



INAOE

Data alignment and association mechanism for multimedia data fusion in distributed sensor networks

por

José Roberto Pérez Cruz

Tesis sometida como requisito parcial para
obtener el grado de

**DOCTOR EN CIENCIAS EN EL ÁREA
DE CIENCIAS COMPUTACIONALES**

en el

**Instituto Nacional de Astrofísica, Óptica
y Electrónica**

Mayo 2014

Tonantzintla, Puebla

Supervisada por:

**Dr. Saúl Eduardo Pomares Hernández,
INAOE**

©INAOE 2014

El autor otorga al INAOE el permiso de
reproducir y distribuir copias en su totalidad o
en partes de esta tesis



Abstract

New applications based on wireless sensor networks (WSN), harvest a large amount of data streams that are simultaneously generated by multiple distributed sources. Specifically, in a WSN this paradigm of data generation/transmission is known as *event-streaming*. In order to be useful, all the collected data must be aligned so that it can be fused at a later phase. To perform such alignment, the sensors need to agree on common temporal references. Unfortunately, this agreement is difficult to achieve mainly due to the lack of perfectly-synchronized physical clocks and the asynchronous nature of the execution. Some solutions tackle the issue of the temporal alignment; however, they demand extra resources to the network deployment since they try to impose global references by using a centralized scheme. In this dissertation, we propose a distributed mechanism that performs at runtime the stream data alignment without requiring the use of synchronized clocks, additional signals or centralized schemes. This is achieved by translating temporal dependencies based on a time-line to causal dependencies among streams. In addition, we propose a spatio-temporal data association approach that extends the data alignment mechanism to include spatial information to perform the data alignment by considering the “closeness” among streams. To achieve this, the approach makes use of a fuzzy-causal relation defined to relate the space domain with the logical/temporal domain. For our case by establishing a fuzzy-causal relation we determine “how long ago” an event happened before another event.

Resumen

Nuevas aplicaciones basadas en redes de sensores inalámbricos (Wireless Sensor Networks (WSN)), cosechan una gran cantidad de flujos de datos, generados simultáneamente por múltiples fuentes distribuidas. Específicamente, en una WSN éste paradigma de generación y transmisión de datos es conocido como *event-streaming*. Con la finalidad de que sean útiles, todos los datos recolectados deben ser alineados para que puedan ser fusionados en etapas posteriores. Para realizar dicha alineación, los sensores necesitan acordar referencias temporales comunes. Desafortunadamente, dicho acuerdo es difícil de lograr principalmente debido a la falta de relojes físicos perfectamente sincronizados y a la naturaleza asíncrona de la ejecución. Algunas soluciones han atacado el problema de la alineación temporal, sin embargo, demandan recursos extras en el despliegue de la red debido a que tratan de imponer referencias globales usando esquemas centralizados. En esta tesis se propone un mecanismo distribuido que realiza la alineación de los flujos de datos en tiempo de ejecución, sin requerir del uso de relojes sincronizados, señales adicionales o esquemas centralizados. Esto se logra traduciendo las dependencias temporales, basadas en una línea de tiempo, a dependencias causales entre flujos. Además, se propone un enfoque para la alineación y asociación espacio-temporal de datos, que extiende el mecanismo de alineación, para incluir información espacial para realizar la alineación de datos considerando la “cercanía” entre flujos. Para lograr ésto, el enfoque usa una relación causal-difusa definida para relacionar el dominio del espacio con el dominio lógico/temporal. Para nuestro caso, estableciendo una relación causal-difusa, se puede determinar “cuanto tiempo atrás” un evento sucedió antes que otro evento.

Contents

Contents	vii
Notation	xi
1 Introduction	1
1.1 Motivation	1
1.2 Problem description	3
1.3 Proposed solution	4
1.4 The objectives of the dissertation	5
1.4.1 The main objective	5
1.4.2 Specific objectives	5
1.5 Document organization	6
2 Related work	9
2.1 Monitoring predicates and distributed events	9
2.1.1 Predicate detection using physical time	10
2.1.2 Predicates and distributed events detection using logical time	11
2.1.2.1 Centralized causality-based predicate detection	11
2.1.2.2 Centralized detection of distributed events	12
2.1.2.3 Distributed causality-based predicate detection	13
2.2 Temporal data alignment for sensor networks	14

3	Background and definitions	15
3.1	Distributed systems and the causal ordering	15
3.1.1	The logical mapping model	19
3.2	Fuzzy sets	20
3.2.1	Fuzzy inference system	24
4	The problem of the data alignment for event-streaming	27
4.1	The system model	27
4.2	Event-streaming abstract data type	28
4.2.1	Atomic event.	29
4.2.2	Local-streams.	29
4.2.3	Event-streamings.	30
4.3	Problem formulation	33
5	Temporal data alignment for event-streaming	35
5.1	The event-streaming logical mapping model (ES-LM)	35
5.1.1	Data alignment description	36
5.1.2	Formal proof of the ES-LM model	43
5.2	Temporal data alignment mechanism for event-streaming	46
5.2.1	Mechanism specification	47
5.2.2	Functional description of the data alignment mechanism	53
5.2.3	Correctness proof	58
5.2.4	Simulation results	60
5.2.4.1	Analysis of the simulation results	63
5.2.5	Analysis of the mechanism	64
6	Temporal data alignment and association for event-streaming	67

6.1	Spatio-temporal data association approach	67
6.1.1	The fuzzy-causal relation (FCR)	68
6.1.1.1	Fuzzification process	69
6.1.1.2	Fuzzy inference	71
6.1.1.3	Fuzzy Causal Relation formal definition	73
6.2	Temporal data alignment and association mechanism	74
6.2.1	Mechanism specification	74
6.2.2	Functional description of the mechanism	81
6.2.3	Discussion	84
7	Conclusions and future work	89
7.1	Conclusions	89
7.1.1	Summary	89
7.1.2	Achievements	90
7.1.3	Limitations	92
7.1.4	Dissertation-derived articles	93
7.2	Future work	93
	Bibliography	99

Notation

The notation used in this document is summarized in Table 1.

Table 1: General notation

Symbol	Meaning
$p_c, p_d, p_i, p_j, p_k, p_x$	processes involved to the WSN
m, x, y	messages exchanged by the processes
$x_1, \dots, y_1, \dots, z_1, \dots$	messages of any stream
\rightarrow	happened-before relation (HBR)
\downarrow	immediate dependency relation (IDR)
\leftarrow	assignation operator
S_i, S_j, X_c, Y_d, Y_k	local-streams
$X_c^-, X_c^+, Y_d^-, Y_d^+$	end points of local-streams
$ES_\Theta, ES_\vartheta, ES_\beta$	event-streamings
Θ, ϑ, β	sets of processes identifiers

Continued on next page

Table 1 – continued from previous page

Symbol	Meaning
$Q_a^{R_a}, Q_b^{R_b}, Q_q^{R_q}, Q_a^{R_a}$	subsets of events originated by different processes
$-Q_q^{R_q}, +Q_q^{R_q}$	endpoints of subset of events of an event-streaming
$\Omega_i, \Omega_j, \Omega_k, \Lambda_i, \Lambda_j, \Lambda_k$	Sets of events originated by the same process
x^-, y^-, x^+, y^+, e^*	endpoints of any stream
$\omega^-, \omega^+, \lambda^-, \lambda^+$	endpoints of sets of events originated by the same process

Chapter 1

Introduction

1.1 Motivation

The advances in MEMS¹ technologies as the progress in wireless communications, have allowed the development of low-cost, low-power sensors that are small in size and are able to communicate with other devices in short distances. The availability of this kind of hardware has motivated the development of wireless sensor networks (WSN). A WSN consists of several spatially distributed autonomous sensors deployed to extract information of the physical environment, such as temperature, humidity, pressure, etc. and to cooperatively pass their data through the network to a remote location [ASSC02].

Moreover, inexpensive emerging hardware such as CMOS² cameras and microphones, that are able to capture multimedia content, have expanded the capabilities of a WSN, enabling the collection of audio and video streams, besides still images and scalar data [AMC07].

WSNs have drawn the attention of research and industrial communities since they have allowed for the enhancement or for the visualization of new applications such as eHealth services, multimedia surveillance systems, person locator services, environmental/earth monitoring services, among others. These types of applica-

¹Micro-Electro-Mechanical Systems

²Complementary Metal-Oxide Semiconductor

tions ubiquitously³ harvest a large amount of data streams that contain numerous audio, video and other time-based data elements that are generated from several sensor nodes in a distributed way.

Specifically, the adopted paradigm to transmit data in a WSN is called *event-streaming*, which represents the generation and the transmission of data as a continuous stream of events reported by multiple sources [KS08, Ksh05, CK05, Rei09].

In addition to the event-streaming paradigm, due to the power and transmission constraints of the sensors, most of the WSNs are deployed under a *multi-hop* topology. Within a multi-hop topology, each node must not only capture and disseminate its own data streams, but also must collaborate to propagate the data in the network, by acting as a relay node.

For example, suppose that in a forest there is a WSN of different types of sensors, which aim to monitor different activities (see Figure 1.1). Each sensor communicates with a certain destination through two or more sensors. Individually, each relay sensor asynchronously generates a data stream in addition to retransmitting data. Thereby, all the collected streams generated by the multiple sensors will compose an event-streaming.

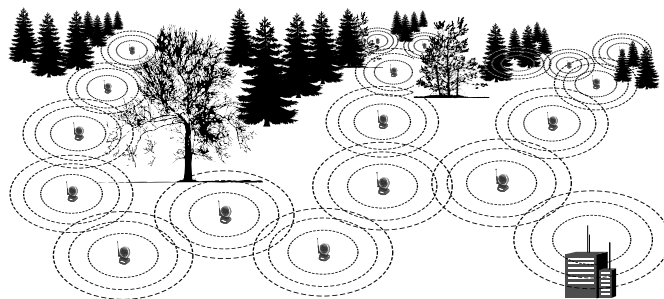


Figure 1.1: Scenario of a monitoring system in a forest.

³In this context, the term ubiquitous refers to the capacity to collect information from several places at the same time.

1.2 Problem description

By performing the data generation and transmission through the event-streaming paradigm along with the multi-hop communication, a considerable amount of meaningless data are exchanged through the nodes of a WSN. In order to make sense of the collected data and produce useful information, all the streams must be fused⁴ in later stages. One main task during the data fusion is the data alignment, which consists in processing the collected data to attain a common spatial and time frame [ESW⁺05]. To achieve this, the sensors in the network may need to agree on some common temporal references of the whole system [BPLOBdlC11, CK08]. However, achieve common references in a WSN is difficult since they must be obtained from multiple asynchronous views.

Some approaches, to establish common temporal references, propose the use of techniques based on the Allen's Interval Algebra [All83] by detecting the temporal dependencies among the exchanged streams [CK08, Ksh05, GYJO10]. In such approaches, by using a common physical time axis, the temporal dependencies are specified by determining the order in which the streams occur or when they co-occur. Unfortunately, the characteristics of a WSN make difficult to establish a single physical time axis, this is mainly due to the absence of shared memory and the lack of perfectly synchronized physical clocks [CK05, KC13].

In order to avoid the use of a single physical time axis a clock-free alignment approach for data streams was proposed in [LS07]. This solution assumes a communication scheme where some sensor nodes act as intermediate nodes to aggregate or to collect the data streams which are later sent to another sensor or sink. Assuming the previous communication scheme, the approach has shown that

⁴Data fusion refers to the alignment, association, correlation, filtration and aggregation of the collected data [HM04, BGM08].

aligning the data streams on the intermediate nodes without synchronizing the clocks of all sensors, is sufficient. Nevertheless, this solution requires a *synchronization server* that broadcasts synchronization signals to the sensors to establish a global reference, and it also needs dedicated devices (*data servers*) to align the streams according to the broadcasted signals. These two additional requirements imply extra network resources.

1.3 Proposed solution

In this dissertation we propose as main contribution, a distributed data alignment and association mechanism for event-streaming in WSNs. Based on causal dependencies, the mechanism is able to establish temporal references among the exchanged data streams, without requiring the use of synchronized clocks or centralized schemes.

For our proposal, we define the event-streaming as an abstract data type (ADT). Such ADT allows us to model the event-streaming as a sequential arrangement of subsets of events generated by multiple sources.

Based on the event-streaming ADT, we propose a new model for data alignment that we have called Event-Streaming Logical Mapping (ES-LM). The ES-LM performs the data alignment by translating the time-line based dependencies among streams to causal dependencies. Such translation allows us to construct a virtual time-line where each event is positioned into a specific time-slot. By using the resulting virtual time-line we can avoid: 1) the use of synchronized clocks, 2) the use of global references, 3) the use of centralized schemes, and 4) the use of additional synchronization signals.

In addition to the ES-LM, we propose a spatio-temporal data association approach. This approach extends the ES-LM to include spatial information to perform the data alignment by considering the “causal closeness” among streams. To achieve this, the approach makes use of a fuzzy-causal relation defined to relate the space domain with the logical/temporal domain. For our case by establishing a fuzzy-causal relation we determine how “close” an event happened before another event.

Taking advantage of the retransmissions of a multi-hop communication, the proposed mechanism uses the intermediate nodes to implement the ES-LM and the spatio-temporal data association approach. In this way, the data alignment and association were performed while the multiple streams are propagated through the network.

1.4 The objectives of the dissertation

1.4.1 The main objective

The main objective of our research work is:

To design and to develop a distributed mechanism for the event-streaming data alignment and association that serves as support for the data fusion process and satisfies the characteristics and restrictions of a WSN.

1.4.2 Specific objectives

1. Model the event-streaming as an abstract data type, in order to achieve a way to handle this paradigm.
2. Establish causal order principles to handle event-streaming.

3. Design algorithms, based on the causal order principles, to detect temporal references from the event-streaming.
4. Establish fuzzy causal principles to handle event-streaming.
5. Design algorithms, based on the fuzzy causal principles, to detect spatio-temporal references from the event-streaming.

1.5 Document organization

This document is organized as follows. In chapter 2 some works related to the detection of temporal patterns from event streams are presented.

In chapter 3 the main concepts about distributed systems, causal ordering and fuzzy sets are introduced.

In chapter 4, the event-streaming is defined as an abstract data type. Based on such a definition, the problem of the data alignment for event-streaming is formally stated.

In chapter 5, the event-streaming logical mapping (ES-LM) model is presented. Firstly, it is explained how data alignment is performed by using the ES-LM. Secondly, based on the ES-LM, the data alignment mechanism for event-streaming in WSN is detailed. The effectiveness of the data alignment mechanism is verified by a correctness proof. Complementarily, the viability of the mechanism is shown through a simulation over a sensor network with multi-hop communication.

In chapter 6, the fuzzy-causal relation (FCR) is defined. Based on the FCR, a spatio-temporal data association for event-streaming approach is presented. In addition, a mechanism that implements the ES-LM along with the spatio-temporal data association approach is detailed. In this chapter also it is explained how the

establishment of a closeness degree among events, can be useful to refine the data alignment. The effectiveness of the resulting mechanism is verified by simulations.

Finally, the conclusions and the future work of this research are summarized in chapter 7.

Chapter 2

Related work

Data alignment and association in WSN are related to the detection of a common time frame by processing several data streams generated and transmitted by multiple distributed sources. Specifically, the adopted paradigm to exchange data in a WSN is called *event-streaming*. The *event-streaming* paradigm has also been employed in different distributed computing environments to detect global predicates¹ and monitor distributed events among multiple processes².

In this chapter we present some of the proposed solutions related to the problem of attain common time references, by processing several streams exchanged through the *event-streaming*. In a first part we briefly discussed some approaches designed for the monitoring of predicates and distributed events in ubiquitous environments. Then, in a second part we present a novel approach for the data streams alignment for sensor networks.

2.1 Monitoring predicates and distributed events

Predicates defined on program variables, that are local to different processes, are used for specifying conditions on the global system state. Predicates are useful for

¹A predicate is a function on a set of terms or variables, so that by satisfying certain conditions, returns a boolean value.

²In this sense a process is a program or instances of program

applications such as debugging, environmental sensing, and in industrial process control. An important paradigm for monitoring distributed events is that of *event-streaming*, wherein streams of relevant events reported from different processes are examined collectively to detect predicates. Typically, the specification of such predicates uses physical or logical time relationships [KS08].

2.1.1 Predicate detection using physical time

Kshemkalyani has proposed algorithms to detect predicates from event streams that are reported by the various components of an ubiquitous computing environment [Ksh05, Ksh07]. These algorithms are based on a system model which assumes a wireless multi-hop communication system, where each entity is able to send its gathered data, eventually and asynchronously, in a FIFO stream to a powerful data fusion server. By considering such a system model, all the devices in the ubiquitous network must timestamp the transmitted streams by using synchronized physical clocks based on a global time axis.

Based on the global time axis, the fusion server can detect a global predicate from the event streams that occur at the same time instant at each process. For this, the fusion server maintains many queues as processes are involved in the system to store the transmitted streams and iteratively detect concurrent pairwise interactions for each pair of processes. Thus, the concurrent pairwise interactions are specified through the temporal relationships among streams, based on the thirteen interval-interval relations proposed by Allen [All83].

A major shortcoming of such proposals is that in many ubiquitous systems, instantaneous observation across various locations is not possible due to the lack of perfectly synchronized clocks. This fact make it difficult to achieve a global time axis. Furthermore, the asynchronous nature of several distributed systems, caused

by unpredictable propagation delays and CPU loads, leads to many interleavings of the events and different observations of their order [CK05, KC13].

2.1.2 Predicates and distributed events detection using logical time

2.1.2.1 Centralized causality-based predicate detection

Based on the system model defined by Kshemkalyani in [Ksh05, Ksh07] (see previous section), Chandra and Kshemkalyani [CK05] propose the use of the happen before relation [Lam78] to detect global predicates in ubiquitous computing environments without using synchronized physical clocks.

To achieve this, they propose the use the well-known vector clock [Mat88, Fid88] structure to provide logical time in the system. Thus, each process maintains a vector clock, of size n (where n is the number of processes in the whole system), to manage the timestamps of each exchanged event. In addition, each process maintains an *interval clock*, also of size n , to store and to manage the timestamps for the endpoints of each transmitted event stream (interval).

Based on the logical time provided by the vector clocks, in a similar way that the approach of Kshemkalyani [Ksh05, Ksh07], a fusion server can detect a predicate by analyzing iteratively each pair of streams, stored in n queues. The concurrent pairwise interactions, necessary to detect a global predicate, are specified by using the 29 interval-interval orthogonal relations proposed by Kshemkalyani [Ksh96].

The main weakness of this approach is that it requires a central server. This implies that a single entity needs to analyze, pair by pair, many streams as processes are involved in the system. According to Chandra and Kshemkalyani [CK08], un-

der such scheme, the workload and space complexity increase linearly with the number of processes, besides that the network traffic creates a bottleneck at the server's links, restricting the system's scalability.

2.1.2.2 Centralized detection of distributed events

Gao et al. in [GYJO10] propose an approach to detect real life events from several event streams reported by the different sensors of a multimedia communication system.

This approach defines a system model based on a concept of atomic event, which is an abstraction to represent actions that occur at certain point of a 2D plane and have a duration. In such a system model, each event is stamped with two *time points*: a start time and an end time, where each time point is defined as an integer in \mathbb{N} . In addition, each atomic event is distinguished from another by its event type. The event type is a descriptor of an interaction such as: “*went from*”, “*walking to*”, “*catching an*”, etc. Thereby, by specifying temporal relationships among atomic events, more meaningful events, called *composite events*, can be produced. These *composite events* are used to describe and recognize complex events such as happenings that involves several people and objects.

To detect the temporal relationships among the atomic events, that originate a composite event, this approach proposes the use of an event processor. Such an event processor is a computational entity where all the collected composite events are pre-registered as queries. Assuming that the events arrive in order of end time, the event processor orders the collected streams according to their sequential and concurrent relationships, which are specified using five basic temporal patterns based on the interval algebra defined by Allen [All83].

Besides the typical problems related to the centralized schemes, this approach does not consider the case when the events arrive, at the event processor, disordered with respect to the end time. Due to the asynchronous nature of most of the distributed systems, an ordered arrival of events is not always possible.

2.1.2.3 Distributed causality-based predicate detection

Chandra and Kshemkalyani in [CK08], proposed a distributed extension of the causality-based predicate detection approach proposed in [CK05]. In such an extension, the predicate detection algorithms are executed by the n processes of the system. Thus, the space complexity and the time complexity get distributed linearly with n , besides that the formation of bottlenecks is avoided.

