facSpec2 facLoc:facLoc2 facSts:2 timeStamp:2014-06-29 22:47:54
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A1 facName:facName1 facSpec:facSpec1 facLoc:facLoc1 facSts:0 timeStamp:2014-06-29 22:47:55
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A0 facName:facName0 facSpec:facSpec0 facLoc:facLoc0 facSts:1 timeStamp:2014-06-29 22:47:56
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A2 facName:facName2 facSpec:facSpec2 facLoc:facLoc2 facSts:0 timeStamp:2014-06-29 22:47:57
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A0 facName:facName0 facSpec:facSpec0 facLoc:facLoc0 facSts:0 timeStamp:2014-06-29 22:47:58
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A0 facName:facName0 facSpec:facSpec0 facLoc:facLoc0 facSts:0 timeStamp:2014-06-29 22:47:59
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A0 facName:facName0 facSpec:facSpec0 facLoc:facLoc0 facSts:0 timeStamp:2014-06-29 22:48:00
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A2 facName:facName2 facSpec:facSpec2 facLoc:facLoc2 facSts:0 timeStamp:2014-06-29 22:48:01
Event received:{facID=A0, beforeFacSts=2, afterFacSts=1}
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A1 facName:facName1 facSpec:facSpec1 facLoc:facLoc1 facSts:2 timeStamp:2014-06-29 22:48:02
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A1 facName:facName1 facSpec:facSpec1 facLoc:facLoc1 facSts:0 timeStamp:2014-06-29 22:48:03
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A1 facName:facName1 facSpec:facSpec1 facLoc:facLoc1 facSts:2 timeStamp:2014-06-29 22:48:04
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A1 facName:facName1 facSpec:facSpec1 facLoc:facLoc1 facSts:2 timeStamp:2014-06-29 22:48:05
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A0 facName:facName0 facSpec:facSpec0 facLoc:facLoc0 facSts:0 timeStamp:2014-06-29 22:48:06
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A1 facName:facName1 facSpec:facSpec1 facLoc:facLoc1 facSts:2 timeStamp:2014-06-29 22:48:07
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A1 facName:facName1 facSpec:facSpec1 facLoc:facLoc1 facSts:2 timeStamp:2014-06-29 22:48:08
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A1 facName:facName1 facSpec:facSpec1 facLoc:facLoc1 facSts:0 timeStamp:2014-06-29 22:48:09
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A2 facName:facName2 facSpec:facSpec2 facLoc:facLoc2 facSts:1 timeStamp:2014-06-29 22:48:10
Event received:{facID=A2, beforeFacSts=2, afterFacSts=1}
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A1 facName:facName1 facSpec:facSpec1 facLoc:facLoc1 facSts:2 timeStamp:2014-06-29 22:48:11
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A1 facName:facName1 facSpec:facSpec1 facLoc:facLoc1 facSts:1 timeStamp:2014-06-29 22:48:12
Event received:{facID=A1, beforeFacSts=2, afterFacSts=1}
Event received:{facID=A1, beforeFacSts=2, afterFacSts=1}
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A1 facName:facName1 facSpec:facSpec1 facLoc:facLoc1 facSts:0 timeStamp:2014-06-29 22:48:13
Event received:{facID=A1, beforeFacSts=2, afterFacSts=1}
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A1 facName:facName1 facSpec:facSpec1 facLoc:facLoc1 facSts:2 timeStamp:2014-06-29 22:48:14
```

图 2 程序运行结果(1)

```
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A0 facName:facName0 facSpec:facSpec0 facLoc:facLoc0 facSts:1 timeStamp:2014-06-29 22:23:27
Event received:{facID=A0, totalFireCount=30}
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A0 facName:facName0 facSpec:facSpec0 facLoc:facLoc0 facSts:1 timeStamp:2014-06-29 22:23:28
Event received:{facID=A0, totalFireCount=31}
