# An Event Mesh for Event Driven IoT Applications

Roberto Berjón*, Montserrat Mateos, M. Encarnación Beato, Ana Fermoso García

Universidad Pontificia de Salamanca, Salamanca (Spain)

Received 21 April 2022 | Accepted 1 July 2022 | Early Access 19 September 2022

## Abstract

In IoT contexts, software solutions are required to have components located in different environments: mobile, edge, fog or cloud. To design this type of application, event driven architecture (EDA) is used to develop distributed, scalable, decoupled, desynchronized and real-time components. The interconnection between the different components is done through event brokers that allow communication based on messages (events). Although the design of the components is independent of the environment in which they are deployed, this environment can determine the infrastructure to be used, for example the event brokers, so it is common to have to make modifications to the applications to adapt them to these environments, which complicates their design and maintenance. It is therefore necessary to have an event mesh that allows the connection between event brokers to simplify the development of applications. This paper presents the SCIFI-II system, an event mesh that allows the distribution of events between event brokers. Its use will allow the design of components decoupling them from the event brokers, which will facilitate their deployment in any environment.

## Keywords

Cloud Computing, CloudEvents, Edge and Fog Environments, Event Driven Architecture, Event Mesh, Internet of Things.

DOI: 10.9781/ijimai.2022.09.003

## I. Introduction

CURRENTLY, all high performance IoT applications, regardless of their computing paradigm: Cloud, Edge, Fog [1], Edge mesh [2] and Cooperative-based systems [3], are developed from an event-driven architecture based on microservices. An IoT application based on microservices is structured through a collection of loosely coupled distributed components that facilitate the scalability and performance of the system. Moreover, the use of event driven architectures allows the real-time processing not only of a data stream coming from different data sources external to the application [4], but also of the data flow exchanged between the different microservices of the application.

There are two key elements to consider in these systems: how to represent the data to be processed and the communication channels through which to transport this data.

The data to be processed is represented in the form of an event. Through it, a series of other elements can be included that can be of great importance during its processing: its source, correlation, transactionality, etc. For this reason, the Cloud Native Computing Foundation (CNCF) promoted the CloudEvents specification [6] for describing event data regardless of the format (json, avro, protocol buffers, xaml) and protocol used for its transport (MQTT, AMQP, Kafka, HTTP, ...), thus guaranteeing its portability and interoperability. A CloudEvent contains two parts: data and metadata.

CloudEvent event data is the data represented through the event. It can be text or binary information. If it is text, the value is included in the "data" attribute of the cloud event. Conversely, if the data is binary, its Base64 encoded value is included in the "data_base64" attribute.

* Corresponding author.
E-mail address: rberjonga@upsa.es

CloudEvent event metadata provides contextual information. It is a set of mandatory and optional attributes in the form of a key value. Table I describes the attributes included in the specification. In addition, if necessary, applications can add new attributes.

TABLE I. CloudEvent Metadata Attributes

| Attribute | Description | Category |
|---|---|---|
| **source** | Represents the identifier of the publisher app that broadcast the event. It is expressed as a URI. | Required |
| **id** | Event identifier. The sender of the event must ensure that two events from the same source must necessarily have different values for this attribute. | Required |
| **type** | A publisher broadcasts different types of events. This attribute (which is a string) indicates the type of event. | Required |
| **subject** | Identifies the context in which the source emits the event. Usually, a consumer subscribes to events broadcast by a given source and subject. | Optional |
| **datacontenttype** | Represents the content type of data value. It is a string in RFC 2046 format. | Optional |
| **dataschema** | It is a URI that identifies the schema in which the data is structured. | Optional |
| **time** | Represents the datetime at which the publisher broadcasted the event. RFC 3339 encoded string. | Optional |

When integrating distributed systems, it is necessary to use event brokers through which data flows. Therefore, the components sending and receiving data are coupled with respect to the event broker used. As discussed above, one of the premises to be ensured when designing