This approach is based on two token-based algorithms, where the processes involved in the system use a token to appoint the entity who is enabled to execute the predicate detection at certain time. The token is formed by two vectors: a vector to store and to manage the timestamps for the endpoints of the event stream transmitted by the n processes, and a boolean vector to indicate the processes, who generated the event streams, have been detected as a part of the current predicate. Thus, individually each process analyzes the temporal relationships between its own generated stream and the streams described by the vectors of the token. In this way, each process only needs to store one single queue to store its own stream, unlike the centralized scheme, where the fusion server needs to store n queues.

The main problem of the token-based algorithms is that to determine temporal references, common to the whole system, it is necessary that the token pass through all the n processes. Thus, a process i ($i < n$), after detecting the temporal relationships between its own stream and the streams described in the

token, needs to wait $n - 1$ additional turns to achieve the global system view. Furthermore, the communication overhead for the control information exchange, is duplicated with respect to the centralized scheme.

2.2 Temporal data alignment for sensor networks

Lee and Shih proposed a clock free streams alignment mechanism for sensor networks [LS07]. Such a mechanism allows to align the sensor data streams without using synchronized physical clocks.

This solution proposes a system model where a sensor network is formed by several sensor groups. A sensor group consists of at most n sensors and a collecting device called *data server*. The *data servers* act as intermediate nodes to aggregate or to collect the data streams which are later sent to another node or to a sink. Furthermore, they propose the use of a *synchronization server* that broadcasts synchronization signals to the whole network to establish a global reference. Thus, a data server aligns the data streams, received from a sensor group, according to the broadcasted synchronization signals.

Although the use of the data servers increases the scalability of the system, with respect to the number of associated nodes in the network, demands extra network requirements. Furthermore, since all the *data servers* use the synchronization signals broadcasted by a single *synchronization server*, if such a server fails, the synchrony of the whole system can be affected.

Chapter 3

Background and definitions

3.1 Distributed systems and the causal ordering

Time is an important theoretical construction to understand how the transactions are executed in certain systems. A practical way to achieve such a construction, is by recording the time when certain events happen to create a temporal ordering of events. Unfortunately, there are some environments where the lack of global references, such as physical clocks perfectly synchronized, make difficult to establish a temporal ordering. An example of such kind of environments is a distributed system.

A distributed system is composed by different entities spatially separated, that communicate with each other by exchanging messages. In a distributed system each entity has its own physical clock that has a certain gap with the others. In the absence of a global physical time, in a distributed system many times is impossible to determine if an event has happened before other, in other words, is impossible to determine the system's causal order.

A causal order establishes a precedence relation between two events in the following way: let e_1 and e_2 be two events causally related, it is said that e_1

happened before $e2$ if there is an information flow¹ from $e1$ to $e2$, and given such a relation, $e1$ must be processed before $e2$.

A typical distributed system is described by a system model composed by the following elements:

- **Processes.** Programs or instances of programs running simultaneously with other programs. Each process belongs to the set $P = \{p_i, p_j, \dots\}$.
- **Messages.** Abstractions to represent frames or packets in a computer network, which can contain arbitrarily complex data structures. In a typical distributed system a process can only communicate with other processes by message passing over a communication network. Each exchanged message in the system belongs to the set M .
- **Events.** An event represents an instant execution performed by a process. In a distributed system, a process can only execute two kind of events: *internal events* and *external events*. An internal event is an action in a process that affects only the process at which it occurs. An external event is also an action in a process, but unlike an internal event, this action is seen by other processes affecting the global system state. For communication interactions there are three types of external process: *send*, *receive* and *delivery*.
 1. The send event refers to the emission of a message, executed by a process.
 2. The receive event refers to the notification on the arrival of a message in a process.

¹Information flow in an information theoretical context is the transfer of information from an entity x to an entity y .

3. The delivery event refers to the execution performed by a process to present the received information to an application or other process.

Definition 1. – The happened-before relation (HBR). *The HBR [Lam78], “ \rightarrow ”, is the smallest relation on a set of events E satisfying the following conditions:*

1. *If a and b are events belonging to the same process, and a was originated before b , then $a \rightarrow b$.*
2. *If a is the sending of a message by one process and b is the receipt of the same message by another process, then $a \rightarrow b$.*
3. *If $a \rightarrow b$ and $b \rightarrow c$, then $a \rightarrow c$.*

Based on Definition 1, Lamport defines that a pair of events is concurrently related “ $a \parallel b$ ” as follows:

Definition 2. *Two events, a and b , are said to be concurrent if $a \not\rightarrow b$ and $b \not\rightarrow a$, which is denoted by $a \parallel b$ [Lam78].*

Definition 3. – Immediate dependency relation (IDR). *The IDR [PFD04] is the transitive reduction of the HBR and is defined as follows:*

Two events $a, b \in E$ have an immediate dependency relation (denoted by “ \downarrow ”) if:

$$a \downarrow b \text{ if } a \rightarrow b \wedge \forall c \in E, \neg(a \rightarrow c \rightarrow b)$$

Definition 4. – Precedence relation of messages. *Let two messages $m, m' \in M$, the precedence relation of messages denoted by $m \rightarrow m'$ is induced by the precedence relation of events, and it is defined by:*

$$m \rightarrow m' \text{ iff } \text{send}(m) \rightarrow \text{send}(m')$$

Definition 5. – Intervals. *An interval is a set of messages sent by a process p_i during a period of time. If the messages that compose an interval satisfy a certain order, then such interval is called ordered interval.*

The works of Shimamura et al. [STT01] and Pomares et al. [PEMR08] define the following ordered interval composition.

Definition 6. *Let X be an interval of sequentially-ordered messages at a process p_i ; $X \subset E$, and $x^-, e, x^+ \in X$; where x^- is the left endpoint and x^+ is the right endpoint of interval X , such that $\forall e \in X, x^- \rightarrow_i e \rightarrow_i x^+$ and $x^- \neq e, x^+ \neq e$.*

When $|X| = 1$, this implies that $x^- = x^+$; in this case, x^- and x^+ are denoted indistinctly by x .

Definition 7. – Happened-before relation for intervals. *Lamport establishes in [Lam86] that an interval A happens before another interval B if all the elements that compose an interval A causally precede all the elements of interval B .*

The causal relation “ \rightarrow ” is established at a set level by satisfying the following conditions:

1. $A \rightarrow B$ if $a \rightarrow b, \forall (a, b) \in A \times B$,
2. $A \rightarrow B$ if $\exists C \mid (A \rightarrow C \wedge C \rightarrow B)$.

According to Definition 6 and Pomares et al. [PEMR08], the happened-before relation in regard to ordered intervals can be expressed only in terms of the endpoints as follows:

Property 1. *Let A, B and C be ordered intervals. A occurs before B if any of the following conditions are satisfied:*

1. $A \rightarrow B$ if $a^+ \rightarrow b^-$
2. $A \rightarrow B$ if $\exists C$, such that $a^+ \rightarrow c^- \wedge c^+ \rightarrow b^-$

Definition 8. Let A and B be two ordered intervals. A and B are said to be simultaneous (denoted by $A|||B$) if the following condition is satisfied [PEMR08]:

$$A|||B \text{ if } a^- || b^- \wedge a^+ || b^+$$

The definition above means that one interval A can take place at the “same time” as another interval B .

3.1.1 The logical mapping model

The logical mapping model introduced in [PEMR08] is useful to represent pairwise interactions between processes. Such a model expresses temporal relations between sets of events in terms of the happened before relation for intervals.

The logical mapping model is intended to be the support in the generation of synchronization specifications for a temporal scenario. Thus, with the logical mapping translation, once a pair of intervals X and Y are identified, they are segmented into four subintervals: $A(X, Y)$, $C(X, Y)$, $D(X, Y)$ and $B(X, Y)$, to construct the general causal structure: $S(X, Y) = A(X, Y) \rightarrow_I W(X, Y) \rightarrow_I B(X, Y)$, where $W(X, Y)$ determines if overlaps exist between the pair of intervals (see Table 2).

Based on the intervals’ segmentation, the logical mapping model identifies five logical mappings which are: *precedes*, *simultaneous*, *ends*, *starts* and *overlaps*. These five logical mappings are sufficient to represent all possible temporal relations between continuous media (interval-interval relations [All83]), discrete media (point-to-point relations), and discrete-continuous media relations [Mar82].

Table 2: Logical mapping

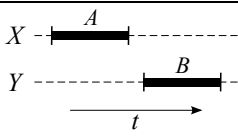
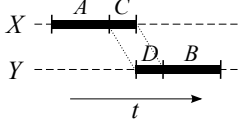
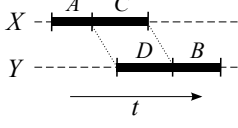
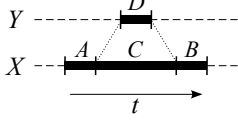
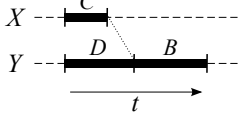
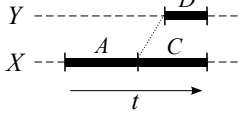
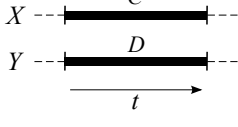
$\forall(X, Y)$	\in	$I \times I$
$A(X, Y)$	\leftarrow	<ul style="list-style-type: none"> • if $x^- \rightarrow y^-$, $\{x \in X : delivery(Part(Y), x) \rightarrow send(y^-)\}$ • otherwise, \emptyset
$C(X, Y)$	\leftarrow	<ul style="list-style-type: none"> • if $y^+ \rightarrow x^+$, $\{x \in X : send(x) \rightarrow delivery(Part(X), y^+)\} - A(X, Y)$ • otherwise, $X - A(X, Y)$
$D(X, Y)$	\leftarrow	<ul style="list-style-type: none"> • if $x^+ \rightarrow y^+$, $Y - \{y \in Y : delivery(Part(Y), x^+) \rightarrow send(y)\}$ • otherwise, Y
$B(X, Y)$	\leftarrow	<ul style="list-style-type: none"> • if $y^+ \rightarrow x^+$, $X - \{A(X, Y) \cup C(X, Y)\}$ • otherwise, $Y - D(X, Y)$
$W(X, Y)$	\equiv	$C(X, Y) D(X, Y)$
$S(X, Y)$	\equiv	$A(X, Y) \rightarrow_I W(X, Y) \rightarrow_I B(X, Y)$

Using the the intervals' segmentation and the identification of the five logical mappings, some synchronization points are detected. With such synchronization points can be determined which parts of the intervals are simultaneous and can be presented in any order, one respect to each other, and determine which parts need to be presented in a causal order.

3.2 Fuzzy sets

Definition 9. Fuzzy sets. Zadeh establishes in [Zad65] that a fuzzy set A is defined as a membership function $f_A(x)$ that maps the elements of a domain or universe X with the elements of the interval $[0, 1]$: $f_A : X \rightarrow [0, 1]$, representing the degree of membership of x in A . The closer the value of $f_A(x)$ to 1, the higher the degree of membership of x in A .

Table 3: Interval-interval relations and their logical mappings

Interval-interval relation	Logical mapping	Logical mapping expressed by endpoints	Scenario example
X precedes Y	$precedes : A \rightarrow_I B$	$a^+ \rightarrow b^-$	
X meets Y	$A \rightarrow_I (C D) \rightarrow_I B$	$a^+ \rightarrow c^-, a^+ \rightarrow d^-$	
X overlaps Y		$c^- d^-, c^+ d^+$	
X contains Y		$c^+ \rightarrow b^-, d^+ \rightarrow b^-$	
X starts Y	$starts : (C D) \rightarrow_I B$	$c^- d^-, c^+ d^+, c^+ \rightarrow b^-, d^+ \rightarrow b^-$	
X finishes Y	$ends : A \rightarrow_I (C D)$	$a^+ \rightarrow c^-, a^+ \rightarrow d^-, c^- d^-, c^+ d^+$	
X equals Y	$simultaneous : C D$	$c^- d^-, c^+ d^+$	

A fuzzy set A can be represented as a set of pairs of values: each element $x \in X$ with its degree of membership in A .

$$A = \{(x, f_A(x)) | x \in X\}$$

Definition 10. Fuzzification is the conversion of a precise quantity to a fuzzy quantity.

Generally, the fuzzification of a real value is performed using intuition, experience and an analysis of the set of conditions associated to the input variables. The most used fuzzifiers, are the based on triangular and trapezoidal functions:

- **Triangular function:** $f_A(x) = \max[\min(\frac{x-L}{C-L}, \frac{R-x}{R-C}), 0]$, (see Figure 3.1)

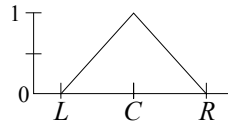


Figure 3.1: Triangular function

where L , C and R are the real scalar values that delimit a fuzzy set A , being C the input value that has the largest membership to A .

- **Trapezoidal function:** $f_A(x) = \max[\min(\frac{x-L}{b-L}, 1, \frac{U-x}{U-c}), 0]$, (see Figure 3.2)

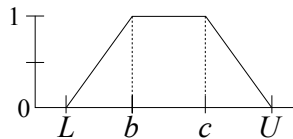


Figure 3.2: Trapezoidal function

where L , U , b and c are the real scalar values that delimit a fuzzy set A , being b and c the boundaries of the range where the inputs have the largest membership to A .

Definition 11. Defuzzification is the conversion of a fuzzy quantity to a precise quantity.

Defuzzification can be performed in several ways; however, the most used defuzzification methods are the based on the called center of area or center of gravity.

Centroid of Area (COA) method. This procedure is the most prevalent and physically appealing of all the defuzzification methods [Sug85, Lee90]; it is given by the algebraic expression

$$COA = \frac{\int f_A(a_c) \cdot a_c da_c}{\int f_A(a_c) da_c},$$

where \int denotes an algebraic integration.

Weighted average (WA) method. The weighted average method is the most frequently used in fuzzy applications since it is one of the more computationally efficient methods [Ros95]. The only restriction is that the output membership functions must be symmetrical. It is given by the algebraic expression

$$WA = \frac{\sum f_A(\bar{a}_c) \cdot \bar{a}_c}{\sum f_A(\bar{a}_c)},$$

where \sum denotes the algebraic sum and where \bar{a}_c is the centroid of each symmetric membership function, for example C (see Figure 3.1). The weighted average method is formed by weighting each membership function in the output by its respective maximum membership value.

Definition 12. Linguistic variables. *The linguistic variables are variables whose values are represented using linguistic terms (low, medium, high, very high, etc.). The meaning of these terms is determined through fuzzy sets. A linguistic variable is characterized by (v, T, X, g, m) , where:*

- v is the name of the variable

- T is the set of linguistic terms of v
- X is the universe of discourse of the variable v
- g is a syntactic rule to generate linguistic terms
- m is a syntactic rule that assigns to each linguistic term t its own meaning $m(t)$, which is a fuzzy set in X

3.2.1 Fuzzy inference system

A fuzzy inference system (FIS) is a way to transform an input space in an output space, using fuzzy logic. The FIS attempts to formalize, using the fuzzy logic, reasoning of human language.

Generally, a FIS has four modules as depicted in Figure 3.3:

- *Fuzzification module*: transforms the system inputs, which are crisp numbers, into memberships to fuzzy sets. This is done by applying a fuzzification function.
- *Knowledge base*: stores *if-then* rules provided by experts.
- *Inference engine*: simulates the human reasoning process by making fuzzy inference on the inputs and *if-then* rules.
- *Defuzzification module*: transforms the memberships to fuzzy sets, obtained by the inference engine, into a crisp value.

The most used FIS are the Mamdani type [MA75] and the Sugeno type [TS85].

- In the Mamdani systems the inputs and the outputs of the inference engine are fuzzy

If x is A and y is B then z is V

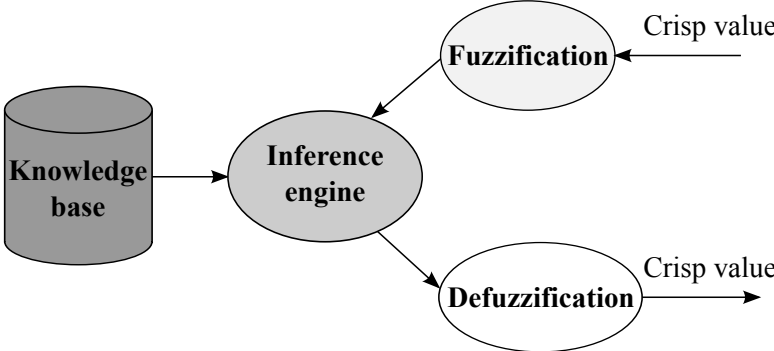


Figure 3.3: Typical FIS modules.

- In the Sugeno systems, the inputs of the inference engine are fuzzy and the output is “crisp”

$$\text{If } x \text{ is } A \text{ and } y \text{ is } B \text{ then } z = f(x, y)$$

Chapter 4

The problem of the data alignment for event-streaming

Before formally defining the problem of the data alignment for event-streaming, we introduce our system model. Then, we define the event-streaming as an abstract data type based on which we formulate the problem of the data alignment for event-streaming.

4.1 The system model

We specify a WSN as a distributed system, consisting of three main components:

- **Processes.** Each entity associated to the WSN (sensors or sink) is represented as an individual process. Each process belongs to the set $P = \{p_i, p_j, \dots\}$ and communicates with other processes by message passing over a multi-hop communication, sending only one message at a time.
- **Messages.** The messages sent by the processes in the WSN belong to the set M . Each message contains a sample of audio, video or many other physical signals that each device collects. Henceforth, we will refer to a sample as a message. For our approach, each message $m \in M$ is a tuple (sts, t_{sts}) , where:

- sts is the identifier of the process that originally generated the message.
 - t_{sts} is the logical sample time, at which a process p_{sts} conducts such a sample data.
- **Events.** We consider three kinds of events:
 1. *send* refers to the emission of a message executed by a process.
 2. *receive* refers to the notification upon the arrival of a message in a process.
 3. *delivery* refers to the execution performed by a process to present the received information to an application or another process.

Let m be a message. We denote by $send(m)$ the emission event, by $receive(m)$ the reception, and by $delivery(p, m)$ the delivery of m to process p .

Furthermore, we consider two main characteristics for the transmissions in a WSN:

- **Transmission delay.** For a message $m \in M$ there is a time period for the medium accessing and the network propagation.
- **Synchronization error of two samples.** This refers to the difference between the local time reference assigned at the reception, which can be used to estimate the sample time, and the sample time of the source.

4.2 Event-streaming abstract data type

We propose a definition of the event-streaming oriented to the data alignment problem. Assuming that an event-streaming is composed of several events, we begin by defining the concept of an atomic event.

4.2.1 Atomic event.

An atomic event indicates that an entity has sent or delivered a message containing a sample. In other words, an atomic event indicates that some data has been collected from the environment.

Definition 13. *An atomic event is a tuple $e(p_j, m(p_i, x))$, where p_j refers to the process where the event is executed and $m(p_i, x)$ is a message ($m \in M$) that has the process p_i as origin.*

As we stated above, we consider three types of events: send, receive and delivery. We denote the atomic delivery event by $delivery(p_j, m(p_i, x))$, the atomic receive event by $receive(p_j, m(p_i, x))$, and we denote the atomic send event only by $send(m(p_i, x))$ since $p_i = p_j$.

Based on the concept of atomic event, for our solution we distinguish two kinds of data streams generated by the nodes in a WSN: the *local-streams* and the *event-streamings*.

4.2.2 Local-streams.

In a WSN, each process generates a certain number of atomic events throughout the communication process. When some of these events are generated sequentially by a process p_i during a period of time, we say that the process $p_i \in P$ has generated a *local-stream*. We formally define a local-stream as follows:

Definition 14. *A local-stream is a poset (S_i, \rightarrow_i) where S_i is a finite set of atomic events $S_i = \{e_1, e_2, \dots, e_n\}$ generated by the process $p_i \in P$ and arranged according to the local causal relation \rightarrow_i .*

A local-stream can be expressed by its endpoints, similarly to the intervals (see Definition 4). For a local-stream $S_i = \{e_1, e_2, \dots, e_n\}$, the endpoints are $S_i^- = e_1$

and $S_i^+ = e_n$. The endpoint S_i^- refers to the beginning of the local-stream, while S_i^+ refers to its end.

For example, suppose that a process p_k sequentially generates 13 events as depicted in Figure 4.1.