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A2 facName:facName2 facSpec:facSpec2 facLoc:facLoc2 facSts:0 timeStamp:2014-06-29 22:23:29
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A0 facName:facName0 facSpec:facSpec0 facLoc:facLoc0 facSts:0 timeStamp:2014-06-29 22:23:30
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A2 facName:facName2 facSpec:facSpec2 facLoc:facLoc2 facSts:2 timeStamp:2014-06-29 22:23:31
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A0 facName:facName0 facSpec:facSpec0 facLoc:facLoc0 facSts:0 timeStamp:2014-06-29 22:23:32
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A2 facName:facName2 facSpec:facSpec2 facLoc:facLoc2 facSts:1 timeStamp:2014-06-29 22:23:33
Event received:{facID=A2, totalFireCount=38}
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A2 facName:facName2 facSpec:facSpec2 facLoc:facLoc2 facSts:1 timeStamp:2014-06-29 22:23:34
Event received:{facID=A2, totalFireCount=39}
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A1 facName:facName1 facSpec:facSpec1 facLoc:facLoc1 facSts:2 timeStamp:2014-06-29 22:23:35
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A1 facName:facName1 facSpec:facSpec1 facLoc:facLoc1 facSts:0 timeStamp:2014-06-29 22:23:36
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A2 facName:facName2 facSpec:facSpec2 facLoc:facLoc2 facSts:1 timeStamp:2014-06-29 22:23:37
Event received:{facID=A2, totalFireCount=40}
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A2 facName:facName2 facSpec:facSpec2 facLoc:facLoc2 facSts:0 timeStamp:2014-06-29 22:23:38
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A0 facName:facName0 facSpec:facSpec0 facLoc:facLoc0 facSts:2 timeStamp:2014-06-29 22:23:39
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A2 facName:facName2 facSpec:facSpec2 facLoc:facLoc2 facSts:2 timeStamp:2014-06-29 22:23:40
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A2 facName:facName2 facSpec:facSpec2 facLoc:facLoc2 facSts:2 timeStamp:2014-06-29 22:23:41
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A2 facName:facName2 facSpec:facSpec2 facLoc:facLoc2 facSts:2 timeStamp:2014-06-29 22:23:42
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A1 facName:facName1 facSpec:facSpec1 facLoc:facLoc1 facSts:2 timeStamp:2014-06-29 22:23:43
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A2 facName:facName2 facSpec:facSpec2 facLoc:facLoc2 facSts:2 timeStamp:2014-06-29 22:23:44
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A0 facName:facName0 facSpec:facSpec0 facLoc:facLoc0 facSts:2 timeStamp:2014-06-29 22:23:45
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A2 facName:facName2 facSpec:facSpec2 facLoc:facLoc2 facSts:2 timeStamp:2014-06-29 22:23:46
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A0 facName:facName0 facSpec:facSpec0 facLoc:facLoc0 facSts:0 timeStamp:2014-06-29 22:23:47
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A1 facName:facName1 facSpec:facSpec1 facLoc:facLoc1 facSts:1 timeStamp:2014-06-29 22:23:48
Event received:{facID=A1, fireCount=26}
Event received:{facID=A1, totalFireCount=26}
Sending facevent: comID:comA buildID:buildA flrID:flrA facID:A2 facName:facName2 facSpec:facSpec2 facLoc:facLoc2 facSts:1 timeStamp:2014-06-29 22:23:49
Event received:{facID=A2, totalFireCount=41}
```

图3 程序运行结果(2)

```
//将封装的对象作为输入事件流
  public static void GenerateRandomTick(EPRuntime cepRT) throws
InterruptedException {
        int id = (int) generator.nextInt(3);
        String comID = "comA";
        String buildID = "buildA";
        String FlrID = "FlrA";
        String facID = "A"+id;
        String facName = "FacName"+id;
        String FacSpec = "FacSpec"+id;
        String FacLoc = "FacLoc"+id;
