University of Namur

Initiation à la démarche scientifique

IHDCB339

State of the Art: Complex Event Processing (CEP) for Internet of Things (IoT)

Author
Kenny Warszawski

Supervisor
Moussa Amrani
Pierre-Yves Schobbens

July 23, 2019

1 Introduction

The Internet of Things is omnipresent and the amount of connected devices is increasing day by day. The Internet of Things is different from a classic information system by its capability for integrating with the physical world. Those devices are used in plenty of sectors (health, industry, transport, home automation, ...) and their users can be both professionals and individuals. Nowadays, we can see the apparition of connected devices for home to facilitate our daily life. (e.g: Google Home, Nest, Philips Hue) IoT is also an interesting technology for Smart Cities use case. We can imagine a city where traffic lights are optimised with the city mobility to avoid traffic jams for example. However, an important problem arises in this type of architecture. How is it possible to handle such an important data traffic efficiently ? Indeed, if an entire city has a huge amount of connected objects, the data flow to be processed is massive. Therefore efficient data processing mechanisms need to be put in place to handle such flows.

Ajouter du texte, pas assez long

2 Context

2.1 IoT architecture

The architecture of an IoT solution can be implemented with various technologies and in very different infrastructures. However, a generic high level architecture is common to all IoT projects:

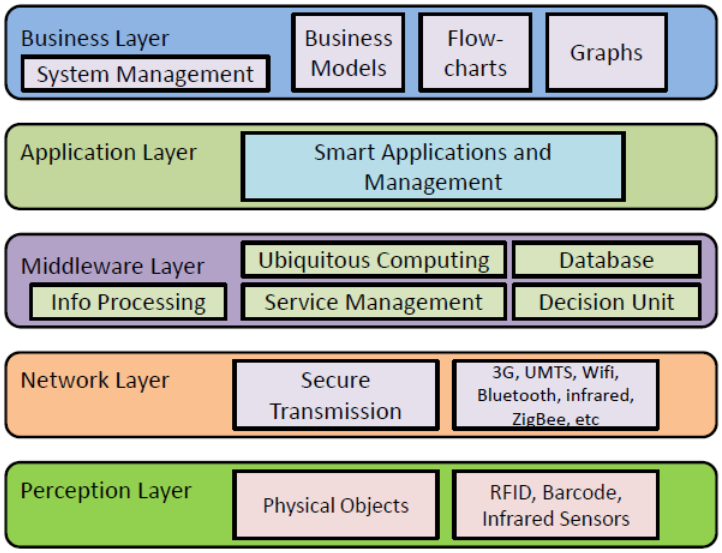

Figure 1: Generic Architecture

Perception Layer: The perception layer or device layer is the layer where physical objects will be able to capture data via sensors. These sensors are able to collect real world data. These data can be or not be processed upstream to add meta-data by the connected object itself and will be forwarded to the Network Layer.

Network Layer: The Network Layer or Transmission Layer. This layer transfers the informations from to Perception Layer to the Middleware Layer. This transfer can be accomplished by different communication technologies like Wifi, 3g, Bluetooth, etc.

Middleware Layer: The Middleware Layer is the common layer for all physical objects of an IoT architecture. The middleware is responsible to take automatic decisions based on informations coming for the devices and store the results in a database.

Application and Business Layer: The Application Layer is where

smart applications are running to produce complex processes. These applications will consume data from connected objects previously processed by the middleware to act effectively on a specific situation. These application can transmit actions to execute on a particular type of service. The Business Layer is important for the global management of the system. This one is rather a layer of global monitoring which produces graphs and statistics at the business level.

2.2 Data Flow

The previous architecture can be simplified to highlight the layer of architecture on which this state of the art will focus.

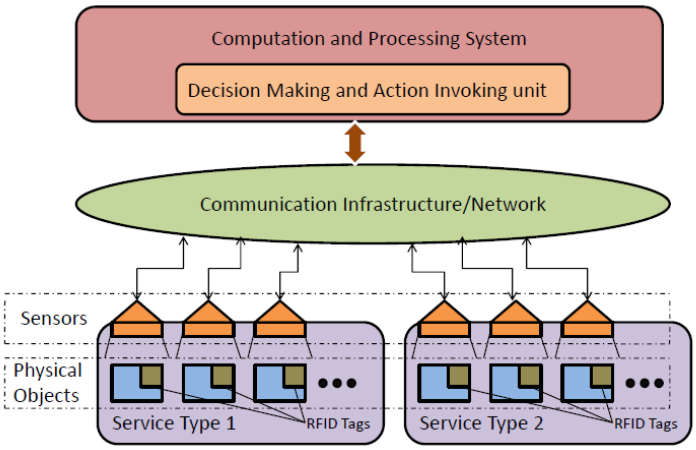

Figure 2: Simple IoT Architecture

This state of the art is focusing on the "Computation and Processing System" layer because at that stage, the data processing is the most critical and the data flow is the highest. Obviously, the Communication Layer must be robust enough to transmit the data efficiently but this topic will not be tackled in this article.