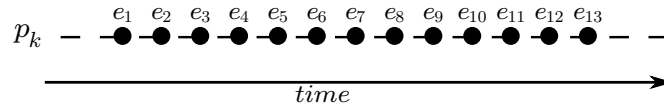


Figure 4.1: Representation of a local-stream generated by a process p_k .

Since all these events were generated one after another by p_k during a period of time, such events are related according to the local happened-before relation \rightarrow_k forming the causal structure:

$$S_k = e_1 \rightarrow_k e_2 \rightarrow_k \cdots \rightarrow_k e_{12} \rightarrow_k e_{13}.$$

In this way, we say that S_k is a local-stream that can be expressed with the endpoints $S_k^- = e_1$ and $S_k^+ = e_{13}$.

4.2.3 Event-streamings.

An event-streaming can be viewed as a collection of local-streams gathered by a process p_j . Any process p_j gathers several local-streams when it relays streams generated by other processes in order to help the data reach its final destination.

Since all the local-streams exchanged within a WSN are asynchronously generated by multiple sources, according to the local view of a process p_j , some segments (subsets) of the gathered local-streams will be simultaneous (generated in the same time slot) while other segments will not. To distinguish the simultaneous segments from the non simultaneous segments, they are grouped into different sets of events.

For example, suppose that two processes p_i and p_k generate two local-streams S_i and S_k , respectively, and that a process p_j is a relay node that helps p_i and p_k propagate their streams along the network. Also suppose that p_j generates the local-stream S_j while relays S_i and S_k . According to the local view of the system's execution at p_j , segments of S_i , S_k and S_j can be simultaneous at different instants, as depicted in Figure 4.2.

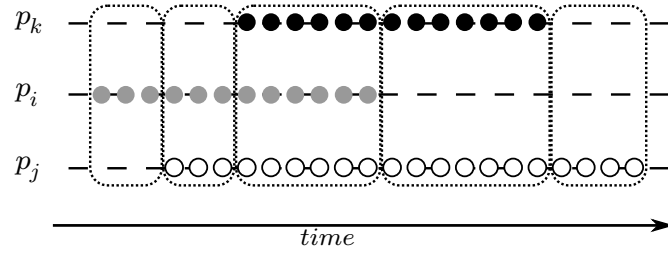


Figure 4.2: Representation of the subsets of an event-streaming.

In this example we can distinguish five different sets (from left to right): a first set containing events exclusively generated by p_i , a second set containing events generated by p_i and p_j , a third set containing events generated by p_i , p_j and p_k , a fourth set containing events generated by p_j and p_k , and a fifth set containing events exclusively generated by p_j .

Formally, let R_q be a set of processes' identifiers. We denote by $Q_q^{R_q}$ the set of events generated by the processes whose identifiers are in R_q . Thus, each set $Q_q^{R_q}$ is a collection of subsets of local-streams $\Omega_i, \Omega_j, \dots, \Omega_k$, where $\Omega_i \subseteq S_i$, $\Omega_j \subseteq S_j$ and $\Omega_k \subseteq S_k$ such that $\Omega_i || \Omega_j || \dots || \Omega_k$, ($i, j, \dots, k \in R_q$). Furthermore, the events in any set Ω_ζ are arranged according to the local causal relation \rightarrow_ζ , which form the poset $(\Omega_\zeta, \rightarrow_\zeta)$.

According to the local view of a process p_j , all the sets $Q_q^{R_q}$ are formed one after another according to the system's execution. Therefore, when a process p_j

causally arranges different sets $Q_q^{R_q}$, one after another, we say that p_j generates an event-streaming.

Let Θ be a set of the processes' identifiers that generates all the local-streams gathered by a process p_j . We denote by ES_Θ the event-streaming formed by the general causal structure:

$$ES_\Theta = Q_1^{R_1} \rightarrow Q_2^{R_2} \rightarrow \dots \rightarrow Q_{m-1}^{R_{m-1}} \rightarrow Q_m^{R_m}$$

We formally define an event-streaming as follows:

Definition 15. *An event-streaming is a poset (ES_Θ, \rightarrow) where ES_Θ is a finite set of subsets of events $ES_\Theta = \{Q_1^{R_1}, Q_2^{R_2}, \dots, Q_m^{R_m}\}$ arranged according to the causal relation \rightarrow ; Θ is the set of the identifiers of the processes that generated the events; and each $Q_q^{R_q} \in ES_\Theta$ is a subset of events generated by the processes whose identifiers form the set R_q .*

We note that in a similar way to the local-streams and the intervals, each subset $Q_q^{R_q} \in ES_\Theta$ can be expressed by its endpoints. However, unlike local-streams and intervals, the endpoints of a subset $Q_q^{R_q}$ are sets of events formed by the endpoints of each set $\Omega_\zeta \in Q_q^{R_q}$, respectively. Thus, when $Q_q^{R_q}$ contains events generated by the processes p_i , p_j and p_k , we denote the left set endpoint as $^-Q_q^{R_q}$ which is composed as follows: $^-Q_q^{R_q} = \{\omega_i^-, \omega_j^-, \omega_k^-\}$, where $\omega_i^- \in \Omega_i$, $\omega_j^- \in \Omega_j$ and $\omega_k^- \in \Omega_k$. Likewise, we denote the right set endpoint as $^+Q_q^{R_q}$ which is composed as follows: $^+Q_q^{R_q} = \{\omega_i^+, \omega_j^+, \omega_k^+\}$, where $\omega_i^+ \in \Omega_i$, $\omega_j^+ \in \Omega_j$ and $\omega_k^+ \in \Omega_k$.

4.3 Problem formulation

Based on the definition of the data stream alignment problem given in [LS07], we define the problem of the data alignment for event-streaming as follows.

Definition 16. *Data alignment problem for event-streaming:* Given a set of local-streams: $\{S_1, S_2, S_3, \dots\}$, and considering a certain maximum transmission delay, the problem is to assign a temporal reference that can be used as an estimated sample time for each interested message, such that for every two messages $m(p_i, x)$ and $m(p_j, y)$, their synchronization error is bounded.

For our solution, the messages of interest are the causal messages sent within an event-streaming. Explicitly, they are the endpoints of the subsets $Q_q^{R_q}$. In this sense, the *synchronization error* establishes the temporal distance between the execution of a pair of messages of interest. Therefore, according to the stated problem, it is necessary to detect some patterns among local-streams that can be used as temporal references to certain messages.

As shown in the works of Chandra and Kshemkalyani [CK02, CK08, Ksh05] a practical way to detect patterns among local-streams is by assuming a global time axis and by using interval-interval relations as defined by Allen [All83]. Unfortunately, establishing a global time axis in a WSN is difficult due to the lack of perfectly-synchronized clocks [CK05]. However, as shown by Pomares et al. with their logical mapping model [PEMR08], the interval-interval relations can be expressed in terms of the happened-before relation, allowing the system to prescind from physical clocks.

For our problem of data alignment, we are specifically interested in detecting when an event e_μ of a local-stream S_i precedes another event e_ν of some local-stream S_j , and when the events of two or more local-streams are concurrent,

since these two facts help us identify the unique time slots during a time-line which can be used as temporal references to situate a message. Furthermore, the concurrences among sets of events identify data that can be fused in further stages.

In this sense, the logical mapping model is focused on detecting temporal dependencies between a pair of intervals. However, for our problem this means only the base case, which is the detection of temporal dependencies between local-streams. The logical mapping does not support the detection of temporal dependencies among event-streamings. To achieve this, we propose an extension to the native logical mapping model [PCPHM12]. We call this extended model event-streaming logical mapping model (ES-LM) [PCPH14a].

Chapter 5

Temporal data alignment for event-streaming

5.1 The event-streaming logical mapping model (ES-LM)

The native logical mapping identifies five logical mappings: *precedes*, *simultaneous*, *ends*, *starts* and *overlaps* to determine how two intervals (local-streams) are related.

Analogously, the ES-LM [PCPH14a] is designed to identify how an event-streaming ES_{Θ} is related to a local-stream Y_k in order to determine the events that have causal dependencies and the subsets of events that concur. To achieve this, it is necessary to determine how each subset $Q_q^{R_q} \in ES_{\Theta}$ is related to a local-stream Y_k . We note that any subset $Q_q^{R_q}$ is considered as a sub-event-streaming. In our ES-LM model, without loss of generality, it is assumed that $\neg Q_q^{R_q} \rightarrow Y_k^-$ or $\neg Q_q^{R_q} \parallel Y_k^-$. Therefore, when a subset $Q_q^{R_q}$ is aligned to a local-stream Y_k , a new sub-event-streaming is generated according to the left column of Table 4. The resultant sub-event-streaming has the following general causal structure:

$$Q_a^{R_a} \rightarrow Q_b^{R_b} \rightarrow Q_w^{R_w}$$

From this causal structure, five new logical mappings are identified. These logical mappings represent all the ways that an event-streaming can be related to a local-stream. These new logical mappings are: *s-precedes*, *s-simultaneous*, *s-ends*, *s-overlaps* and *s-starts* (see right column of Table 4).

Table 4: Event-streaming logical mapping

Data alignment for event-streaming		ES logical mappings
$Q_a^{R_a}$	← <ul style="list-style-type: none"> • if $\forall x^- \in {}^-Q_q^{R_q}, \exists y^- \in Y_k : x^- \rightarrow y^-$, $\{x \in Q_q^{R_q} : \text{delivery}(p_k, x) \rightarrow \text{send}(y^-)\}$ • otherwise, \emptyset 	<i>s-precedes</i> : $Q_a^{R_a} \rightarrow Q_w^{R_w}$
$Q_b^{R_b}$	← <ul style="list-style-type: none"> • if $y^+ \in Y_k : {}^+Q_q^{R_q} \rightarrow y^+$, $(Q_q^{R_q} - Q_a^{R_a}) \cup (Y_k - \{y \in Y_k : \text{delivery}(p_k, x^+) \rightarrow \text{send}(y)\})$ • if $\forall x^+ \in {}^+Q_q^{R_q}, \exists y^+ \in Y_k : y^+ \rightarrow x^+$, $(\{x \in Q_q^{R_q} : \text{delivery}(p_k, x) \rightarrow \text{send}(y^+)\} - Q_a^{R_a}) \cup Y_k$ • otherwise, $(Q_q^{R_q} - Q_a^{R_a}) \cup Y_k$ 	<i>s-simultaneous</i> : $\emptyset \rightarrow Q_b^{R_b}$
$Q_w^{R_w}$	← <ul style="list-style-type: none"> • if $\forall x^+ \in {}^+Q_q^{R_q}, \exists y^+ \in Y_k : y^+ \rightarrow x^+$, $Q_q^{R_q} - \{x \in Q_q^{R_q} : \text{delivery}(p_k, x) \rightarrow \text{send}(y^+)\}$ • otherwise, $\{y \in Y_k : \text{delivery}(p_k, x) \rightarrow \text{send}(y^+)\}$ 	<i>s-ends</i> : $Q_a^{R_a} \rightarrow Q_b^{R_b}$
		<i>s-overlaps</i> : $Q_a^{R_a} \rightarrow Q_b^{R_b} \rightarrow Q_w^{R_w}$
		<i>s-starts</i> : $Q_b^{R_b} \rightarrow Q_w^{R_w}$

The ES-LM is the core of the data alignment scheme that we propose, which is presented in the following section.

5.1.1 Data alignment description

The data alignment process is described through four stages as follows.

A.1 Initial stage: alignment of two local-streams Initially, we have two local-streams X_c and Y_d ($X_c^- \rightarrow Y_d^-$ or $X_c^- \parallel Y_d^-$). Applying the native logical mapping, we generate the first event-streaming ES_Θ as follows.

We construct a first subset $Q_1^{\{c\}}$ with the first non-concurrent events in X_c (see Table 5). To determine those non-concurrent events, we need to identify all the events $x \in X_c$ that precede the *beginning* of Y_d (see Figure 5.1).

Then, according to Table 5, we proceed to construct a second subset $Q_2^{\{c,d\}}$ with the concurrent events between X_c and Y_d . The concurrent segments of both local-streams will be bound by the *beginning* of Y_d and the *end* of any of the two local-streams (see Figure 5.2).

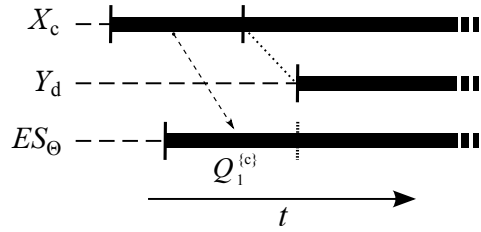


Figure 5.1: Aligning the first subset of the first event-streaming.

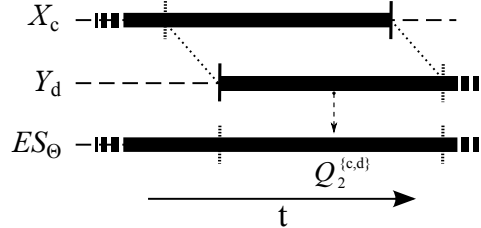


Figure 5.2: Aligning the second subset of the first event-streaming.

The last subset $Q_3^{\{w\}}$ is constructed depending on which local-stream finishes first. If X_c finishes first, the last subset will contain the remaining events of Y_d (see Table 5). Otherwise, the last subset will contain the remaining events of X_d (see Table 5). These two cases are illustrated in Figure 5.3.

Therefore, the first event-streaming has the general causal structure $ES_\Theta = Q_1^{\{c\}} \rightarrow Q_2^{\{c,d\}} \rightarrow Q_3^{\{w\}}$, where w may be c or d .

Once the first stage has finished, we proceed to align two streams: an event-streaming and a local-stream. The event-streaming is labeled as the set X_β , where

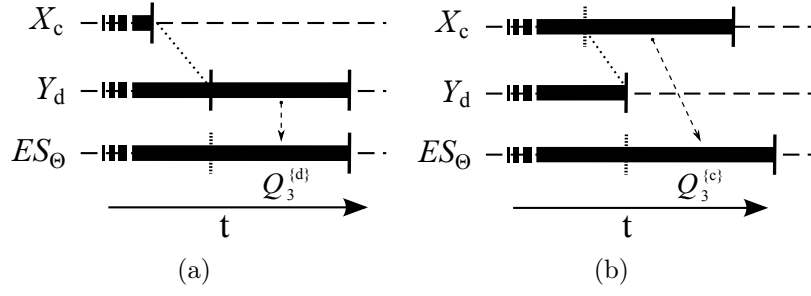


Figure 5.3: Aligning the last subset of the first event-streaming. (a) X_c finishes first (b) Y_d finishes first.

β is the set of identifiers of the processes that generated the events, while the local-stream is labeled as Y_k , where k is the identifier of the local process. Each resultant event-streaming is merged with the next most-left local stream Y_k according to the causal dependencies among the events of Y_k and the event-streaming X_β .

Using X_β and Y_k , we construct a new event-streaming forming the subsets $Q_q^{T_q}$ of the general causal structure $ES_\Theta = Q_1^{T_1} \rightarrow Q_2^{T_2} \rightarrow \dots \rightarrow Q_{n-1}^{T_{n-1}} \rightarrow Q_n^{T_n}$ by detecting the concurrences between X_β and Y_k .

Table 5: A.1 Alignment of two local-streams

$Q_1^{\{c\}}$	\leftarrow	<ul style="list-style-type: none"> • if $x^- \in X_c, y^- \in Y_d : x^- \rightarrow y^-$, $\{x \in X_c : delivery(p_d, x) \rightarrow send(y^-)\}$ • otherwise, \emptyset
$Q_2^{\{c,d\}}$	\leftarrow	<ul style="list-style-type: none"> • if $x^+ \in X_c, y^+ \in Y_d : x^+ \rightarrow y^+$, $(X_c - Q_1^{\{c\}}) \cup (Y_d - \{y \in Y_d : delivery(p_d, x^+) \rightarrow send(y)\})$ • if $x^+ \in X_c, y^+ \in Y_d : y^+ \rightarrow x^+$, $(\{x \in X_c : send(x) \rightarrow delivery(p_c, y^+)\} - Q_1^{\{c\}}) \cup Y_d$ • otherwise, $(X_c - Q_1^{\{c\}}) \cup Y_d$
$Q_3^{\{w\}}$	\leftarrow	<ul style="list-style-type: none"> • if $x^+ \in X_c, y^+ \in Y_d : y^+ \rightarrow x^+$, $X_c - (\{x \in X_c : send(x) \rightarrow delivery(p_c, y^+)\} - Q_1^{\{c\}})$ • otherwise, $\{y \in Y_d : delivery(p_d, x^+) \rightarrow send(y)\}$

Assuming that the initial stage was accomplished, the logical mapping proceeds according to the three stages that are detailed below.

B.1 Aligning the first subsets of events without concurrences between an event-streaming and a local-stream

In the first step, we determine if there are some subsets $Q_a^{R_a} \in X_\beta$ that precede the local-stream Y_k to form the first subsets of the new event-streaming (see stage B.1 of Table 6). These subsets have events that are not concurrent with the events of Y_k and are integrated directly to the new event-streaming to form the first subsets $Q_a^{T_a} \in ES_\Theta$ (see Figure 5.4).

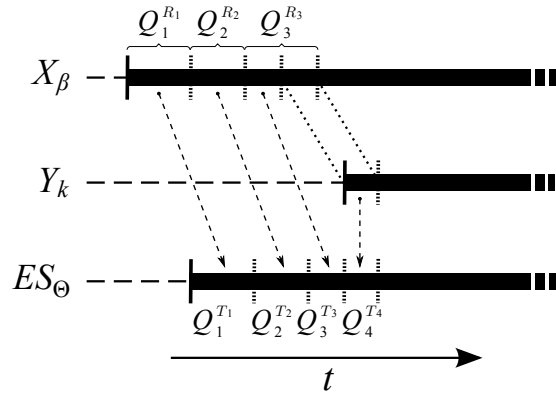


Figure 5.4: Aligning the first subsets of events of an event-streaming

If a subset $Q_a^{R_a} \in X_\beta$ has events that are concurrent with a part of the local-stream Y_k , this subset is segmented to form two new subsets for the new event-streaming ES_Θ . The new subset $Q_a^{T_a}$, the first of the two new subsets, will contain the part of $Q_a^{R_a}$ whose events have no concurrence. To determine the events without concurrences, it is necessary to identify the event $x \in Q_a^{R_a}$ that immediately precedes the beginning of the local-stream Y_k , (see line 1.2 of Table 6). For the example depicted in Figure 5.4, the new subset $Q_a^{T_a}$ corresponds to the subset $Q_3^{T_3}$. The remaining events of $Q_a^{R_a}$ are aligned as stated in the following stage.