        String FacSts = (int) generator.nextInt(3)+"";
        Thread.sleep(2000);
        String timeStamp = new
      SimpleDateFormat("yyyy-MM-ddHH:mm:ss").format(new Date());
      FacEventfacevent=newFacEvent(comID,buildID,FlrID,facID,facNam
e,FacSpec,FacLoc,FacSts,timeStamp;
      System.out.println("Sending facevent: " + facevent);
        cepRT.sendEvent(facevent); }
//Esper复杂事件处理过程
  public static void main(String[ args) {
        Configuration cepConfig = new Configuration();
        cepConfig.aEventType("FacEvent", FacEvent.class.getName());
//注册引擎
        EPServiceProvier cep =
        EPServiceProvierManager.getProvier("myCEPEngine",
        cepConfig);
        EPRuntime cepRT = cep.getEPRuntime();
        EPAministrator cepAm = cep.getEPAministrator();
//根据EPL生成statement并用引擎注册
        EPStatement cepStatement1= cepAm.createEPL("select
        count(*) as fireCount from FE(facID='A1',
        FacSts='1').win: time(300 sec)");
        EPStatement cepStatement2= cepAdm.createEPL("select
        a.facID as facID,a.facSts as beforeFacSts,"
        +"b.facSts as afterFacSts from pattern  [ "
```


© 中国科学院软件研究所 http://www.c-s-a.org.cn

```
        +"every a=FacEvent(facSts='2') ->
        (timer:interval(10 sec) "
        + "and b=FacEvent(facSts='1',b.facID=a.facID))]");
    EPStatement cepStatement3 = cepAm.createEPL("select
    count(facID),facID from FE.win: time(300 sec)"
    +"where (facID ='A0' or facID='A1' or
    facID='A2') an FacSts='1' group by facID");
    //通过实现Esper提供的UpateListener接口生成listener
    cepStatement1.addListener(new CEPListener());
    cepStatement2.addListener(new CEPListener());
    cepStatement3.addListener(new CEPListener());
    //随机采集10万条消防设施运行状态信息事件流, 并把事件流通
过cepRT.sendEvent()发送到Esper引擎, 并根据定义的EPL规则进行相关
处理
      for (int i = 0; i < 100000; i++) {
        try {
            GenerateRanomTick(cepRT);
        } catch (InterrupteException e) {
            e.printStackTrace();
        }}}
```

程序采集 10 万条消防设施运行数据来验证复杂事件处理技术监测报警事件的有效性.图 2、图 3 分别截取了 Esper 引擎根据 EPL 语句监测输出的部分数据流, 图 2 显示消防设施 A_0 在 2014-6-29 22:47:51 时刻发生故障, 而在 22:47:53 时刻发生火警且报警与故障发生时间间隔小于 10 秒, A_2 在 2014-6-29 22:48:11 时刻发生故障, 而在 22:48:12 时刻发生火警且报警与故障发生时间间隔小于 10 秒, 因此, 可以判断消防设施 A_0 在 22:47:53 时刻的火警为误报, A_2 在 22:48:12 时刻的火警为误报, 同理, 可以判断任意 10 秒内消防设施先故障报警并在 10 秒内的火警报警为误报.图 3 显示其中消防设施 A_1 在短时间内(本文截取开始 300 秒的数据)发生火警报警 26 次, 而联网单位同一楼层中的消防设施 A_0、A_1、A_2 短时间内联动报警分别为 31 次、26 次和 41 次, 对应的概率分别为 32%、26%和 42%, 即同一楼层内消防设施 A_2 发生真实火警的概率最高, 而其他两点则相对较低.根据上述结果, 可模糊判断消防设施 A_2 发生火警, 而 A_0、A_1 则为误报.同理, 可以判断任意 300 秒内消防设施报警情况.这种基于 CEP 的火警误报监测以事件流(而非历史数据)为处理对象, 具有很强的实时性, 在一定程度上可以减少火警误报漏报率, 对于前移防灭火关口、提高单位建筑消防设施完好率、提高社会火灾防控水平、减少火灾损失, 具有十分重要的意义.

5 结语

消防物联网监控系统中普遍存在的问题是由社会单位自动消防设施管理缺位、消防设施完好率低等原因引起的消防火警误报漏报.本文将 CEP 引入消防物联网火警误报监测系统, 设计了火警误报情境, 并构造了其复杂事件及判别规则, 使用 EPL 语句和 Esper 引擎编写程序, 实现了火警误报信号的实时甄别, 比较完整给出了复杂事件处理平台的应用技术与开发过程, 验证了本文方法的可行性与实用性, 在一定程度上解决了消防远程监控系统中的火警漏报、误报问题.下阶段主要工作包括: 建立复杂事件库及相应规则库、开发事件规则库管理系统及其自动触发机制、对历史数据分析与挖掘技术在复杂事件规则中的应用进行深入研究.

参考文献

1 丁祥郭.“智慧消防”建设与发展的思考.计算机安全,2012,(10):66–69.

2 胡丽霞.基于无线网络的消防监测系统的设计与实现[学位论文].大连:大连海事大学,2010.

3 周志军,邵新.城市消防安全远程监控系统的现状与发展探讨.2011 中国消防协会科学技术年会论文集.288–291.

4 钟新跃.模糊算法在智能火灾报警器中的应用.制造业自动化,2009,31(8):128–130.

5 李文杰,廖晓纬,沈晓波.基于 ZigBee 的多传感器报警设计.计算机工程与科学,2013,35(7):175–180.

6 荆心,张璟,李军怀.复杂事件处理技术及其在物流物联网中的应用.计算机应用,2013,33(7):2026–2030.

7 Cugola G, Margara A. Processing flows of information: From data stream to complex event processing. ACM Computing Surveys(CSUR), 2012, 44(3): 15.

8 汤玲丽.复杂事件处理引擎关键技术研究[学位论文].哈尔滨:哈尔滨工程大学,2012.

9 蔡昭权,索剑,汪华斌,卢庆武,罗伟.基于 Esper 和 Nagios 的网络监控系统设计与实现.计算机工程与科学,2012,34(9):8–12.

10 姚敦红,彭小宁,石元泉.基于 Pushlet 与 Esper 轻量级实时 Web 系统的设计.计算机应用与软件,2013,30(5):163–166.

© 中国科学院软件研究所 http://www.c-s-a.org.cn