Distributed Real-Time Complex Event Processing

Per Lindgren
based on work with
Pawel Pietrzak and Henrik Mäkitaavola
Distributed Real-Time Complex Event Processing

- **Background Complex Event Processing (CEP)**
  - Events (Atomic/Composed)
  - Queries (Temporal condition rules)
  - Mapping to Concurrent Reactive Objects

- Requirements for the use of CEP in the field of automatic monitoring and control

- Recent, on-going and future work at EISLAB
Complex | Event | Processing

- **Complex**
  - Made up of multiple detailed parts.

- **Event**
  - Something that happens

- **Processing**
  - A series of events to produce a result

Source: Wiktionary.org
Events and Complex Events

• Basic (atomic) events
  – A: church bells ringing,
  – B: the appearance of a man in a tuxedo with a woman in a flowing white gown,
  – C: rice flying through the air

  – Complex event: wedding?

W = WHEN ANY (A, B, C)
Events and Complex Events

• Basic (atomic) events
  – A: church bells ringing,
  – B: the appearance of a man in a tuxedo with a woman in a flowing white gown,
  – C: rice flying through the air

  – Complex event: wedding?

  \[ W = \text{WHEN SEQUENCE (A, B, C)} \]
Events and Complex Events

• Basic (atomic) events
  – A: church bells ringing,
  – B: the appearance of a man in a tuxedo with a woman in a flowing white gown,
  – C: rice flying through the air

  – Complex event: wedding?

  \[ W = \text{WHEN ALL (A, B, C)} \]
Events and Complex Events

Events may have a Duration (start/stop)

\[ W = \text{WHEN ALL} (A, B, C) \]
Complex Event Processing

• Most applications today in Business world
  – Algorithmic stock-trading
  – Detection of credit-card fraud.
• Etc.

Business aspects, e.g., Microsoft

Theoretical aspects, e.g., MDH
CEP vs SQL

SQL queries typically
- “one shot” (new queries created dynamically)
- return a result
- operate on database (relatively static)

CEP queries typically
- typically act as a filter (standing query)
- produce event when matching
- operate on streaming data (dynamic)
Complex Event Processing in “Our World”

- Combines data from multiple sources
- Infer events or patterns that suggest more complicated circumstances.

The goal:

- **Identify** meaningful events (such as opportunities or threats)
- **Respond**, either by emitting new events or taking some action.
Embedded System

• Designed for specific purpose

• Often operating under real-time constraints

• Limited resources
  – CPU
  – Memory
  – Communication bandwidth
Motivation

- Pushing monitoring & control down to embedded devices
  Local “intelligence”

- Inherently event-based
  Control systems traditionally scan-based current trend towards event-based

- Better responsiveness
  Can act directly when an event occurs independent of scan time

- Minimize network traffic, save bandwidth, etc.

CEDR - CRO

• CEDR : Complex Event Detection and Response
  – Query language used commercially (Microsoft)
  – Capture event types and order

• CRO : Concurrent Reactive Objects
  – Execution and concurrency model
  – React to atomic events
  – Between events: idle
  – Built in notion of time
EVENT < query name >
WHEN < event expression >
WHERE < correlation expression >
OUTPUT < instance transformation conditions >
CEDR - WHEN

- ALL($E_1, \ldots, E_k$)
- ANY($E_1, \ldots, E_k$)
- SEQUENCE($E_1, \ldots, E_k$)
- UNLESS($E_1, E_2, t$)
- etc.
WHERE

- Correlation between values in events payload

Example:

EVENT Temperature_alarm
WHEN Any(Water_temp, Air_temp) AS x
WHERE x.temperature > 100
OUTPUT Alarm x.ID
Concurrent Reactive Objects

- Reactivity
  - React to external stimuli or internal events

- Objects and state encapsulation
  - All states are encapsulated in objects O1, … , On

- Message passing and specification of timing behavior
  - ASYNC(O, m, t) : Asynchronous (may be delayed)
  - SYNC(O, m) : Synchronous call returns with a result

- Execution model behind the Timber language
CRO/Timber - Example
Event Processing in CRO

CEDR query into a CRO program

EVENT Master_alarm
WHEN UNLESS(ANY(High_temp,High_press) AS x, Button pressed , 10 seconds)
WHERE x.Tank Id = Tank #1 OR x.Tank Id = Tank #3
OUTPUT Alarm x.Tank Id
EVENT Master_alarm
WHEN UNLESS(ANY(High_temp,High_press) AS x, Button pressed , 10 seconds)
WHERE x.Tank_Id = Tank #1 OR x.Tank_Id = Tank #3
OUTPUT Alarm x.Tank_Id
An Event Expression Tree

WHEN \textbf{UNLESS}(\textbf{ANY}(\text{High\_temp}, \text{High\_press}) \text{ AS } x, \text{ Button pressed } , 10\text{ seconds})
Event Recognizer States

- **Ignore** the recognizer is not activated
- **Await** even pattern is under recognition process, but it is not completed as more input is needed
- **Accept** when the event pattern has been successfully accepted
- **Fail** when the pattern definitely is not occurring, regardless of subsequent events that will follow.