3 Complex Event Processing

3.1 Definition

Complex Event Processing (CEP) is a set of methods and techniques for tracking and analysing real-time streams of informations and detecting pat-

terns or correlations of unrelated data (complex events) that are of interest to a particular business. [**1**] The complex event processing is the engine of real-time applications that needs to have real-time informations from an environment to respond as fast as possible. A standard Request/Reply with synchronous processes cannot correlate different type of events and respond in a timely manner. Complex Event Processing mechanisms are often used in Event Driven architectures. This architecture fits with the Internet of Things needs. This kind of architecture is driven by events sent by connected objects. Then, the services at the application layer are consuming those events.

3.2 CEP Engine

The CEP engine is consuming the events sent by the physical objects on the Communication Layer. This engine can define a bunch of rules where each event received will be evaluated. A rule represents a complex condition that can be based on multiple events and on a time window. For example, if all connected cameras in a house are sending that there aren't any move in the house in a time window of 60 seconds, an event saying that nobody is in the house will be sent. Then, some specific actions can be taken by connected devices in the house.

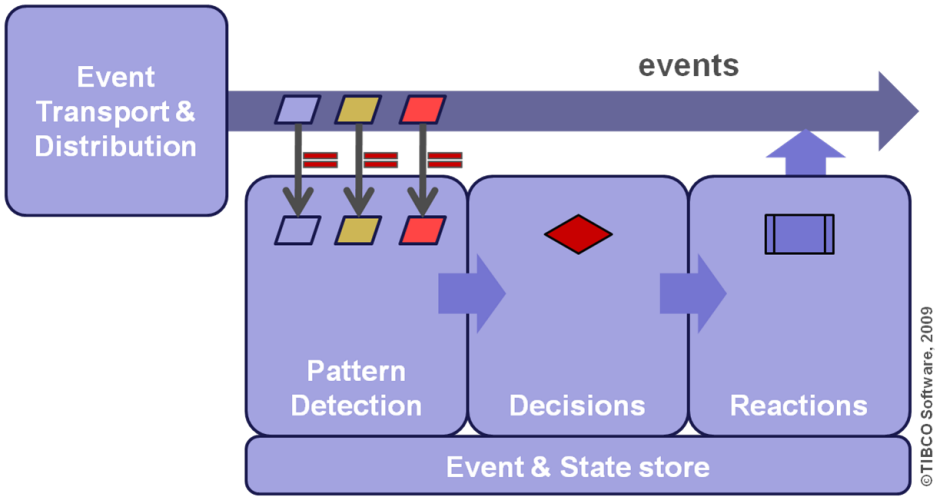

Figure 3: CEP Pattern Decision Reaction

The technologies implementing CEP concepts are querying Time Series Databases where each event received are stored. Thoses databases are opti-

mised for queries based on a time window. In a lot of CEP framework like Apache Flink, those CEP rules are describe like an SQL Query and uses Event Pattern Languages.

3.3 Event Pattern Languages

An Event Pattern Language (EPL) is a language used by CEP engines in order to describe the relations between events that match a specific pattern. The syntax of EPL's are quite similar to SQL queries. The Example 1 is retrieving dates where the temperature was bigger than 30 degrees Celsius.

Example 1:

```
SELECT Thermometer.date
FROM DataSource
WHERE temperature > 30
```

A plenty of primitive operators exists in Event Pattern Languages to describe the rule's behaviour. The following operators are coming from ThingML framework but are available in most of CEP frameworks:

Selection: Filters relevant events based on the values of their attributes.[**lu**] For example, we can select all Air Pressure events between 101300 Pa and 101400 Pa.
Projection: Extracts or transforms a subset of attributes of the events.[**lu**]
Window: Defines which portions of the input events to be considered for detecting pattern.[**lu**] For example, a rule can concern the last 30 seconds events.
Conjunction: Consider the occurrences of two or more events.[**lu**] This operator is basically a logical 'AND' operator.
Disjunction: Consider the occurrences of either one ore more events in a predefined set.[**lu**] This operator is basically a logical 'OR' operator.
Sequence: Introduces ordering relations among events of a pattern which is satisfied when all the events have been detected in the specified order.[**lu**]
Repetition: Considers a number of occurrences of a particular event.[**lu**]
Aggregation: Introduces constraints involving some aggregated attribute values.[**lu**] For example, the average temperature of all Meteo events is an aggregation.
Negation: Prescribes the absence of certain events.[**lu**] This operator is basically a logical 'NOT' operator.

List of Figures