Table 6: Alignment process between an event-streaming and a local-stream

B.1	If $\forall x^- \in X_\beta^-, \exists y^- \in Y_k : x^- \rightarrow y^-$ a new set $Q_a^{T_a}$ is created as follows:			
	$Q_a^{T_a}$	\leftarrow	<ul style="list-style-type: none"> • for each $Q_a^{R_a} \in X_\beta : \neg(\exists x \in Q_a^{R_a} : \text{delivery}(p_k, x) \downarrow \text{send}(y^-))$ 	1.1
		\leftarrow	<ul style="list-style-type: none"> • if $\exists x \in Q_a^{R_a} : \text{delivery}(p_k, x) \downarrow \text{send}(y^-)$, $Q' = \{x \in Q_a^{R_a} : \text{delivery}(p_k, x) \rightarrow \text{send}(y^-)\}$	1.2
Case A. If $Q_a^{R_a} - Q' \neq \emptyset$, a new set $Q_c^{T_c}$ is created by:				
	$Q_c^{T_c}$	\leftarrow	$(Q_a^{R_a} - Q') \cup (Y_k - \{y \in Y_k : \text{delivery}(p_k, {}^+ Q_a^{R_a}) \rightarrow \text{send}(y)\})$	2.1
Case B. If $\forall x^+ \in X_\beta^+, \exists y^+ \in Y_k : x^+ \rightarrow y^+$ a new set $Q_c^{T_c}$ as follows:				
	$Q_c^{T_c}$	\leftarrow	<ul style="list-style-type: none"> • for each $Q_b^{R_b} \in X_\beta - (Q_1^{R_1} \cup \dots \cup Q_a^{R_a})$, $Q_b^{R_b} \cup \{(Y_k - \{y \in Y_k : y \in Q_1^{T_1} \cup \dots \cup Q_{c-1}^{T_{c-1}}\}) - \{y \in Y_k : \text{delivery}(p_k, {}^+ Q_b^{R_b}) \rightarrow \text{send}(y)\}\}$	2.2
B.2	Case C. If $\forall x^+ \in X_\beta^+, \exists y^+ \in Y_k : y^+ \rightarrow x^+$ a new set $Q_c^{T_c}$ is created for each $Q_b^{R_b} \in X_\beta - (Q_1^{R_1} \cup \dots \cup Q_a^{R_a})$, such that $\forall x \in Q_b^{R_b} : \text{delivery}(p_k, x) \rightarrow \text{send}(y^+)$, as follows:			
	$Q_c^{T_c}$	\leftarrow	<ul style="list-style-type: none"> • for each $Q_b^{R_b} \in X_\beta : \neg(\exists x \in Q_b^{R_b} : \text{delivery}(p_k, x) \downarrow \text{send}(y^+))$, $Q_b^{R_b} \cup \{(Y_k - \{y \in Y_k : y \in Q_1^{T_1} \cup \dots \cup Q_{c-1}^{T_{c-1}}\}) - \{y \in Y_k : \text{delivery}(p_k, {}^+ Q_b^{R_b}) \rightarrow \text{send}(y)\}\}$	2.3
		\leftarrow	<ul style="list-style-type: none"> • if $\exists x \in Q_b^{R_b} : \text{delivery}(p_k, x) \downarrow \text{send}(y^+)$, $Q'' = \{x \in Q_b^{R_b} : \text{delivery}(p_k, x) \rightarrow \text{send}(y^+)\} \cup (Y_k - \{y \in Y_k : y \in Q_1^{T_1} \cup \dots \cup Q_a^{T_a}\})$	2.4
Case A. If $\forall x^+ \in X_\beta^+, \exists y^+ \in Y_k : y^+ \rightarrow x^+$ a new set $Q_d^{T_d}$ as follows:				
B.3	$Q_d^{T_d}$	\leftarrow	• if $Q_b^{R_b} - Q'' \neq \emptyset$, $Q_b^{R_b} - Q''$	3.1
		\leftarrow	• otherwise, for each $Q_c^{R_c} \in \{X_\beta - Q_1^{T_1} \cup \dots \cup Q_n^{T_n}\}$	3.2
	Case B. If $\forall x^+ \in X_\beta^+, \exists y^+ \in Y_k : x^+ \rightarrow y^+$ a new set $Q_d^{T_d}$ is created by:			
	$Q_d^{T_d}$	\leftarrow	$Y_k - \{y \in Y_k : y \in Q_1^{T_1} \cup \dots \cup Q_n^{T_n}\}$	3.3

B.2 Aligning the subsets of events with concurrences between an event-streaming and a local-stream

If during stage B.1 a subset $Q_a^{R_a}$ was detected containing events concurrent with a portion of the local-stream Y_k , such a portion of Y_k is attached to the part of $Q_a^{R_a}$ with concurrent events to form a new subset $Q_c^{T_c}$ (see line 2.1 of Table 6).

Once the beginning of the concurrent parts of both streams are detected, according to stage B.2 of Table 6 (lines 2.2 and 2.3), all the subsequent subsets $Q_b^{R_b} \in X_\beta$ are attached with the corresponding concurrent events of Y_k , until one of the two streams finishes. This means that for each subset $Q_b^{R_b}$ in the concurrent part of X_β , a new subset $Q_c^{T_c}$ will be constructed for the new event-streaming ES_Θ (see Figure 5.5).

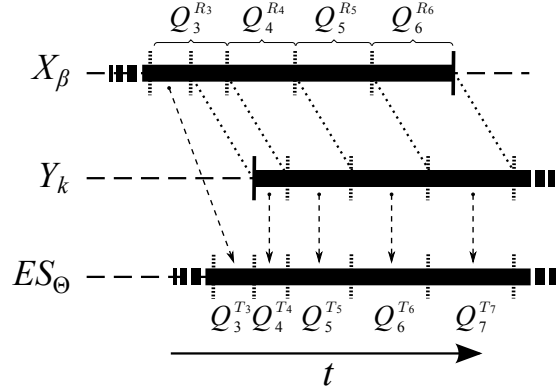


Figure 5.5: Aligning the subsets of events with concurrences.

The final subsets of the resulting event-streaming will be constructed depending upon which stream finishes first.

If the local-stream Y_k finishes first ($y^+ \rightarrow X_\beta^+$), the last concurrent subset $Q_b^{Rb} \in X_\beta$ can contain some events that are concurrent with Y_k and other events that have no concurrence (see Figure 5.6). If this is the case, Q_b^{Rb} needs to be segmented to construct two new subsets for the new event-streaming ES_Θ . The new subset Q_c^{Tc} , the first of the two new subsets, will contain the concurrent part of Q_b^{Rb} and the concurrent events of Y_k . To determine such concurrent events, it is necessary to identify the event $x \in Q_b^{Rb}$ that immediately precedes the end of the local-stream Y_k and the concurrent part of the local-stream (see line 2.4 in Table 6). For the example depicted in Figure 5.6, the new subset Q_c^{Tc} corresponds to subset Q_7^{T7} . The remaining events of Q_b^{Rb} are aligned as stated in the following stage.

B.3 Aligning the last subsets of events without concurrences This stage is explained through two cases.

Case A. Y_k finishes before X_β . If at the end of stage B.2, the last subset Q_b^{Rb} was segmented, the second created subset, denoted as Q_d^{Td} , will contain the

remaining non-concurrent events of $Q_b^{R_b}$ (see line 3.1 of Table 6). In the example of Figure 5.6, such subset $Q_d^{T_d}$ corresponds to the subset $Q_8^{T_8}$.

The fact that the local-stream Y_k finishes first ($y^+ \rightarrow X_\beta^+$) implies that the concurrent parts of both streams finish along with Y_k . Therefore, according to line 3.2 of Table 6, the remaining subsets $Q_c^{R_c} \in X_\beta$ will become the last subsets $Q_d^{T_d} \in ES_\Theta$. In the example of Figure 5.6, the last subsets $Q_d^{T_d}$ are the subsets $Q_9^{T_9}$, $Q_{10}^{T_{10}}$ and $Q_{11}^{T_{11}}$.

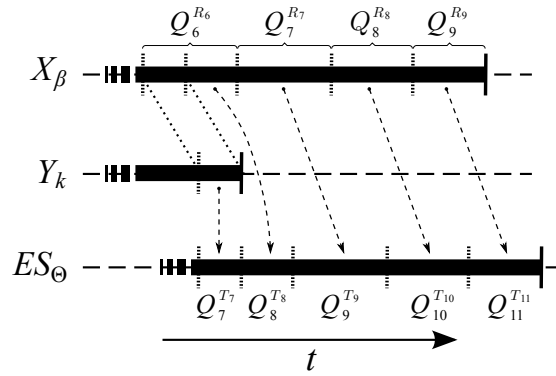


Figure 5.6: Aligning the last subsets of events without concurrences when $y^+ \rightarrow X_\beta^+$.

Case B. X_β finishes before Y_k . The fact that the event-streaming X_β finishes first ($X_\beta^+ \rightarrow y^+$, $y^+ \in Y_k$) means that the concurrent parts of both streams finish along with the event-streaming X_β . After the last subset $Q_c^{T_c} \in ES_\Theta$ was constructed with the concurrent events of X_β and Y_k , only one more subset $Q_d^{T_d}$ is constructed according to line 3.3 of Table 6. Such subset $Q_d^{T_d}$ will contain the remaining events of the local-stream Y_k . In the example of Figure 5.7, the last subset $Q_d^{T_d}$ corresponds to the subset $Q_8^{T_8}$.

The event-streaming logical mapping will continue until there are no more concurrent local-streams to be merged.

In terms of data alignment, we note that by the way in which the subsets of events are constructed and causally ordered, each resultant event-streaming ES_Θ

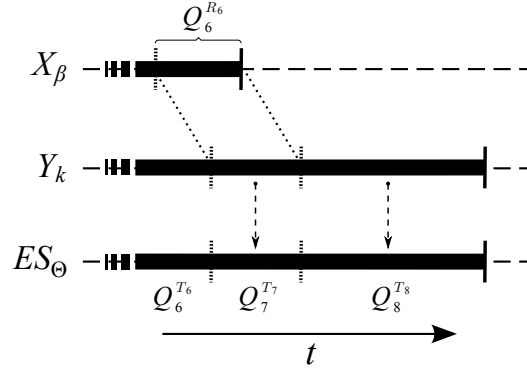


Figure 5.7: Aligning the last subsets of events without concurrences when $X_\beta^+ \rightarrow y^+$.

is a finite collection of disjoint subsets $Q_q^{T_q}$ arranged one after another without interruption. This arrangement of subsets $Q_q^{T_q}$ in an ES_Θ allows us to establish a virtual time-line where each subset $Q_q^{T_q}$ represents a unique time-slot. The fact that the subsets $Q_q^{T_q}$ are disjoint implies that each event in an event-streaming belongs to a unique subset $Q_q^{T_q}$, and therefore it is located at a specific time-slot.

5.1.2 Formal proof of the ES-LM model

In this section we prove that by following the ES-LM data alignment model, the sequential arrangement of subsets of events that compose an event-streaming establishes a virtual time-line, where each subset represents a unique time-slot and each event is aligned with respect to only one of them.

Theorem 1. *The arrangement of subsets $Q_q^{R_q}$ in an event-streaming establishes a virtual time-line, where each subset $Q_q^{R_q}$ represents a unique time-slot and each event belongs to a unique time-slot.*

Proof. We divide this proof into two parts. In the first part we prove that an event-streaming is a causal arrangement of subsets of events that establishes a

time-line. In the second part we prove that each subset $Q_q^{R_q}$ in an event-streaming represents a unique time-slot.

Part I. To demonstrate that an event-streaming is a causal arrangement of subsets of events that establishes a time-line, we formulate and prove the following Lemma:

Lemma 1. *An event-streaming is a causal arrangement of subsets of events that establishes a time-line.*

Before proving Lemma 1, we need to consider the following. Definition 14, states that a local-stream is a poset (S_i, \rightarrow) , where S_i is a set of atomic events generated by the same process. Thus, (S_i, \rightarrow) is a sequence $S_i = \{e_\alpha \rightarrow e_{\alpha+1} \rightarrow \dots \rightarrow e_{n-1} \rightarrow e_n\}$. The fact that the events of S_i are arranged by \rightarrow implies that each event happens before another at different time instant, which determines a chronologically order. Therefore, a local-stream S_i represents a time-line for a process i .

Proof of Lemma 1. We demonstrate Lemma 1 by a direct proof. According to the ES-LM (Tables 4, 5 and 6) during the data alignment, the subsets of events that compose a new event-streaming ES_β , are formed by segmenting two streams (a local-stream and an event-streaming or two local-streams). From Tables 5 and 6 (specifically, stage B.1; cases B and C of stage B.2; and stage B.3) we have that a segmentation is triggered by the identification of an endpoint which determines the beginning or the ending of an overlap between a pair of streams. Whichever the case, each segmentation establishes the creation of two new subsets $Q_{u-1}^{R_u-1}$ and $Q_u^{R_u}$. Let Y_c be a local-stream and let \mathfrak{X} denote a local-stream or an event-streaming such that $\forall x^- \in \mathfrak{X}, \exists y^- \in Y_c : x^- \rightarrow y^-$. Assuming that e^* ($e^* \in Y_c$

or $e^* \in \mathfrak{X}$) is the endpoint ($e^* = y^- \vee e^* = y^+ \vee e^* = x^+$), whose identification triggered the segmentation, $\forall x^+ \in \mathfrak{X}$, $Q_{u-1}^{R_{u-1}}$ and $Q_u^{R_u}$ are constructed according to one of the three following cases:

- a) if $e^* \in Y_c$, $e^* = y^-$, and $x^- \rightarrow y^-$ then $Q_{u-1}^{R_{u-1}} = \{x \in \mathfrak{X} : \text{delivery}(p_c, x) \rightarrow \text{send}(e^*)\}$ and $Q_u^{R_u} = \{e^*\} \cup \{x \in \mathfrak{X} : \text{send}(e^*) \rightarrow \text{delivery}(p_c, x)\} \cup \{y \in Y_c : \text{send}(y) \rightarrow \text{delivery}(p_c, x^+)\}$;
- b) if $e^* \in \mathfrak{X}$, $e^* = x^+$, and $x^+ \rightarrow y^+$ then $Q_{u-1}^{R_{u-1}} = \{e^*\} \cup \{x \in \mathfrak{X} : \text{delivery}(p_c, x) \rightarrow \text{delivery}(p_c, e^*)\} \cup \{y \in Y_c : \text{send}(y) \rightarrow \text{delivery}(p_c, e^*)\}$ and $Q_u^{R_u} = \{y \in Y_c : \text{delivery}(p_c, e^*) \rightarrow \text{send}(y)\}$;
- c) if $e^* \in Y_c$, $e^* = y^+$, and $y^+ \rightarrow x^+$ then $Q_{u-1}^{R_{u-1}} = \{e^*\} \cup \{x \in \mathfrak{X} : \text{delivery}(p_c, x) \rightarrow \text{send}(e^*)\} \cup \{y \in Y_c : \text{send}(y) \rightarrow \text{send}(e^*)\}$ and $Q_u^{R_u} = \{x \in \mathfrak{X} : \text{send}(e^*) \rightarrow \text{delivery}(p_c, x)\}$.

On each of these three cases, happened-before relationships are established among the left endpoints of $Q_{u-1}^{R_{u-1}}$ and the right endpoints of $Q_u^{R_u}$. This means that for a pair of subsets $Q_{u-1}^{R_{u-1}}$ and $Q_u^{R_u}$ ($Q_{u-1}^{R_{u-1}}, Q_u^{R_u} \in ES_\beta$) we have that $\forall (\omega_i^+, \omega_k^-) \in {}^+Q_{u-1}^{R_{u-1}} \times {}^-Q_u^{R_u} : \omega_i^+ \rightarrow \omega_k^-$ ($i, k \in \beta$). Furthermore, all the events that compose $Q_{u-1}^{R_{u-1}}$ and $Q_u^{R_u}$ are extracted from local-streams by preserving their causal order. Thus, let ω_i and ω_i^+ be two events such that $\omega_i, \omega_i^+ \in \Omega_i$, $\Omega_i \in Q_{u-1}^{R_{u-1}}$ and $\Omega_i \subseteq S_i$ (for any local-stream S_i) if $\omega_i \neq \omega_i^+$ then $\omega_i \rightarrow \omega_i^+$. By the transitive property of the HBR we have that $\omega_i \rightarrow \omega_k^-$, $\forall \omega_k^- \in Q_u^{R_u}$. Moreover, for any event $\omega_k \in Q_u^{R_u}$, $\omega_k \neq \omega_k^-$, $\omega_k^- \rightarrow \omega_k$; thereby, transitively $\omega_i \rightarrow \omega_k$. Therefore, by Definition 7 we have $Q_{u-1}^{R_{u-1}} \rightarrow Q_u^{R_u}$. Thus the subsets of an event-streaming are chronologically ordered representing a time-line, where each subset $Q_q^{R_q}$ is a time-slot. \square

Corollary 1. *Each subset $Q_q^{R_q} \in ES_\Theta$ represents a time-slot.*

Part II. To demonstrate that each subset $Q_q^{R_q}$ in an event-streaming represents a unique time-slot, we formulate and prove the following Lemma:

Lemma 2. *Each subset $Q_q^{R_q}$ represents a unique time-slot, therefore, any pair of subsets $Q_u^{R_u}$ and $Q_v^{R_v}$, that compose an event-streaming ES_Θ , is disjoint.*

$$\forall Q_u^{R_u}, Q_v^{R_v} \in ES_\Theta, u \neq v : Q_u^{R_u} \cap Q_v^{R_v} = \emptyset$$

Proof of Lemma 2. We prove Lemma 2 by contradiction. Therefore, we suppose that $\forall Q_u^{R_u}, Q_v^{R_v} \in ES_\Theta, u \neq v : Q_u^{R_u} \cap Q_v^{R_v} \neq \emptyset$, i.e., $\exists x_\mu : x_\mu \in Q_u^{R_u} \wedge x_\mu \in Q_v^{R_v}$.

According to the ES-LM model, the subsets $Q_q^{R_q}$ of an event-streaming are created by aligning two local streams or aligning a local stream with an event-streaming. Whichever way an event-streaming is generated, a subset $Q_u^{R_u}$ must be related to another $Q_v^{R_v}$ ($u \neq v$) according to one of the five logical mappings described in Table 2. This means that for any $e_\mu \in Q_u^{R_u}$ and any $c_\mu \in Q_v^{R_v}$, $e_\mu \rightarrow c_\mu$ or $c_\mu \rightarrow e_\mu$. Thus, if there is x_μ such that $x_\mu \in Q_u^{R_u} \wedge x_\mu \in Q_v^{R_v}$, implies that $x_\mu \rightarrow x_\mu$ is a contradiction, according to the assumptions by which the happened-before relation are defined (systems in which an event can happen before itself do not seem to be physically meaningful). \square

5.2 Temporal data alignment mechanism for event-streaming

The proposed data alignment mechanism fulfills at runtime three main tasks: first, it detects the temporal dependencies based on a time-line and translates them to causal dependencies among streams; second, it performs the construction of disjoint causally-ordered subsets of events to arrange the events that are concurrently

generated; and finally, it performs the arrangement of such subsets of events in a virtual time-line. The proposed mechanism is designed to perform the data alignment in a cooperative way by the different nodes involved in a WSN.

5.2.1 Mechanism specification

The data alignment mechanism is specified by three distributed algorithms for the data exchange and two algorithms for managing the control information. The data exchange algorithms use three different causal messages: *begin*, *end* and *discrete*, and a type of FIFO messages: *fifo_p*. A brief description of the main components of the algorithms is provided below.

Messages. A message m in the data exchange algorithms is a tuple $m = (k, sts, t_{sts}, TP, H_m, data)$, where:

- k is the local process identifier.
- sts is the identifier of the process that originally generated the messages of an event-streaming (original source¹).
- t_{sts} is the value of the local clock, in the original source.
- TP is the message type (*begin*, *end*, *discrete* and *fifo_p*).
- H_m , the immediate history of m , contains the identifiers of the messages (k, sts, t_{sts}) that immediately precede m . For *fifo_p* the set H_m is always the empty set.
- $data$ is the structure that carries the media data.

¹When $k = sts$, the stream is a local-stream, otherwise is an event-streaming

Data structures. The status of a process k and the streams that it is aligning are defined by four structures and one variable.

- $VT_p[]$ is the vector clock established by Mattern and Fidge [Mat88, Fid88]. The size of $VT_p[]$ is equal to the number of nodes that are associated with a node which performs the data alignment.
- $P_involved_p$ is a set with the identifiers of the processes that originated the messages that compose an event-streaming.
- $Causal_m_p$ is a set with the identifiers of the processes that generated the last aligned subset $Q_q^{R_q}$.
- ES_CI_p is a set composed by entries (k, sts, t_{sts}) used to store the immediate history that will be piggybacked in certain messages.
- ES_Act is a state variable that indicates when a process is aligning an event-streaming.

As we previously mentioned, the data alignment mechanism is specified by five algorithms: two algorithms for managing the control information (Algorithms 1 and 2) and three algorithms for the data exchange (Algorithms 3, 4 and 5).

Algorithm 1 performs the initialization of the data structures, used for managing the control information.

Algorithm 1: Initialization

```

1: procedure INITIALIZE()
2:    $VT_p[k] \leftarrow 0, \forall k : 1 \dots n$ 
3:    $P\_involved_p \leftarrow \emptyset$ 
4:    $Causal\_m_p \leftarrow \emptyset$ 
5:    $ES\_CI_p \leftarrow \emptyset$ 
6:    $ES\_Act \leftarrow 0$ 
7: end procedure

```

Algorithm 2 performs the operations required to construct the immediate history sets, that are piggybacked in certain messages in order to establish a segmentation of a local-stream or an event-streaming.

Algorithm 2: Construction of the immediate history sets

```

8: procedure SEGMENTING_ES( $TP = \{begin|end|discrete\}$ )
9:   if  $Causal\_m_p \neq \emptyset$  then
10:      $ES\_CI_p \leftarrow \emptyset$ 
11:     for all  $x \in P\_involved_p$  do
12:        $ES\_CI_p \leftarrow ES\_CI_p \cup \{(i, x, VT_p[x])\}$ 
13:     end for
14:   end if
15: end procedure

```

Algorithm 3 performs the operations required to send two types of causal messages, *begin* and *end*, and the FIFO type message *fifo_p*.