Event Pattern – Reactive Object

Status
• Ignore
• Await
• Accept
• Fail

start, drop, accept
changes state
WHEN UNLESS(ANY(High_temp, High_press), Button pressed, 10 seconds)
CEDR - CRO

$O^1$ (High_temp)

\begin{align*}
\text{Status} & := \text{Ignore} \\
\text{start:} & \\
\text{Status} & := \text{Await} \\
\text{accept:} & \\
\text{IF} \text{ Status} = \text{ Await} & \text{ THEN} \\
\text{Status} & := \text{Accept} \\
\text{ASYNC}(O^3, \text{ accept}) & \\
\text{FI} &
\end{align*}

$O^2$ (High_press)

\begin{align*}
\text{Status} & := \text{Ignore} \\
\text{start:} & \\
\text{Status} & := \text{Await} \\
\text{accept:} & \\
\text{IF} \text{ Status} = \text{ Await} & \text{ THEN} \\
\text{Status} & := \text{Accept} \\
\text{ASYNC}(O^3, \text{ accept}) & \\
\text{FI} &
\end{align*}

$O^3$ (ANY)

\begin{align*}
\text{Status} & := \text{Ignore} \\
\text{start:} & \\
\text{Status} & := \text{Await} \\
\text{ASYNC}(O^1, \text{ start}) \\
\text{ASYNC}(O^2, \text{ start}) &
\end{align*}

$O^4$ (Button_pressed)

\begin{align*}
\text{Status} & := \text{Ignore} \\
\text{start:} & \\
\text{Status} & := \text{Await} \\
\text{ASYNC}(self, \text{ drop, 10 sec, } \infty) &
\end{align*}

\begin{align*}
\text{accept:} & \\
\text{IF} \text{ Status} = \text{ Await} & \text{ THEN} \\
\text{Status} & := \text{Accept} \\
\text{ASYNC}(O^5, \text{ drop}) & \\
\text{FI} &
\end{align*}

\begin{align*}
\text{drop:} & \\
\text{IF} \text{ Status} \neq \text{ Accept} & \text{ THEN} \\
\text{Status} & := \text{Fail} \\
\text{ASYNC}(O^5, \text{ accept}) & \\
\text{FI} &
\end{align*}

$O^5$ (UNLESS)

\begin{align*}
\text{Status} & := \text{Ignore} \\
\text{start:} & \\
\text{Status} & := \text{Await} \\
\text{ASYNC}(O^3, \text{ start}) &
\end{align*}

\begin{align*}
\text{accept:} & \\
\text{IF} \text{ Status} = \text{ Await} & \text{ THEN} \\
\text{Status} & := \text{Accept} \\
\text{ASYNC}(O^{\text{Master}_\text{-alarm}}, \text{ accept}) & \\
\text{FI} &
\end{align*}

\begin{align*}
\text{drop:} & \\
<\text{default action}> &
\end{align*}
Towards a Lightweight Complex Event Processing Engine for Embedded Systems

IECON 2012 : 38th Annual Conference of the IEEE Industrial Electronics Society
Pawel Pietrzak, Per Lindgren, Henrik Mäkitaavola

• Lightweight and efficient implementation in TinyTimber (C-Code API to CRO kernel)
  – Simple states, no queues (besides inside kernel)
  – Static structure facilitates analysis
• Subset of CEDR rules (easily extendable)

Future work
• Support dynamic queries
• Support re-occurring/overlapping events
• Support real-time constraints
Distributed Real-Time Complex Event Processing

• Background Complex Event Processing (CEP)
  – Events (Atomic/Composed)
  – Queries (Temporal condition rules)
  – Mapping to Concurrent Reactive Objects

• Requirements for the use of CEP in the field of automatic monitoring and control

• Recent, on-going and future work at EISLAB
Process monitoring and control

• Well… I’m no expert, but our control group are so I asked Thomas Gustafsson et al. ;)

• We are limited by:

\[ ||P|| + ||\Delta|| = K' \quad P'*L<K'' \quad \text{(stability)} \]

• \( P, P' \) are “performance”
• \( \Delta \) are the errors
• \( L \) is the (total) delay
• \( K', K'' \) are constants

In order to improve performance, we need to reduce errors & delay

One way is to reduce inaccuracy for sensing/actuating

– sensing errors, computation errors, communication errors
– accuracy of time of data acquisition (time-stamp jitter)
– computation and communication delays
Automatic Control System

- A “distorts”
  - signal corruption, delays, jitter,
- A’ models the “distortion”
  - (e.g., compensate for delay)
  - allows for
    - sensor fusion
    - soft sensors

Model state can be updated based on the confidence (QoS) we give input/model. Based on model, model state and target we may request QoS on input.
CEP in Automatic Control System

• CEP for Control
  – Powerful mechanism to process streaming data (events can be time triggered, similar to scan-based)
  – Inherent support for event based control
  – Can connect loosely coupled Systems of Systems