Algorithm 3: Sending of messages, performed by a process i

```

16: procedure SENDCONTINUOUS( $TP = \{begin|end|fifo\_p\}$ )
17:    $VT_p[i] \leftarrow VT_p[i] + 1$ 
18:   if  $TP = begin$  then
19:     if  $ES\_Act \neq 0$  then           /* if process  $i$  is generating a local-stream */
20:       SEGMENTING_ES( $TP$ )           /* or is aligning an event-streaming, */
21:     else                             /* it performs a segmentation */
22:        $ES\_Act \leftarrow 1$ 
23:     end if
24:      $Causal\_m_p \leftarrow P\_involved_p$ 
25:      $P\_involved_p \leftarrow P\_involved_p \cup \{i\}$ 
26:     SENDING( $i, i, VT_p[i], TP, ES\_CI_p, data$ )
27:   else if  $TP = end$  then
28:      $H_m \leftarrow ES\_CI_p$ 
29:     if  $ES\_Act \neq 0$  then
30:       SEGMENTING_ES( $TP$ )
31:     end if
32:      $P\_involved_p \leftarrow P\_involved_p - \{i\}$ 
33:      $Causal\_m_p \leftarrow P\_involved_p$ 
34:     if  $P\_involved_p = \emptyset$  then

```

Continued on next page

Algorithm 3 – continued from previous page

```

35:   |   |  $ES\_Act \leftarrow 0$ 
36:   |   | end if
37:   |   |  $SENDING(i, i, VT_p[i], TP, H_m, data)$ 
38:   |   | else
39:   |   |   | if  $i \in Causal\_m_p$  then
40:   |   |   |   |  $Causal\_m_p \leftarrow Causal\_m_p - \{i\}$            /* the first messages after a */
41:   |   |   |   |  $SENDING(i, i, VT_p[i], begin, ES\_CI_p, data)$        /* segmentation are sent */
42:   |   |   |   | if  $Causal\_m_p = \emptyset$  then                       /* as begin */
43:   |   |   |   |   |  $ES\_CI_p \leftarrow \emptyset$ 
44:   |   |   |   |   | end if
45:   |   |   |   | else
46:   |   |   |   |   |  $SENDING(i, i, VT_p[i], TP, \emptyset, data)$ 
47:   |   |   |   |   | end if
48:   |   |   |   | end if
49: end procedure

```

Algorithm 4 performs the operations required to send the special type of causal message *discrete*. In Section 5.2.2, we explain the utility of the *discrete* messages and when such type of messages are used.

Algorithm 4: Sending of *discrete* messages, performed by a process i

```

50: procedure SENDDISCRETE( $TP = \{discrete\}$ )
51:   |  $VT_p[i] \leftarrow VT_p[i] + 1$ 
52:   | if  $ES\_Act \neq 0$  then
53:   |   |  $SEGMENTING\_ES(TP)$ 
54:   |   |  $Causal\_m_p \leftarrow P\_involved_p$ 
55:   |   |  $H_m \leftarrow ES\_CI_p$ 
56:   |   |  $ES\_CI_p \leftarrow \{(i, i, VT_p[i])\}$ 
57:   |   | else
58:   |   |   |  $H_m \leftarrow \emptyset$ 
59:   |   |   |  $ES\_CI_p \leftarrow \{(i, i, VT_p[i])\}$ 
60:   |   | end if
61:   |  $SENDING(i, i, VT_p[i], TP, H_m, data)$ 
62: end procedure

```

Algorithm 5 performs the operations required to retransmit or deliver the received packets.

Algorithm 5: Reception of messages, performed by a process j

```

63: procedure RECEIVE( $m = (i, sts, t_{sts}, TP, H_m, data)$ )
64:   if  $t_{sts} = VT_p[sts] + 1$  then                                /* FIFO delivery condition */
65:     if  $TP \neq fifo\_p$  then
66:       if  $t' \leq VT_p[l] \ \forall (u, l, t') \in H_m$  then          /* causal delivery condition */
67:          $VT_p[sts] \leftarrow VT_p[sts] + 1$ 
68:         if  $TP = begin$  then
69:           if  $sts \notin Causal\_m_p$  then
70:             if  $ES\_Act \neq 0$  then
71:               SEGMENTING_ES( $begin$ )
72:                $Causal\_m_p \leftarrow P\_involved_p$ 
73:             else
74:                $ES\_Act \leftarrow 1$ 
75:             end if
76:              $P\_involved_p \leftarrow P\_involved_p \cup \{sts\}$ 
77:           else
78:              $Causal\_m_p \leftarrow Causal\_m_p - \{sts\}$ 
79:           end if
80:           DELIVERY( $j, sts, t_{sts}, TP, ES\_CI_p, data$ )          /* if it is required by  $j$  */
81:           SENDING( $j, sts, t_{sts}, TP, ES\_CI_p, data$ )          /* if  $j$  is the sink, Sending */
82:         else if  $TP = end$  then                                /* is not executed */
83:            $H_m \leftarrow ES\_CI_p$ 
84:           if  $|P\_involved_p| > 1$  then
85:             SEGMENTING_ES( $TP$ )
86:           else
87:              $ES\_CI_p \leftarrow \emptyset$ 
88:              $ES\_Act \leftarrow 0$ 
89:           end if
90:           DELIVERY( $j, sts, t_{sts}, TP, H_m, data$ )              /* if it is required by  $j$  */
91:           SENDING( $j, sts, t_{sts}, TP, H_m, data$ )              /* if  $j$  is the sink, Sending */
92:            $P\_involved_p \leftarrow P\_involved_p - \{sts\}$         /* is not executed */
93:           if  $Causal\_m_p = \emptyset$  then
94:              $Causal\_m_p \leftarrow P\_involved_p$ 
95:           else
96:              $Causal\_m_p \leftarrow Causal\_m_p - \{sts\}$ 

```

Continued on next page

Algorithm 5 – continued from previous page

```

97:         end if
98:     else if  $TP = discrete$  then
99:         if  $ES\_Act \neq 0$  then
100:             SEGMENTING_ES( $TP$ )
101:              $Causal\_m_p \leftarrow P\_involved_p$ 
102:              $H_m \leftarrow ES\_CI_p$ 
103:              $ES\_CI_p \leftarrow \{(j, sts, t_{sts})\}$ 
104:         else
105:              $H_m \leftarrow \emptyset$ 
106:         end if
107:         DELIVERY( $j, sts, t_{sts}, TP, H_m, data$ )           /* if it is required by j */
108:         SENDING( $j, sts, t_{sts}, TP, H_m, data$ )          /* if j is the sink, Sending */
109:     end if                                               /* is not executed */
110: else
111:     WAIT()
112: end if
113: else if ( $sts \in P\_involved_p$ ) then
114:      $VT_p[sts] \leftarrow VT_p[sts] + 1$ 
115:     if  $sts \in Causal\_m_p$  then
116:         DELIVERY( $j, sts, t_{sts}, begin, ES\_CI_p, data$ )
117:         SENDING( $j, sts, t_{sts}, begin, ES\_CI_p, data$ )
118:          $Causal\_m_p \leftarrow Causal\_m_p - \{sts\}$ 
119:         if  $Causal\_m_p = \emptyset$  then
120:              $ES\_CI_p \leftarrow \emptyset$ 
121:         end if
122:     else
123:         DELIVERY( $j, sts, t_{sts}, TP, H_m, data$ )           /* if it is required by j */
124:         SENDING( $j, sts, t_{sts}, TP, H_m, data$ )          /* if j is the sink, Sending */
125:     end if                                               /* is not executed */
126: end if
127: else
128:     WAIT_FIFO()
129: end if
130: end procedure

```

5.2.2 Functional description of the data alignment mechanism

Generation of local-streams. In the proposed data alignment mechanism, the two endpoints λ_i^- and λ_i^+ that delimit a local-stream S_i (see Definition 14), are determined by the emission or the reception of two causal messages: *begin* for λ_i^- and *end* for λ_i^+ . Therefore, a local-stream is generated by sending a *begin* message followed by certain number of *fifo_p* messages (see lines 18-26 and 46 of Algorithm 3). To notify another process that the generation of a local-stream has finished, the source process sends an *end* message (see lines 27-37 of Algorithm 3).

Alignment of two local-streams. In order to explain how the mechanism performs the alignment of two local-streams, we take the scenario depicted in Figure 5.8 as an example.

A process c generates the local-stream X_c by sending the messages $x_1 \dots x_6$ (see Figure 5.8).

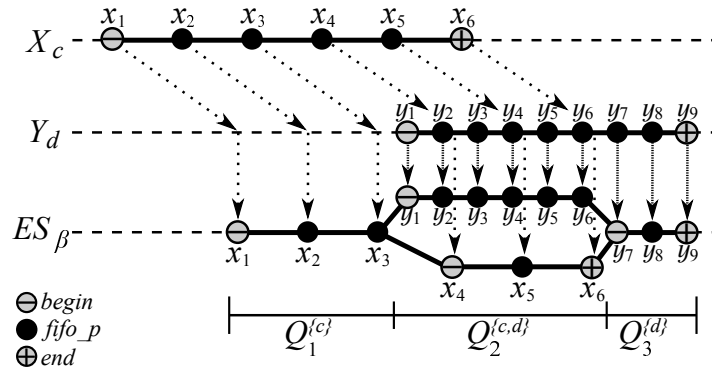


Figure 5.8: Data alignment of two local-streams, performed collaboratively by two different processes.

Once process d receives the messages of X_c , it simply forwards these messages until it starts to generate its own local-stream (see lines 67-81 and 113-123 of Algorithm 5). Any process can receive the messages of a local-stream from another process, before or during the generation of its own local-stream. Whatever the case, as soon as an overlap between the local-streams is detected, an event-streaming is generated by performing the necessary segmentations.

When process d starts to generate the local-stream Y_d , a segmentation is performed to determine $Q_1^{\{c\}}$ and $Q_2^{\{c,d\}}$. To do this, the first message of the sequence $y_1 \dots y_9$ and the next forwarded message x_4 are sent as a *begin* message (lines 41 and 116). The *begin* messages contain, as immediate history H_m , the identifiers of each of the latest messages sent by the processes whose events compose the event-streaming. In this example, y_1 and x_4 contain $H_m = \{x_3\}$. In this way, $Q_1^{\{c\}} \in ES_\beta$ is determined by $\{x_1, x_2, x_3\}$, where x_1 and x_3 are the endpoints. The next subset $Q_2^{\{c,d\}} \in ES_\beta$ is determined by y_1 and x_4 along with the following *fifo_p* messages generated by processes c and d .

Process c sends x_6 as an *end* message to indicate that X_c has finished. By finishing X_c , the part with concurrences between X_c and Y_d also finishes. For this, at the arrival of x_6 , process d performs a segmentation by sending the next message y_7 as *begin*, with $H_m = \{x_6, y_6\}$ (see lines 83-96 and 116 of Algorithm 5). Thereby, $-Q_2^{\{c,d\}} = \{x_4, y_1\}$ and $+Q_2^{\{c,d\}} = \{x_6, y_6\}$. Finally, process d sends y_9 as an *end* message, finishing the generation of the event-streaming. Thus $Q_3^{\{d\}} \in ES_\beta$ is determined by $\{y_7, y_8, y_9\}$ with y_7 and y_9 as the endpoints.

Alignment of an event-streaming with a local-stream. We take the scenario example of Figure 5.9 to explain how the mechanism performs the stages B.1, B.2 and B.3 of the ES-LM.

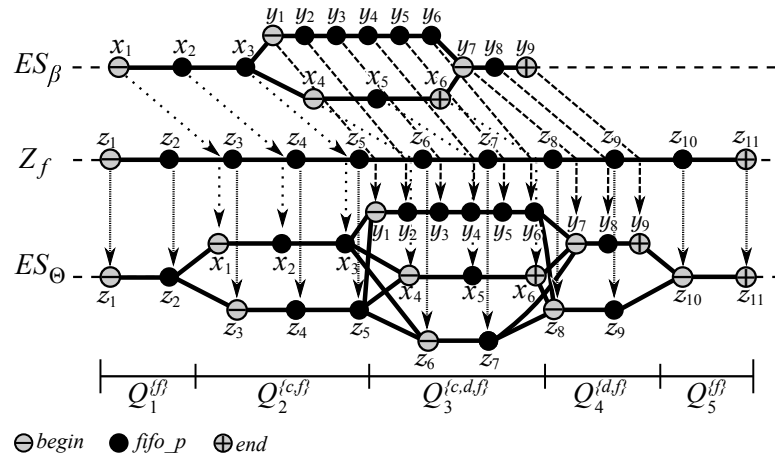


Figure 5.9: Data alignment of three local-streams, performed collaboratively by three different processes.

In this example, process f receives the messages of the event-streaming ES_β during the generation of the local-stream Z_f . Once the process f receives x_1 , a segmentation is performed to determine $Q_1^{\{f\}}$ and $Q_2^{\{c,f\}}$ by forwarding x_1 , and sending z_3 as *begin* messages, both with $H_m = \{z_2\}$ (see lines 67-81 of Algorithm 5 and lines 39-41 of Algorithm 3). Thus, z_1 and z_2 determine $Q_1^{\{f\}}$, while $Q_2^{\{c,f\}}$ is determined by x_1 and z_3 along with the next *fifo_p* messages generated by c and f , prior to the reception of y_1 .

At the reception of y_1 , process f performs a segmentation by sending y_1 , x_4 and z_6 as *begin* messages with $H_m = \{x_3, z_5\}$. In this way, the construction $Q_3^{\{c,d,f\}}$ is started. When process f receives the *end* message x_6 , it performs one more segmentation by sending z_7 and y_7 as *begin* messages with $H_m = \{x_6, y_6, z_7\}$. In this way, $Q_3^{\{c,d,f\}}$ is determined by the messages y_1 to y_6 , x_4 to x_6 , z_6 and z_7 .

When the reception of y_9 indicates the end of ES_β , process f performs the last segmentation by sending z_{10} as *begin* with $H_m = \{y_9, z_9\}$ and $Q_4^{\{d,f\}}$ is determined by y_7, y_8, y_9, z_8 and z_9 .

Finally, when process f finishes generating the local-stream Z_f , and thereby the construction of ES_Θ , it sends x_{11} as an *end* message. Thus, the last subset $Q_5^{\{f\}}$ is determined by z_{10} and z_{11} .

Therefore, the event-streaming ES_Θ , formed by the messages sent by processes c , d and f , is determined as the causal structure $Q_1^{\{c\}} \rightarrow Q_2^{\{c,d\}} \rightarrow Q_3^{\{c,d,f\}} \rightarrow Q_4^{\{d,f\}} \rightarrow Q_5^{\{f\}}$.

Aligning discrete messages. According to Definition 6, when a local-stream W_d contains only one message ($|W_d| = 1$), such a single message represents both endpoints $\lambda_d^- = \lambda_d^+$, $\lambda_i^-, \lambda_i^+ \in W_d$. In the data alignment mechanism, when a local-stream is composed by only one message, it is considered as a *discrete* message. In this case, since $\lambda_d^- = \lambda_d^+$, when a *discrete* message is aligned with a local-stream or an event-streaming, it determines a subset $Q_q^{R_q}$ for itself.

In the scenario of the Figure 5.10, when process d receives a local-stream X_c and generates a *discrete* message (see lines 50-53 of Algorithm 4), a segmentation is performed. In this way, the *discrete* message w_1 is sent with $H_m = \{x_2\}$. The next forwarded message x_3 of the local-stream X_c is sent as *begin* message with $H_m = \{w_1\}$. Thereby, $Q_1^{\{c\}}$ is determined by x_1 and x_2 , $Q_2^{\{d\}}$ is determined by w_1 , while $Q_3^{\{c\}}$ is determined by the remaining messages of X_c .

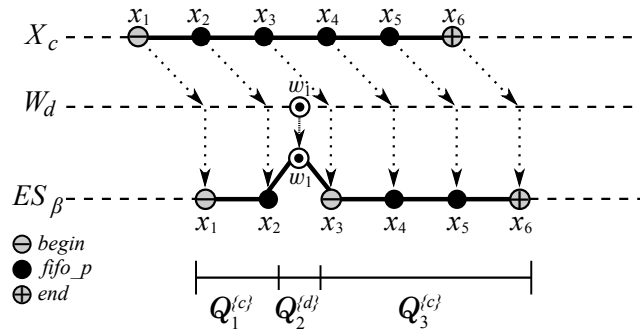


Figure 5.10: Data alignment of a discrete message with a local-stream.

When a *discrete* message is aligned with an event-streaming, such a discrete message also determines a subset $Q_q^{R_q}$. Taking the scenario of Figure 5.11, when process f receives w_1 , it performs a segmentation (see lines 98-108 of Algorithm 5) by forwarding w_1 with $H_m = \{x_2, z_5\}$, and sending the messages x_3 and z_6 as *begin* with $H_m = \{w_1\}$. Thereby, the event-streaming ES_Θ is composed by the subsets $Q_1^{\{f\}} = \{z_1, z_2, z_3, z_4\}$, $Q_2^{\{c,f\}} = \{x_1, x_2, z_5\}$, $Q_3^{\{d\}} = \{w_1\}$, $Q_4^{\{c,f\}} = \{x_3, x_4, x_5, x_6, z_6, z_7, z_8, z_9\}$ and $Q_5^{\{f\}} = \{z_{10}, z_{11}\}$.

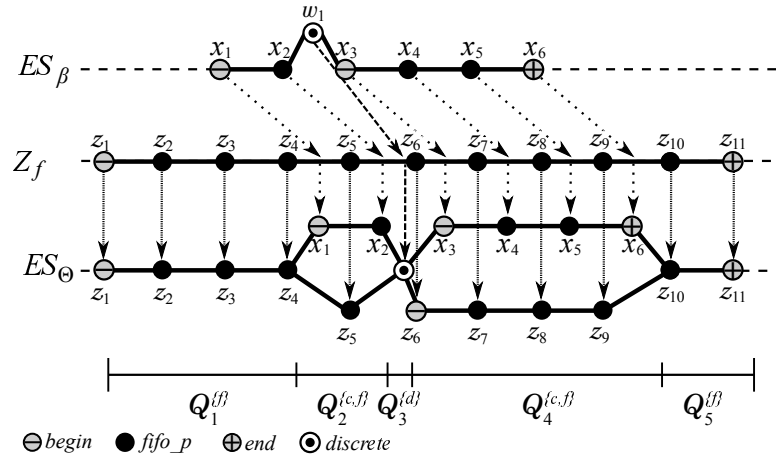


Figure 5.11: Data alignment of a discrete message with an event-streaming.

5.2.3 Correctness proof

In this section we prove the correctness of the temporal data alignment mechanism for event streaming. For this, we demonstrate that an event-streaming, generated by using the proposed mechanism, satisfies the conditions stated by Theorem 1.

This proof is divided into two parts. In the first part we prove that an event-streaming generated by the proposed mechanism establishes a time-line, satisfying Lemma 1. In the second part we prove that each subset $Q_q^{R_q}$ of an event-streaming represents a unique time-slot, satisfying Lemma 2.

Proposition 1. *An event-streaming generated by the data alignment mechanism satisfies Lemma 1.*

Proof. In the proposed mechanism, a local-stream S_j is generated by sequentially sending the following messages: a *begin* message followed by a certain number of *fifo_p* messages, followed by an *end* message. This means that $S_j = \{\lambda_j^- \downarrow \cdots \downarrow m_j \downarrow \cdots \downarrow \lambda_j^+\}$, where m_j is any *fifo_p* message. Each of these messages is time-stamped with a local logical clock t_{sts} and the identifier of its source sts . According to Algorithm 3, t_{sts} is determined by the current value of the $VT_p[sts]$ structure at the moment in which such message is generated (see lines 26, 37, 64 and 66). For the delivery, each message must satisfy the FIFO and the Causal conditions (see lines 66 and 68 of Algorithm 5). Since $VT_p[]$ has a monotonical increasing behavior and each message satisfies the respective delivery conditions, we can ensure that the messages of a local-stream are chronologically-ordered according to its causal dependencies. Thus, a local-stream can be seen as a local time-line for a process.

During the generation of an event-streaming, the subsets $Q_q^{R_q}$ are formed by performing some segmentations (see Section 5.1.1). A segmentation establishes

the end of a subset $Q_{u-1}^{R_{u-1}}$ and the initiation of a subset $Q_u^{R_u}$. This is achieved by registering all the latest messages received from the processes involved with the event-streaming in the field H_m of each subsequent *begin* message. This implies that for a pair of subsets $Q_{u-1}^{R_{u-1}}$ and $Q_u^{R_u}$, happened-before relations (HBRs) among the left endpoints of $Q_{u-1}^{R_{u-1}}$ and the right endpoints of $Q_u^{R_u}$ are established, such that, $\forall(\lambda_i^+, \lambda_k^-) \in {}^+Q_{u-1}^{R_{u-1}} \times {}^-Q_u^{R_u} : \lambda_i^+ \rightarrow \lambda_k^-$ ($i, k \in P_involved_p$). Since any message $m_i \in Q_{u-1}^{R_{u-1}}$ is extracted from a local-stream $(\lambda_i^+, m_i \in \Lambda_i, \Lambda_i \subseteq S_i)$, if $m_i \neq \lambda_i^+$ then $m_i \rightarrow \lambda_i^+$. By the transitive property of the HBR we have $m_i \rightarrow \lambda_k^-$, $\forall \lambda_k^- \in Q_u^{R_u}$. Moreover, for any message $m_k \in Q_u^{R_u}$, $m_k \neq \lambda_k^-$, $\lambda_k^- \rightarrow m_k$; thereby, transitively $m_i \rightarrow m_k$, $\forall(m_i, m_k) \in Q_{u-1}^{R_{u-1}} \times Q_u^{R_u}$. Therefore, by Definition 7 and Property 1, we have $Q_{u-1}^{R_{u-1}} \rightarrow Q_u^{R_u}$. Thus, the subsets of an event-streaming are chronologically ordered representing a time-line, where each subset $Q_q^{R_q}$ is a time-slot. \square

Proposition 2. *Any two subsets $Q_u^{R_u}$ and $Q_v^{R_v}$, generated by the data alignment mechanism, are disjoint, satisfying Lemma 2.*

Proof. We prove this fact by contradiction. Therefore, we suppose that $\forall Q_u^{R_u}, Q_v^{R_v} \in ES_{\Theta}, u \neq v : Q_u^{R_u} \cap Q_v^{R_v} \neq \emptyset$, i.e., $\exists x_\mu : x_\mu \in Q_u^{R_u} \wedge x_\mu \in Q_v^{R_v}$.

According to the proposed mechanism, the subsets $Q_q^{R_q}$ of an event-streaming are created by aligning two local-streams or aligning an event-streaming with a local-stream. Thus, let $Q_u^{R_u}$ and $Q_v^{R_v}$ subsets of an event-streaming ES_{Θ} ; and two events $c_\mu \in Q_u^{R_u}$ and $e_\mu \in Q_v^{R_v}$, we have two cases: $Q_u^{R_u} \rightarrow Q_v^{R_v}$ that implies $c_\mu \rightarrow e_\mu$; or $Q_v^{R_v} \rightarrow Q_u^{R_u}$ that implies $e_\mu \rightarrow c_\mu$. Whichever case, for any x_μ such that $x_\mu \in Q_u^{R_u} \wedge x_\mu \in Q_v^{R_v}$ implies that $x_\mu \rightarrow x_\mu$, which is a contradiction according to the assumptions of the happened-before relation definition (systems in which an event can happen before itself do not seem to be physically meaningful). On the other hand, if we assume two messages, m_c and m_e represent the same event, such

messages are mainly identified as: $m_c = (k', sts', t'_{sts})$, $m_e = (k'', sts'', t''_{sts})$, where $k' = k''$, $sts' = sts''$ and $t'_{sts} = t''_{sts}$. Furthermore, if we assume that $m_c \in Q_u^{R_u}$ and $m_e \in Q_v^{R_v}$, by the monotonical increasing behavior of $VT_p[]$ and the delivery conditions, $Q_u^{R_u} \rightarrow Q_v^{R_v}$ implies $t'_{sts} < t''_{sts}$ and $Q_v^{R_v} \rightarrow Q_u^{R_u}$ implies $t'_{sts} > t''_{sts}$, which contradicts the assumption: $t'_{sts} = t''_{sts}$. \square

5.2.4 Simulation results

We have simulated the temporal data alignment mechanism for event-streaming using the Castalia simulator [Bou13]. We configured an arrangement of 50 nodes, separated by distances between 10 and 15 meters in a field of 200 x 200 meters, with a multi-hop communication, as shown in the network graph of Figure 5.12. With this configuration, the depth of the network graph is established by the number of hops, thus, for our network scenario, the graph depth is up to 9. In addition to the network deployment, we adopted a special interference model, which allows to fix the delivery retries executed by a node, to ensure the reception of a packet by another node within a determined operational area. In this way, we obtained an average transmission delay of 0.30442 seconds between a pair of nodes.

The simulation was configured with the TMAC protocol for the MAC sublayer and the CC2420 radio protocol for wireless transmissions. The data payload for the Application layer packets was fixed to 2000 bytes.

In the simulation scenario, each node generated a random number of local-streams throughout the simulation (in an hour of simulation, each node generated around 90 and 130 local-streams). Each generated local-stream was composed of a random number of messages between 9 and 100, which were generated using

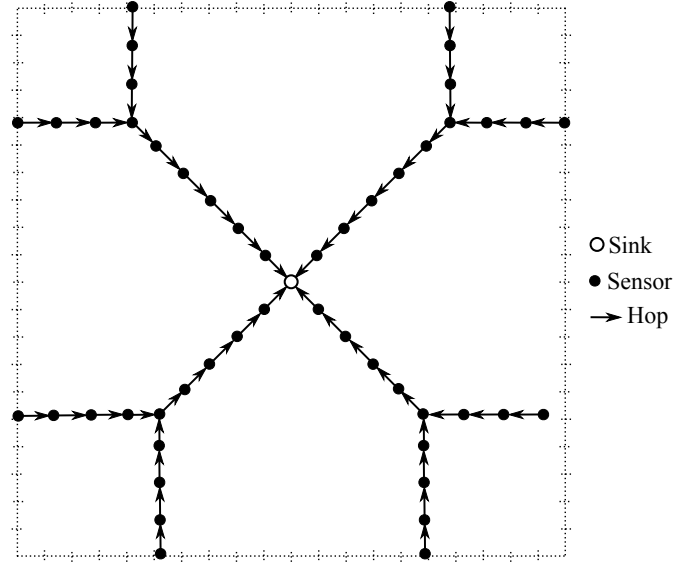


Figure 5.12: Sensor network deployment.

a fixed sampling rate. The simulation scenario was proved with sampling rates between 25 and 1000 ms.

In order to measure and to show that the synchronization error is bounded, we took the simulation time as a global clock. We note that the simulation time was not used in the mechanism. The mechanism used only the causal dependencies between the event-streamings to perform the data alignment and did not use any kind of physical time. For this, at each hop we took the *begin* and *end* causal messages exchanged during the transmissions of the local-streams to determine the synchronization error between a pair of streams.

Let e_α^* denote the send event executed by process p_i to transmit the causal message $m(p_i, \alpha)$ (*begin* or *end*), and let e_ρ^* be the send event executed by process p_j to transmit $m(p_j, \rho)$, which is the latest aligned message before the reception of $m(p_i, \alpha)$; when p_j receives $m(p_i, \alpha)$, the synchronization error of a pair of local-streams is determined by the difference between the sampling time of $m(p_i, \alpha)$ and $m(p_j, \rho)$.

For example, in the scenario depicted in Figure 5.13, process p_i sends the message $m(p_i, \alpha)$ to p_j , which indicates the beginning of the local-stream S_i . Assuming that p_j generates the local-stream S_j , the synchronization error between S_i and S_j is determined by the difference between the sampling time of $m(p_i, \alpha)$ and the sampling time of $m(p_j, \rho)$.

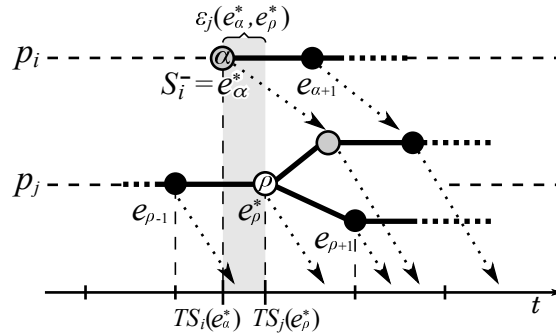


Figure 5.13: Example of the alignment of causal messages.

Thus, by considering the simulation time as a global clock, the synchronization error is estimated by using the following formula:

$$\varepsilon_j(e_\alpha^*, e_\rho^*) = TS_j(e_\rho^*) - TS_i(e_\alpha^*) : e_\alpha^* \in S_i, e_\rho^* \in S_j, e_\alpha^* \downarrow e_\rho^* \quad (5.1)$$

where ε_j is the synchronization error measured at process p_j , and TS_x is a sample time at process p_x . Using sample rates between 25 and 1000 milliseconds, we show that the synchronization error can be bounded according to the average transmission delay (see Figure 5.14).

The simulation results show that the synchronization error among streams is bounded according to the average transmission delay when sample rates are between 75 ms and 830 ms; this fulfills the requirements for the transmission of multimedia data [ITU01].

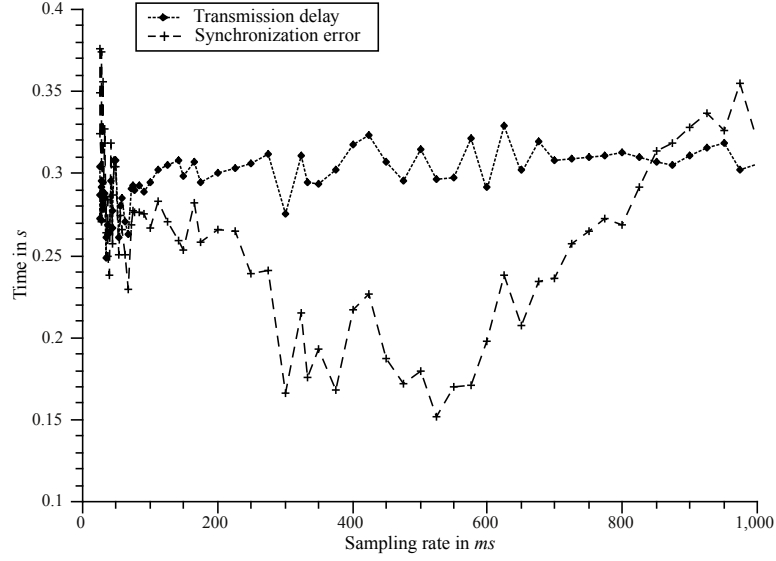


Figure 5.14: Contrast between the synchronization error and the transmission delay.

5.2.4.1 Analysis of the simulation results

The proposed mechanism performs the data alignment in the intermediate nodes while the streams are propagated through the network. In our case, the data alignment is achieved by ensuring that at each intermediate node the synchronization error $\varepsilon_j(e_\alpha^*, e_\rho^*)$ is bounded by the transmission delay for a pair of endpoints e_α^* and e_ρ^* . This means that the execution of the send events of such pair of endpoints takes place at most at $\varepsilon_j(e_\alpha^*, e_\rho^*)$ units of time, one with respect to the another. This result is achieved as follows. When p_j locally constructs an event-streaming, such event-streaming is the output of the alignment of two or more local-streams according to the causal relations established by the mechanism. In such event-streaming, a pair of endpoints (e_α^*, e_ρ^*) is related at p_j , as shown in Figure 5.13. When such endpoints are retransmitted/propagated to another node (final or intermediate), and because they have the same source, their transmission delays will

be affected by the same network conditions, and therefore the synchronization error remains bounded according to the transmission delay of the current hop. This phenomenon is quite similar to the relative velocity between two physical objects.

In Figure 5.14, we can see that the average synchronization error is bounded between sampling rates of 75 ms to 830 ms. The synchronization error cannot be bounded lower than 75 ms because a faster sampling rate causes a greater saturation in communication channels and indeterminism of the transmission delay. The latter is mainly due to the signal-to-noise ratio (SNR), as well as to the medium access contention. On the other hand, the synchronization error cannot be bounded when the sampling rates are greater than 830 ms since the temporal distance between two consecutive local samples at a process p_j is much larger than the maximum transmission delay of the current hop.

5.2.5 Analysis of the mechanism

Storage overhead. To identify and establish the causal relations among streams, the mechanism uses a boolean variable along with the data structures $VT_p[]$, $P_involved_p$, $Causal_m_p$ and ES_CI_p . The size of these structures depends directly on n_r , the number of nodes related to the node that is performing the data alignment. Since the mechanism is designed to perform the data alignment in the intermediate nodes, n_r is determined by the number of hops along the path from the farthest node to the node that performs the data alignment. We note that in a WSN with multi-hop communication, we have that $n_r \ll n$, where n is the number of nodes in the whole network.

Let ς be the number of bytes used to represent an integer value, the storage overhead of a node which performs the data alignment is determined by:

$$SC_p = \varsigma(|VT_p| + |P_involved_p| + |Causal_m_p| + |ES_CI_p|) + 1 \quad (5.2)$$

Assuming the worst case, where all the processes in the node's path are involved with the generation of the event-streaming and a segmentation is performed, we have that $SC_p = \varsigma(6n_r - 1) + 1$, since $|VT_p| = |P_involved_p| = n_r$, $|Causal_m_p| = n_r - 1$ and $|ES_CI_p| = 3n_r$. Therefore, by (5.2) the asymptotic storage overhead is:

$$SC_p \sim O(n_r) \quad (5.3)$$

Communication overhead. Besides the data payload, the mechanism piggybacks control information on each transmitted message. This control information comprises three integers for the processes identifiers and the logical time value, a byte-length variable to identify the message type and the structure H_m . Thereby, the communication overhead is determined by:

$$CO_p = \varsigma(|H_m| + 3) + 1 \quad (5.4)$$

The size of H_m is 0 for the *fifo_p* messages, which is the most frequently message type exchanged. For the causal messages *begin*, *end* and *discrete* H_m depends of n_r . Therefore, by (5.4) the asymptotic communication overhead is:

$$CO_p \sim O(n_r) \quad (5.5)$$

Computational cost. In the mechanism, the major computational cost is related to the segmentation procedure. Within the segmentation procedure a loop, that iterates many times as identifiers are in the structure $P_involved_p$, is used to read the values stored in the structure $VT_p[]$ and construct the immediate history set ES_CI_p . Let τ be the time required to read the value of $VT_p[i]$ (for $0 \leq i \leq |P_involved_p|$) and perform the assignment used to store the entries for the immediate history set ES_CI_p . The time required to perform the segmentation procedure is determined by:

$$CC_p = |P_involved_p|\tau \quad (5.6)$$

Therefore, assuming the worst case where $|P_involved_p| = n_r$, by (5.6) the asymptotic computational cost is:

$$CC_p \sim O(n_r) \quad (5.7)$$

We note that the segmentation procedure is only performed when a *begin* or *end* message is received, the less exchanged message types.

Chapter 6

Temporal data alignment and association for event-streaming

6.1 Spatio-temporal data association approach

As we showed in the previous chapter, by performing the data alignment following the ES-LM model, the synchronization error among streams is bounded according to the average transmission delay between a pair of processes. However, there exist certain cases where such a bound is not sufficient to estimate the original sample time. In environments with multi-hop communication, where each hop induces extra propagation delays, or where not always are all the devices transmitting local-streams, the causal dependencies are not sufficient to establish temporal references according to the real execution of the system.

In order to mitigate such problems, we propose a spatio-temporal association taking advantage of two concepts: the place where the messages were originated and the logical/temporal distance among their transmission events [PCPH14b]. For this, we define a fuzzy-causal relation in such a way that a *degree of closeness* among events can be inferred, considering the information about the spatial and the logical/temporal distance of the events' sources.

6.1.1 The fuzzy-causal relation (FCR)

We formulate the FCR to relate the logical/temporal domain with the spatial domain in the following way:

“how ancient and how far it happened imply how close an event e_1 happened before an event e_2 ”.

To achieve this, we define the following three linguistic variables:

- **Causal distance (CD)**, is the variable whose universe of discourse is the logical/temporal domain. Based on the definition of causal distance proposed by [PHLDRGF09], we define the causal distance as follows.

Definition 17. *The causal distance (CD) is the number of send events involved in the linearization $send_1(m(s, t_s)) \rightarrow send_2(m(i, t_i)) \cdots \rightarrow send_\delta(m(k, t_k))$, such that $send_\delta(m(k, t_k)) \downarrow delivery(m(k, t_k), f)$ with $s \neq i \neq k \neq f$, constructed from the multi-hop transmission of a sample m from its original source s until the final destination f .*

- **Physical distance (PD)**, whose universe of discourse is the spatial domain, refers to the distance between two points in a coordinate system.
- **Fuzzy-causal closeness (FCC)**, whose universe of discourse is the *degree of closeness* among events considering both logical/temporal and spatial domains.

To determine the fuzzy-causal closeness, the causal and the physical distances are related by using a fuzzy inference system (FIS) as explained in the following sections.

6.1.1.1 Fuzzification process

For the Mamdani FIS, we fuzzify the inputs and the outputs using triangular fuzzifiers (see Definition 10). For each linguistic variable, we define five triangular membership functions (fuzzy sets) related to five linguistic terms as follows:

- for the *causal distance*: $VR(e)$ related to *very recent*, $R(e)$ related to *recent*, $MR(e)$ related to *medium recent*, $A(e)$ related to *ancient*, $VA(e)$ related to *very ancient*;
- for the *physical distance*: $VN(e)$ related to *very near*, $N(e)$ related to *near*, $MN(e)$ related to *medium near*, $F(e)$ related to *far*, $VF(e)$ related to *very far*;
- and finally for the *fuzzy-causal closeness*: $VC(e)$ related to *very close*, $N(e)$ related to *close*, $MC(e)$ related to *medium close*, $D(e)$ related to *distant*, and $VD(e)$ related to *very distant*.

The different fuzzy sets are delimited as shown in Table 12.

Table 12: Values of variables used in definition of membership functions.

Universe of discourse	Set	L	C	R
Causal distance	VR	$\sigma_0 - \sigma_2$	σ_0	σ_2
	R	σ_0	σ_2	σ_4
	MR	σ_2	σ_4	σ_6
	A	σ_4	σ_6	σ_8
	VA	σ_6	σ_8	$\sigma_8 + \sigma_2$
Physical distance	VN	$\varepsilon_0 - \varepsilon_2$	ε_0	ε_2
	N	ε_0	ε_2	ε_4
	MN	ε_2	ε_4	ε_6
	F	ε_4	ε_6	ε_8

Continued on next page

Table 12 – continued from previous page

	VF	ε_6	ε_8	$\varepsilon_8 + \varepsilon_2$
Fuzzy-causal closeness	VC	$\varphi_0 - \varphi_2$	φ_0	φ_2
	C	φ_0	φ_2	φ_4
	MC	φ_2	φ_4	φ_6
	D	φ_4	φ_6	φ_8
	VD	φ_6	φ_8	$\varphi_8 + \varphi_2$

The values σ_0 , σ_8 , ε_0 , ε_8 , φ_0 and φ_8 must be chosen using previous knowledge about the network conditions. Thus, σ_0 and σ_8 are determined by the total number of hops; ε_0 and ε_8 are the minimum and the maximum linear distances; while φ_0 and φ_8 can be estimated from the average network delays.

The graphical representations of the fuzzy sets described in Table 12 are shown in Figures 6.1, 6.2 and 6.3.

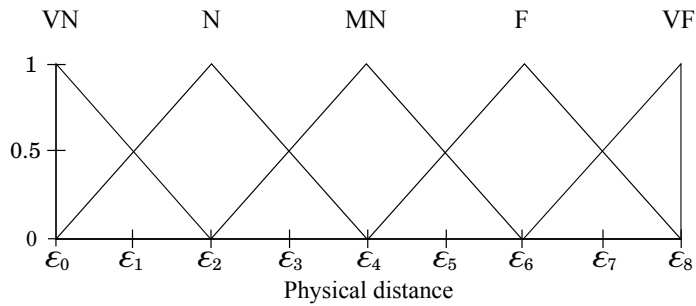


Figure 6.1: Input fuzzy sets for the logical/temporal domain.