• Requirements (quality)
  – Accurate time of event occurrence
  – Bound & small delay of delivery
    • processing and communication, boils down to
    • scheduling of nodes and channels
    • data representation
Distributed Real-Time Complex Event Processing

• Background Complex Event Processing (CEP)
  – Events (Atomic/Composed)
  – Queries (Temporal condition rules)
  – Mapping to Concurrent Reactive Objects

• Requirements for the use of CEP in the field of automatic monitoring and control

• Recent, on-going and future work at EISLAB
Real-Time Complex Event Processing using Concurrent Reactive Objects

- Accepted to: 2013 IEEE International Conference on Industrial Technology (ICIT 2013)
  Per Lindgren, Paweł Pietrzak, Henrik Mäkitaavola

- Addresses
  - non-deterministic (out of order) execution
  - reoccurring events
  - overlapping timing windows and
  - real-time properties
Motivation

• Traditional CEP only deals with \textit{time} in order to correlate events, not the \textit{time} to process

• Scheduling (prioritizing) is \textit{at best} achieved through assigning priority to a thread that process the query
  – \textit{Time separated from query}
  – \textit{Non compositional}
Our approach

• CRO Timing model

Message should be processed within execution window
Each clause (sub-expression) is associated with a message
CEP encoding in Timber

unless e w d1 d2 = class
    tmr = new timer
    abortT := []

start = before w + d1 action
    after w + d1 send accept

drop = before d1 action
    abortT := (<- tmr.sample) : abortT

accept = before d2 action
    t = <-tmr.sample
    if elem True (map (inW t) abortT) then
        send e.drop
    else
        send e.accept
    abortT := (filter (inWOrL t) abortT)
where
    inW t at = (at > t-w-d1) && (at < t-d1)
    inWOrL t at = at > t-w-d1-d2

result Cep {..}
A SOA Approach to Delay and Jitter Tolerant Distributed Real-Time Complex Event Processing

Submitted to:
22\textsuperscript{nd} IEEE International Symposium on Industrial Electronics
Per Lindgren, Rumen Kyusakov, Jens Eliasson, Henrik Mäkitaavola, Paweł Pietrzak

• Addresses
  – SOA and CEP in factory automation
  – Distributed and heterogeneous
  – Real-time aspects are highlighted
    • accurate time-stamping of physical events in a distributed system
    • end-to-end timing including communication and CEP query processing
Motivation

• Traditional CEP only deals with \textit{time} in order to correlate events, not the \textit{time} to process

• Scheduling (prioritizing) is \textit{at best} achieved through assigning priority to a thread that process the query
  – \textit{Time separated from query}
  – \textit{Non compositional}


Recall the Alarm CEP query...

a), b) and c) are loosely coupled systems with local time domains...

d), e) links which introduce delay
Time Synchronization in a distributed system

- **GPS/Atomic clocks**
  - Nanosecond accuracy (accuracy limited by other parts of the system architecture)

- **Network Time Protocol (NTP)**
  - Millisecond accuracy

- **Precision Time Protocol (PTP)**
  - 10s of microseconds
  - middle ground between dedicated hardware and software solutions
  - good enough for monitoring/control in many cases
Real-Time Service Oriented Architecture

- Focus on interoperability and flexibility

- Standards based “Restful” Web Services
  - CRUD operations (Create, Read, Update and Delete)

- Lightweight
  - CoAP is an IETF application protocol specification for eventing and subscriptions
  - Efficient XML Interchange (EXI) format developed by W3C
    - Relatively low CPU and Memory requirements
Real-Time Robust CEP in Timber

• Inaccurate time-stamps (synchronization jitter)
  – leads to incorrect event detection

• Delays (processing/communication)
  – *may* lead to incorrect event detection (in cases when decisions are taken prematurely)

• Jitter and Delay Tolerant CEP
  – *safe* certain and potential matches
  – *safe* (non-premature) end-end timing
Real-Time Robust CEP in Timber

- **Jitter and Delay Tolerant CEP**
  - *safe* certain and potential matches
  - *safe* (non-premature) end-end timing

unless_certain e w d1 j1 d2 j2 = class
...
accept = before d2 action
...
where
  inW t at = (at>t-j1-j2) && (at<t+w+j1+j2)
inWOrL t at = at > t-d2-j1-j2

result Cep {...}
Related work


Not suited to embedded systems
Related work

Dan O’Keeffe, “Distributed Complex Event Detection for Pervasive Computing,” University of Cambridge, Tech. Rep., 2010
(Based on his recent PhD work/thesis)

Excellent overview of data acquisition and processing methods in sensor networks.

Proposes a language and implementation, no in depth comparison made yet
Future Work

• Implement and study of use-case
  – AESOP project, LKAB lubrication system

• Control in System of Systems
  – Further investigate CEP for control

• Patterns/Templates for CEP in Timber
  – Facilitate design of CEP queries

• Automatic generation of CRO/Timber CEP from domain specific language (e.g., CEDR)