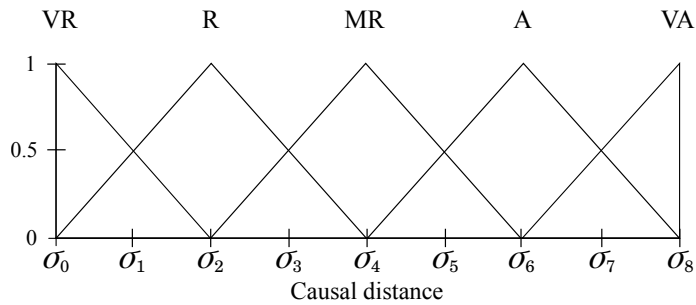


Figure 6.2: Input fuzzy sets for the spatial domain.

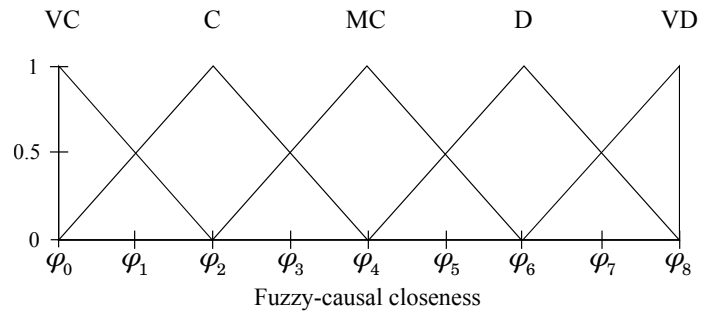


Figure 6.3: Output fuzzy sets

6.1.1.2 Fuzzy inference

Once the inputs and outputs variables were fuzzified, the degree of fuzzy-causal closeness is determined by a Mamdani-type FIS [MA75]. Such a FIS consists of 25 *if-then* rules, that relate the different values of the fuzzy inputs as shown in Table 13.

Table 13: Inference rules

- 1) If PD is VN and CD is VR, then fuzzy-causal closeness is VC
- 2) If PD is VN and CD is R, then fuzzy-causal closeness is C
- 3) If PD is VN and CD is MR, then fuzzy-causal closeness is C
- 4) If PD is N and CD is VR, then fuzzy-causal closeness is C
- 5) If PD is N and CD is R, then fuzzy-causal closeness is C
- 6) If PD is MN and CD is VR, then fuzzy-causal closeness C
- 7) If PD is MN and CD is R, then fuzzy-causal closeness is C
- 8) If PD is F and CD is VR, then fuzzy-causal closeness C

Continued on next page

Table 13 – continued from previous page

9)	If PD is VN and CD is VA, then fuzzy-causal closeness is MC
10)	If PD is VN and CD is A, then fuzzy-causal closeness is MC
11)	If PD is N and CD is MR, then fuzzy-causal closeness is MC
12)	If PD is N and CD is A, then fuzzy-causal closeness is MC
13)	If PD is MN and CD is MR, then fuzzy-causal closeness is MC
14)	If PD is MN and CD is A, then fuzzy-causal closeness is MC
15)	If PD is F and CD is MR, then fuzzy-causal closeness is MC
16)	If PD is F and CD is R, then fuzzy-causal closeness is MC
17)	If PD is N and CD is VA, then fuzzy-causal closeness is D
18)	If PD is MN and CD is VA, then fuzzy-causal closeness is D
19)	If PD is F and CD is A, then fuzzy-causal closeness is D
20)	If PD is F and CD is VA, then fuzzy-causal closeness is D
21)	If PD is VF and CD is VR, then fuzzy-causal closeness MC
22)	If PD is VF and CD is R, then fuzzy-causal closeness is MC
23)	If PD is VF and CD is MR, then fuzzy-causal closeness is D
24)	If PD is VF and CD is A, then fuzzy-causal closeness is D
25)	If PD is VF and CD is VA, then fuzzy-causal closeness is VD

The outputs of the inference system are fuzzy output variables. For this, it is necessary to convert the fuzzy output variables into crisp values, through the de-

fuzzification process. Defuzzification can be performed in several ways. We choose the Weighted Average method (see Definition 11) since it has a low computational cost and we have defined symmetrical membership functions. Therefore, the overall input-output surface corresponding to the above membership functions, values of variables, and rules is depicted in Figure 6.4.

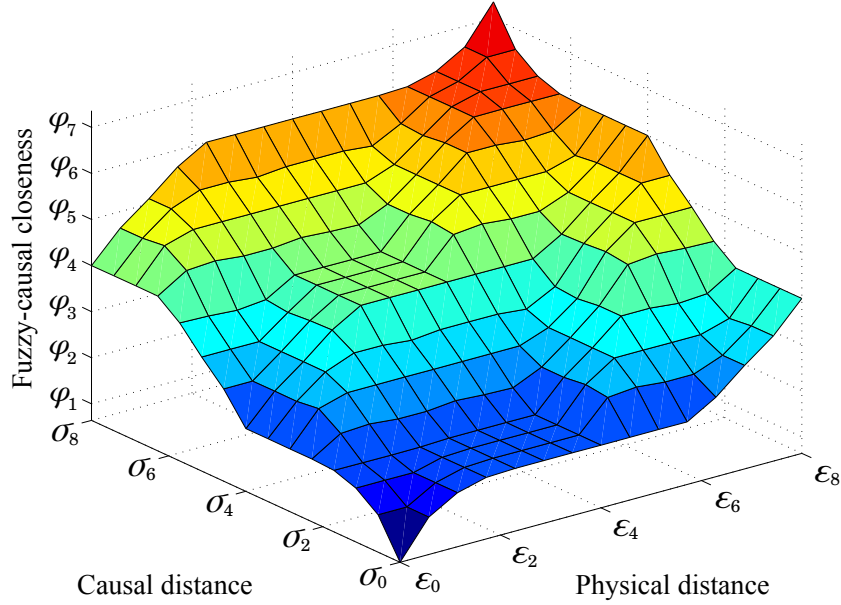


Figure 6.4: Input-output surface corresponding to the membership values of PD, CD and FCC.

6.1.1.3 Fuzzy Causal Relation formal definition

Considering the inferred FCC value, we formally define the FCR (denoted by $\xrightarrow{\lambda}$) as follows:

Definition 18. *The FCR over a set of events must satisfy:*

1. $a \xrightarrow{\lambda} b$ If $a \rightarrow b$ and $0 < FCC < \varphi_8$
2. $a \xrightarrow{\lambda} b$ If $\exists c \mid a \xrightarrow{\lambda} c \xrightarrow{\lambda} b$ and $0 < FCC < \varphi_8$,

where *FCC* is the degree of fuzzy-causal closeness between a and b .

From this definition we have that for a pair of causally-related events we can estimate *how long ago* an event happened with respect the other, by using the physical location of their sources, and the transmission events involved to their final delivery.

Based on this principle, in the following section we describe how the data alignment, performed through the ES-LM model, can be refined by incorporating the notion of the fuzzy-causal relation.

6.2 Temporal data alignment and association mechanism

The proposed mechanism is an extension of the data alignment mechanism described in Section 5.2. Such extension, incorporates the Mamdani FIS (described in Section 6.1.1.2) to the data alignment mechanism in order to establish how long ago the events, of a received stream, were generated.

6.2.1 Mechanism specification

In the same way that described in section 5.2, the mechanism uses three different causal messages: *begin*, *end* and *discrete*, and a type of FIFO messages: *fifo_p*. However, one new field has to be added to the general format of the messages. Such new field is used to exchange the information used by the FIS. Thus, a message m is a tuple $m = (k, sts, t_{sts}, TP, H_m, DI, data)$, where:

- k is the local process identifier.
- sts is the identifier of the process that originally generated the message of an event-streaming.

- t_{sts} is the value of the local clock, in the original source.
- TP is the message type (*begin*, *end*, *discrete* and *fifo_p*).
- H_m , the immediate history of m , contains the identifiers of the messages (k, sts, t_{sts}) that immediately precede m . For *fifo_p* the set H_m is always the empty set.
- DI is a tuple $(loc(x, y), cdist)$, where $loc(x, y)$ are the coordinates of the physical location of p_{sts} and $cdist$ is the *causal distance* associated to m . For *end*, and *fifo_p* the set DI is always the empty set.
- $data$ is the structure that carries the media data.

Respect to the data structures that each process has to store, only one new structure is added. Thereby, process k has to store the following data structures:

- $VT_p[]$ is the vector clock established by Mattern and Fidge [Mat88, Fid88]. The size of $VT_p[]$ is equal to the number of nodes that are associated with a node which performs the data alignment.
- $P_involved_p$ is a set with the identifiers of the processes that originated the messages that compose an event-streaming.
- $Causal_m_p$ is a set with the identifiers of the processes that generated the last aligned subset $Q_q^{R_q}$.
- ES_CI_p is a set composed by entries (k, sts, t_{sts}) used to store the immediate history that will be piggybacked in certain messages.
- ES_Act is a state variable that indicates when a process is aligning an event-streaming.

- $FCC[]$ is a vector which stores the fuzzy-causal degree associated to certain streams.

The data alignment and association mechanism is specified by six algorithms: two algorithms for managing the control information (Algorithms 6 and 7), an algorithm to infer the fuzzy-causal closeness by performing the Mamdani FIS (Algorithm 8), and three algorithms for the data exchange (Algorithms 9, 10 and 11).

Algorithm 6: Initialization

```

1: procedure INITIALIZE()
2:    $VT_p[k] \leftarrow 0, \forall k : 1..n$ 
3:    $FCC[k] \leftarrow 0, \forall k : 1..n$ 
4:    $P\_involved_p \leftarrow \emptyset$ 
5:    $Causal\_m_p \leftarrow \emptyset$ 
6:    $ES\_CI_p \leftarrow \emptyset$ 
7:    $ES\_Act \leftarrow 0$ 
8:   define:  $d\_required = \{true|false\}$ 
9:   define:  $l_k^{pd}, \bar{a}_{c_k}^{pd}, r_k^{pd} \forall k : 1..5$            /* definition of the 5 triangular*/
10:  define:  $l_k^{cd}, \bar{a}_{c_k}^{cd}, r_k^{cd} \forall k : 1..5$          /* membership functions for */
11:  define:  $l_k^{cc}, \bar{a}_{c_k}^{cc}, r_k^{cc} \forall k : 1..5$        /* each linguistic variable*/
12: end procedure

```

Algorithm 7 performs the operations required to construct the immediate history sets, that are piggybacked in certain messages, in order to establish a segmentation of a local-stream or an event-streaming.

Algorithm 7: Construction of the immediate history sets

```

13: procedure SEGMENTING_ES( $TP = \{begin|end|discrete\}$ )
14:   if  $Causal\_m_p \neq \emptyset$  then
15:      $ES\_CI_p \leftarrow \emptyset$ 
16:     for all  $x \in P\_involved_p$  do
17:        $ES\_CI_p \leftarrow ES\_CI_p \cup \{(i, x, VT_p[x])\}$ 
18:     end for
19:   end if
20: end procedure

```

Algorithm 8 performs the Mamdani FIS to infer the fuzzy-causal closeness degree, by using the exchanged information about the physical location of the original source of the received stream and the causal distance associated to the received messages.

Algorithm 8: Fuzzy-causal closeness computation

```

21: procedure MAMDANIFIS( $x1, x2, y1, y2, cdist, sts$ )
22:   for  $i \leftarrow 1$  to  $i = 5$  do
23:      $pd_i \leftarrow \max(\min(\frac{\sqrt{(x1-x2)^2+(y1-y2)^2} - l_i^{pd}}{\bar{a}_{c_i}^{pd} - l_i^{pd}}, \frac{r_i^{pd} - \sqrt{(x1-x2)^2+(y1-y2)^2}}{r_i^{pd} - \bar{a}_{c_i}^{pd}}), 0)$ 
24:      $cd_i \leftarrow \max(\min(\frac{cdist-l_i^{cd}}{\bar{a}_{c_i}^{cd}-l_i^{cd}}, \frac{r_i^{cd}-cdist}{r_i^{cd}-\bar{a}_{c_i}^{cd}}), 0)$ 
25:   end for
   /* If-then rules, evaluated by using the Mamdani fuzzy implication: */
26:    $f_{cc1} \leftarrow \min(pd_1, cd_1)$     $f_{cc2} \leftarrow \min(pd_1, cd_2)$     $f_{cc3} \leftarrow \min(pd_1, cd_3)$ 
27:    $f_{cc4} \leftarrow \min(pd_2, cd_1)$     $f_{cc5} \leftarrow \min(pd_2, cd_2)$     $f_{cc6} \leftarrow \min(pd_3, cd_1)$ 
28:    $f_{cc7} \leftarrow \min(pd_3, cd_2)$     $f_{cc8} \leftarrow \min(pd_4, cd_1)$     $f_{cc9} \leftarrow \min(pd_1, cd_5)$ 
29:    $f_{cc10} \leftarrow \min(pd_1, cd_4)$     $f_{cc11} \leftarrow \min(pd_2, cd_3)$     $f_{cc12} \leftarrow \min(pd_2, cd_4)$ 
30:    $f_{cc13} \leftarrow \min(pd_3, cd_3)$     $f_{cc14} \leftarrow \min(pd_3, cd_4)$     $f_{cc15} \leftarrow \min(pd_4, cd_3)$ 
31:    $f_{cc16} \leftarrow \min(pd_4, cd_2)$     $f_{cc17} \leftarrow \min(pd_5, cd_1)$     $f_{cc18} \leftarrow \min(pd_5, cd_2)$ 
32:    $f_{cc19} \leftarrow \min(pd_2, cd_5)$     $f_{cc20} \leftarrow \min(pd_3, cd_5)$     $f_{cc21} \leftarrow \min(pd_4, cd_4)$ 
33:    $f_{cc22} \leftarrow \min(pd_4, cd_5)$     $f_{cc23} \leftarrow \min(pd_5, cd_3)$     $f_{cc24} \leftarrow \min(pd_5, cd_4)$ 
34:    $f_{cc25} \leftarrow \min(pd_5, cd_5)$ 
35:    $FCC[sts] \leftarrow \frac{f_{cc1} \cdot \bar{a}_{c1}^{cc} + (\sum_{i=2}^8 f_{cc_i} \cdot \bar{a}_{c2}^{cc}) + (\sum_{i=9}^{18} f_{cc_i} \cdot \bar{a}_{c3}^{cc}) + (\sum_{i=19}^{24} f_{cc_i} \cdot \bar{a}_{c4}^{cc}) + f_{cc25} \cdot c_5^{cc}}{\sum_{i=1}^{25} f_{cc_i}}$ 
36: end procedure

```

Algorithm 9 performs the operations required to send *begin*, *end*, and *fifo_p* messages.

Algorithm 9: Sending of messages, performed by a process i

```

37: procedure SENDCONTINUOUS( $TP = \{begin|end|fifo\_p\}$ )
38:    $VT_p[i] \leftarrow VT_p[i] + 1$ 
39:   if  $TP = begin$  then
40:     if  $ES\_Act \neq 0$  then                                     /* if process  $i$  is generating a local-stream */
41:       SEGMENTING_ES( $TP$ )                                       /* or is aligning an event-streaming, */
42:     else                                                         /* it performs a segmentation */

```

Continued on next page

Algorithm 9 – continued from previous page

```

43:   |  $ES\_Act \leftarrow 1$ 
44:   end if
45:    $Causal\_m_p \leftarrow P\_involved_p$ 
46:    $P\_involved_p \leftarrow P\_involved_p \cup \{i\}$ 
47:    $SENDING(i, i, VT_p[i], TP, ES\_CI_p, (loc(x_i, y_i), 1), data)$ 
48: else if  $TP = end$  then
49:   |  $H_m \leftarrow ES\_CI_p$ 
50:   if  $ES\_Act \neq 0$  then
51:   |  $SEGMENTING\_ES(TP)$ 
52:   end if
53:    $P\_involved_p \leftarrow P\_involved_p - \{i\}$ 
54:    $Causal\_m_p \leftarrow P\_involved_p$ 
55:   if  $P\_involved_p = \emptyset$  then
56:   |  $ES\_Act \leftarrow 0$ 
57:   end if
58:    $SENDING(i, i, VT_p[i], TP, H_m, \emptyset, data)$ 
59: else
60:   if  $i \in Causal\_m_p$  then                                     /* the first messages after a */
61:   |  $Causal\_m_p \leftarrow Causal\_m_p - \{i\}$                                /* segmentation are sent */
62:   |  $SENDING(i, i, VT_p[i], begin, ES\_CI_p, (loc(x_i, y_i), 1), data)$  /* as begin */
63:   | if  $Causal\_m_p = \emptyset$  then
64:   | |  $ES\_CI_p \leftarrow \emptyset$ 
65:   | end if
66:   else
67:   |  $SENDING(i, i, VT_p[i], TP, \emptyset, \emptyset, data)$ 
68:   end if
69: end if
70: end procedure

```

Algorithm 10 performs the operations required to send *discrete* messages.

Algorithm 10: Sending of *discrete* messages, performed by a process i

```

71: procedure SENDDISCRETE( $TP = \{discrete\}$ )
72:   |  $VT_p[i] \leftarrow VT_p[i] + 1$ 
73:   if  $ES\_Act \neq 0$  then
74:   |  $SEGMENTING\_ES(TP)$ 
75:   |  $Causal\_m_p \leftarrow P\_involved_p$ 

```

Continued on next page

Algorithm 10 – continued from previous page

```

76:   |   |  $H_m \leftarrow ES\_CI_p$ 
77:   |   |  $ES\_CI_p \leftarrow \{(i, i, VT_p[i])\}$ 
78:   |   | else
79:   |   |   |  $H_m \leftarrow \emptyset$ 
80:   |   |   |  $ES\_CI_p \leftarrow \{(i, i, VT_p[i])\}$ 
81:   |   | end if
82:   |   |  $SENDING(i, i, VT_p[i], TP, H_m, (loc(x_i, y_i), 1), data)$ 
83:   | end procedure

```

Algorithm 11 performs the operations required to retransmit or deliver the received packets.

Algorithm 11: Reception of messages, performed by a process j

```

84: procedure RECEIVE( $m = (i, sts, t_{sts}, TP, H_m, data)$ )
85:   | if  $t_{sts} = VT_p[sts] + 1$  then                               /* FIFO delivery condition */
86:   |   | if  $TP \neq fifo\_p$  then
87:   |   |   | if  $t' \leq VT_p[l] \ \forall (u, l, t') \in H_m$  then       /* causal delivery condition */
88:   |   |   |   |  $VT_p[sts] \leftarrow VT_p[sts] + 1$ 
89:   |   |   |   | if  $TP = begin$  then
90:   |   |   |   |   | if  $sts \notin Causal\_m_p$  then
91:   |   |   |   |   |   | if  $ES\_Act \neq 0$  then
92:   |   |   |   |   |   |   |  $SEGMENTING\_ES(begin)$ 
93:   |   |   |   |   |   |   |  $Causal\_m_p \leftarrow P\_involved_p$ 
94:   |   |   |   |   |   | else
95:   |   |   |   |   |   |   |  $ES\_Act \leftarrow 1$ 
96:   |   |   |   |   |   | end if
97:   |   |   |   |   |   |  $P\_involved_p \leftarrow P\_involved_p \cup \{sts\}$ 
98:   |   |   |   |   | else
99:   |   |   |   |   |   |  $Causal\_m_p \leftarrow Causal\_m_p - \{sts\}$ 
100:  |   |   |   |   | end if
101:  |   |   |   | if  $d\_required = true$  then
102:  |   |   |   |   |  $MAMDANIFIS(x_j, x_{sts}, y_j, y_{sts}, cdist, sts)$ 
103:  |   |   |   |   |  $DELIVERY((j, sts, t_{sts}, TP, ES\_CI_p, data), FCC[sts])$ 
104:  |   |   |   | else
105:  |   |   |   |   |  $SENDING(j, sts, t_{sts}, TP, ES\_CI_p, (loc(x_{sts}, y_{sts}), cdist + 1), data)$ 
106:  |   |   |   | end if
107:  |   |   | else if  $TP = end$  then

```

Continued on next page

Algorithm 11 – continued from previous page

```

108:    $H_m \leftarrow ES\_CI_p$ 
109:   if  $|P\_involved_p| > 1$  then
110:     | SEGMENTING_ES( $TP$ )
111:   else
112:     |  $ES\_CI_p \leftarrow \emptyset$ 
113:     |  $ES\_Act \leftarrow 0$ 
114:   end if
115:   if  $d\_required = true$  then
116:     | DELIVERY( $(j, sts, t_{sts}, TP, H_m, \emptyset, data), FCC[sts]$ )
117:   else
118:     | SENDING( $j, sts, t_{sts}, TP, H_m, \emptyset, data$ )
119:   end if
120:    $P\_involved_p \leftarrow P\_involved_p - \{sts\}$ 
121:   if  $Causal\_m_p = \emptyset$  then
122:     |  $Causal\_m_p \leftarrow P\_involved_p$ 
123:   else
124:     |  $Causal\_m_p \leftarrow Causal\_m_p - \{sts\}$ 
125:   end if
126:   else if  $TP = discrete$  then
127:     | if  $ES\_Act \neq 0$  then
128:       | SEGMENTING_ES( $TP$ )
129:       |  $Causal\_m_p \leftarrow P\_involved_p$ 
130:       |  $H_m \leftarrow ES\_CI_p$ 
131:       |  $ES\_CI_p \leftarrow \{(j, sts, t_{sts})\}$ 
132:     | else
133:       |  $H_m \leftarrow \emptyset$ 
134:     | end if
135:     | if  $d\_required = true$  then
136:       | MAMDANIFIS( $x_j, x_{sts}, y_j, y_{sts}, cdist, sts$ )
137:       | DELIVERY( $(j, sts, t_{sts}, TP, H_m, data), FCC[sts]$ )
138:     | else
139:       | SENDING( $j, sts, t_{sts}, TP, H_m, (loc(x_{sts}, y_{sts}), cdist + 1), data$ )
140:     | end if
141:   | end if
142:   else
143:     | WAIT()
144:   end if

```

Continued on next page

Algorithm 11 – continued from previous page

```

145:   else if ( $sts \in P\_involved_p$ ) then
146:      $VT_p[sts] \leftarrow VT_p[sts] + 1$ 
147:     if  $sts \in Causal\_m_p$  then
148:       if  $d\_required = true$  then
149:          $DELIVERY((j, sts, t_{sts}, begin, ES\_CI_p, data), FCC[sts])$ 
150:       else
151:          $SENDING(j, sts, t_{sts}, begin, ES\_CI_p, (loc(x_{sts}, y_{sts}), cdist + 1), data)$ 
152:       end if
153:        $Causal\_m_p \leftarrow Causal\_m_p - \{sts\}$ 
154:       if  $Causal\_m_p = \emptyset$  then
155:          $ES\_CI_p \leftarrow \emptyset$ 
156:       end if
157:     else
158:       if  $d\_required = true$  then
159:          $DELIVERY((j, sts, t_{sts}, TP, H_m, data), FCC[sts])$ 
160:       else
161:          $SENDING(j, sts, t_{sts}, TP, H_m, \emptyset, data)$ 
162:       end if
163:     end if
164:   end if
165: else
166:    $WAIT\_FIFO()$ 
167: end if
168: end procedure

```

6.2.2 Functional description of the mechanism

The data alignment and association is performed by taking advantage of the multiple retransmissions of a multi-hop communication. This implies that the data alignment and association take place during the execution of the send, receive and delivery events.

Sending of messages A process p_i generates a local-stream by sequentially sending a certain number of messages. All the sent messages must be time-

stamped by setting the field t_{sts} with the current value of the local vector clock. Besides the time-stamp, the *begin* messages, the first of the sequence, must be stamped with the physical location of the process by setting the corresponding values of $loc(x, y)$ and setting the field $cdist = 1$ (see lines 47 and 62 of Algorithm 9). As explained in section 5.2.2, there are some local-streams that are composed by only one message. For such cases, the emission is performed by sending a *discrete* message. As the *begin* messages, the discrete messages must be stamped with the physical location of the process by setting the corresponding values of $loc(x, y)$ and setting the field $cdist = 1$ (see line 82 of Algorithm 10).

Reception of messages Any process p_j can receive the messages of a local-stream or an event-streaming from another process before or during the generation of its own local-stream. Whatever the case, as soon as an overlap between the local-streams is detected, an event-streaming is generated by performing the necessary segmentations according to Cases A.1, B.1, B.2 and B.3 of the ES-LM (see Section 5.1.1).

A process p_j can deliver the received messages to the application or forward it to another process (using a send event). When a process p_j receives a *begin* or a *discrete* message and p_j is acting as a relay node, it forwards such a message increasing by one the value of the field $cdist$ (see lines 105, 139 and 151 of Algorithm 11).

Delivery of messages If p_j needs to deliver the *begin* or the *discrete* message to the application, it calculates the fuzzy-causal closeness of such a message by using the Mamdani FIS (see lines 102 and 136). By defuzzifying the output of the Mamdani FIS, the process p_j establishes the degree of fuzzy-causal closeness of

the whole stream that has begun to receive. Thereby, it delivers a causal message, associated with a fuzzy-causal degree to the application.

We explain the previous procedures with the following example. Consider that there are three processes p_i , p_j and p_k in a WSN with a mult-hop communication, where the maximum number of hops is 8 and the maximum physical distance is 100. p_i (with coordinates (60, 15)) can communicate with p_k (with coordinates (17, 40)) through p_j . At a certain time t_i , p_i starts the transmission of a local-stream S_i , as depicted in Figure 6.5.

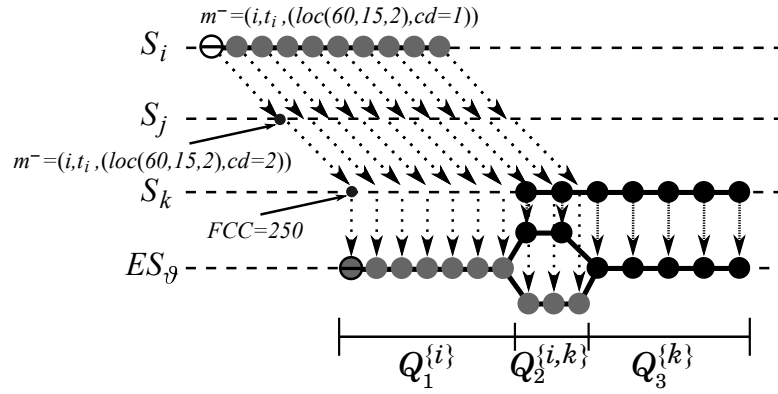


Figure 6.5: Alignment and association of two local-streams at p_k .

When p_j receives the *begin* message of S_i , it simply forwards such a message to p_k , setting $cdist = 2$. When p_k receives *begin* message, it determines the fuzzy-causal closeness of such a message. By fuzzifying the values of $cdist$ and the physical distance PD , the respective degrees of membership to the fuzzy sets related to the physical and causal distances are obtained as shown in Figure 6.6.

By using the Mamdani FIS with the fuzzy variables for the values PD and $cdist$, the fuzzy causal-closeness is determined by the degree of membership to the fuzzy set C , as shown in Figure 6.7.

For this example we set the scalar values of the FCC between 0 and 1000, however, such scale can be adjusted according to the known transmission delays

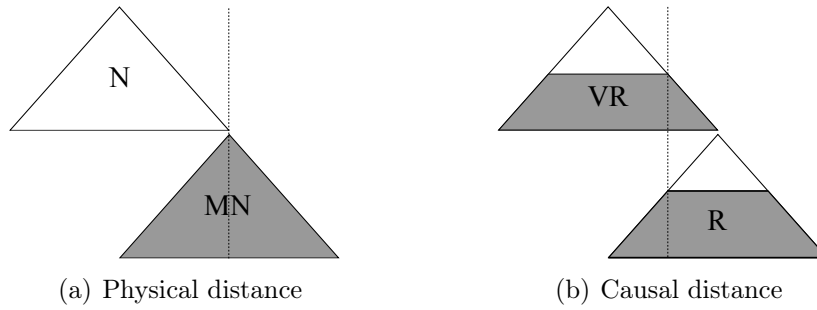


Figure 6.6: Membership degrees for $PD = 49.74$ and $cdist = 2$.

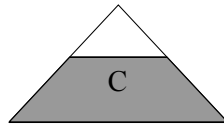


Figure 6.7: Degree of membership to the set C for the FIS output.

depending on the network conditions. Therefore, with the WA defuzzification method, we determine that the value of the FCC is 250; in terms of the temporal domain this can be interpreted by p_k , that the messages of S_i was generated 250 units of physical time ago. With this information, the process p_k delivers the messages of the stream S_i along with a closeness degree to the application. In this case, such closeness degree determines that the streams S_i and S_k are not completely overlapped. We note that for this case, based only on causal dependencies, the alignment of the two local-streams cannot be established according to the real execution, since with causal dependencies we cannot establish the time in which an event was generated before another process.

6.2.3 Discussion

As we previously argued, the FIS used to determine the fuzzy-causal closeness, like all the fuzzy systems, is configured using previous knowledge about the system properties and conditions. In this sense, the values that describe the fuzzy sets

related to the fuzzy-causal closeness can be configured by considering the average network delays between nodes.

We have simulated the temporal data alignment and association mechanism for event-streaming using the Castalia simulator [Bou13]. We took the same network configuration depicted in Section 5.2.4, which consists of an arrangement of 50 nodes, separated by distances between 10 and 15 meters in a field of 200 x 200 meters, with a multi-hop communication. With such a configuration, the input crisp values to delimit the fuzzy sets related to the causal distance (CD) are between 1 and 9, that are the maximum number of hops. On the other hand, the values for the physical distance are between 17.97 and 152.73 meters. By adopting the same interference model depicted in Section 5.2.4, we obtained an average transmission delay of 0.3045 seconds between a pair of nodes. Based on such average delay, the values for the fuzzy-causal closeness were fixed between -0.3045 and 3.3492.

In the same manner as in the simulation described in Section 5.2.4, each node generated a random number of local-streams throughout the simulation (in an hour of simulation, each node generated around 90 and 130 local-streams). Each generated local-stream was composed of a random number of messages between 9 and 100, which were generated using a fixed sampling rate. The simulation scenario was proved with sampling rates between 25 and 1000 ms.

In order to show how the fuzzy-causal closeness can be used to estimate how long ago an event happened, we took the simulation time as a global clock to compare the sampling time of an event e , registered at its source (process p_i), and the estimated sampling time, computed from the FCC value and inferred in the receptor process (process p_j). Let $ET_j(e)$ be the estimated sampling time of an event e , computed in process p_j , and let $ST_i(e)$ be the sampling time of the event

e in process p_i . For a message $m(p_i, \alpha)$ the estimated sampling time is computed as follows:

$$ET_j(\text{send}(m(p_i, \alpha))) = ST_j(\text{receive}(p_j, m(p_i, \alpha))) - FCC. \quad (6.1)$$

Since the FCC value is an estimation of the transmission duration of $m(p_i, \alpha)$, from p_i to p_j (see Figure 6.8), we determine the error of such estimation in the following way:

$$\varepsilon(FCC) = |ST_i(\text{send}(m(p_i, \alpha))) - ET_j(\text{send}(m(p_i, \alpha)))|. \quad (6.2)$$

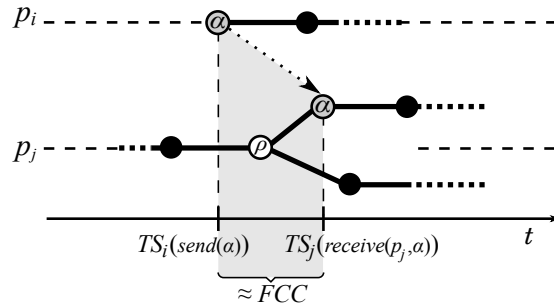


Figure 6.8: Example of the alignment of causal messages.

By simulating the mechanism with sampling rates between 25 and 1000 ms, and using the equation 6.2 to compute the error $\varepsilon(FCC)$, between the estimated sampling time and the real sampling time of the events involved in the transmission of each *begin* message; we obtained that $\varepsilon(FCC)$ is always under the average transmission delay as shown in Figure 6.9.

As can be noticed in Figure 6.9, unlike the synchronization error $\varepsilon_j(e_\alpha^*, e_\rho^*)$, which establishes a temporal distance between a pair of endpoints of two aligned streams (see Section 5.2.4); the error $\varepsilon(FCC)$, of the temporal distance inferred

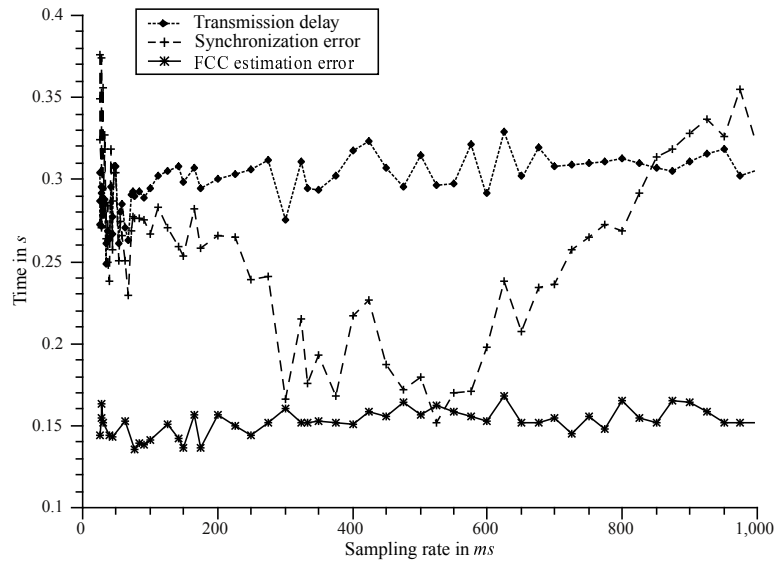


Figure 6.9: Contrast among the error of the estimated sampling time, established by $\varepsilon(FCC)$; the synchronization error of the simple temporal data alignment $\varepsilon_j(e_\alpha^*, e_\rho^*)$, and the transmission delay.

from the FCC, remains under the average transmission delay. In this way, the spatio-temporal association mechanism can be used to refine the virtual time-line established by the simple temporal data alignment.

Chapter 7

Conclusions and future work

7.1 Conclusions

7.1.1 Summary

In this dissertation, a distributed data alignment and association mechanism for the event-streaming in wireless sensor networks has been presented. This mechanism has three main characteristics:

- a) the data streams generated by the various sensors of a network are aligned according to a virtual time-line specified through the causal dependencies among streams, without requiring a global physical clock;
- b) the alignment is performed in the intermediate nodes while the streams are propagated through the network, by considering a multi-hop communication;
- c) by using the information about the physical location of the sources and the causal information of the streams, any node can estimate how long ago an event happened before another event.

In the course of this research, we found that the event-streaming can be represented as a partially ordered set of disjoint subset of events generated by multiple

sources. Based on such representation, we defined the event-streaming as an abstract data type (ADT).

From the event-streaming ADT, we found that if the subsets of events of an event-streaming are causally ordered and arranged, one after another without interruption, a virtual time line is established, where each subset of events represents a unique time-slot. Considering this fact, we designed a new model for data alignment, called Event-Streaming Logical Mapping (ES-LM). With the ES-LM model, the data alignment is performed by constructing an event-streaming as a causal arrangement of subset of events, in such a way that each event must be located in an unique time-slot, without requiring global references or synchronized physical clocks. Thus, the ES-LM model is the main contribution of this work.

Another important finding is that by relating the logical/temporal domain (determined by the causal order) with the spatial domain, a degree of “closeness” between events can be inferred and it can be established “how long ago” an event happened before another. For this, we defined a fuzzy-causal relation that relates the spatial and logical/temporal domains and infer a degree of “causal closeness” between events. By including the fuzzy-causal relation to the data alignment based on the ES-LM, a degree of “causal closeness” among streams can be established. In this way, the definition of the fuzzy-causal relation is a secondary contribution of this work.

7.1.2 Achievements

During the development of this research, some original and innovative findings were reached. These findings can be useful to solve some open problems related to distributed systems, as we explain below.

- **The usefulness of the event-streaming abstract data type.** There are several distributed applications, such as event stream processing or global predicate detection, where it is necessary to detect some temporal patterns from the various streams exchanged by multiple sources. To detect such temporal patterns, existing solutions recursively examine all the exchanged streams pair by pair. This last requires that all the exchanged streams be stored to perform an outline examination; or that all the involved sources collectively examine the streams using pairwise interactions among them. Whichever of those two schemes increase the requirements of storage, time or communication overheads.

We defined the event-streaming as an abstract data type, based on a causal data structure, which inherently holds all the temporal patterns detected among various examined streams. In this way, when an event-streaming is analyzed along another stream, temporal patterns among several streams can be detected, instead only detect patterns between a pair of streams. In this way, the detection of the temporal patterns can be performed on the fly without store all the involved streams, while the number of pairwise interactions is reduced, decreasing the communication overhead.

- **The usefulness of the fuzzy-causal relation.** Causal ordering is an important research subject in distributed systems since allows to provide reliability for some applications, preserving the asynchronous execution of the system. However, for certain applications, for example multimedia synchronization, where some degradation of the system is allowed, ensuring the causal order based on the happened-before relation is rigid, and negative affects the performance of the system.

In this dissertation we proposed a fuzzy-causal relation with the aim to infer a degree of causal closeness between events. In this sense, the fuzzy-causal closeness can be used to determine if the delivery of certain messages is mandatory or such messages can be discarded. Thereby, by applying a delivery criteria, based on the fuzzy-causal closeness, the asynchrony of a system can be increased, improving its performance.

7.1.3 Limitations

The mechanisms proposed in this dissertation were designed to achieve certain objectives. For this reason, the mechanisms have the two limitations explained below.

- **The linear growth of the causal structures.** For the development of the data alignment and association mechanism, the well-known vector clock structure [Mat88, Fid88] was used to identify and preserve the causal relations among events. Such a structure is characterized by its linear growth, bounded by the number of nodes that are associated with the entity which executes the mechanism. In order to control the growth of such a structure, and make an appropriate use of the network and computational resources, a WSN can be organized following a clustering technique (for example the LEACH protocol [HCB00]). However, the development or the implementation of clustering techniques for WSN is out of the scope of this work and was left as an open topic.
- **The lack of packet loss tolerance.** The development of the mechanism was based on a specific system model which considered reliable communication channels. For this reason, we do not implemented techniques to support

the packet loss. To support the packet loss, it is necessary to develop or implement methods for the detection and the recovery of lost messages (for example forward error correction). Nevertheless, the packet loss tolerance was not part of the objectives of this dissertation.

7.1.4 Dissertation-derived articles

- Jose Roberto Perez Cruz, Saul Eduardo Pomares Hernandez, and Enrique Munoz de Cote. Data alignment for data fusion in wireless multimedia sensor networks based on M2M. *KSII Transactions on Internet and Information Systems*, 6(1):229–240, 2012.
- Jose Roberto Perez Cruz and Saul E. Pomares Hernandez. Temporal data alignment and association for event-streaming in ubiquitous environments based on fuzzy-causal dependencies. *In 2014 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, pages 127–134. IEEE Computer Society, February 2014.
- Jose Roberto Perez Cruz and Saul E. Pomares Hernandez. Temporal alignment model for data streams in wireless sensor networks based on causal dependencies. *International Journal of Distributed Sensor Networks*, 2014:1–11, 2014.

7.2 Future work

During the development of this research, it was observed that the proposed theory can be extended in some directions in order to solve some open problems in some kinds of distributed systems. Thereby, the future directions of this work include:

the development of novel solutions for intermedia synchronization in multimedia distributed systems, data alignment and association in mobile ad-hoc networks, and event-streaming processing for the self-managing in Machine-to-Machine environments (M2M). A more detailed description of the future directions of this work is presented below.

- **Distributed intermedia synchronization.** The native Logical Mapping Model [PEMR08], designed to perform intermedia synchronization in multimedia distributed systems, is designed to determine synchronization points between a pair of streams. In some environments where streams are generated and exchanged among several sources, performing the synchronization pair by pair can be highly expensive, since requires pairwise interactions among the all processes that generate the streams.

The ES-LM model can be adapted to perform intermedia synchronization to allow that a single stream can be synchronized with several streams. By the form that the ES-LM was designed, the endpoints of the subsets of events that compose an event-streaming can determine the synchronization points for several streams. In this way, a pair of processes can exchange collections of synchronization points, avoiding the interactions with all the related sources.

- **Establishment of a delivery order for concurrent events.** In some emerging applications, such as interactive and collaborative systems like *haptic*¹ teleoperations, in order to ensure a consistent system view, the users must see, hear and/or feel the actions (events) as they were originally ex-

¹Haptic technology, or haptics, is a tactile feedback technology which takes advantage of the sense of touch by applying forces, vibrations, or motions to the user, “doing for the sense of touch what computer graphics does for vision” [RDLT09].

ecuted. For this reason, such systems must be reactive to stimulus which result in consecutive series of concatenated cause-effect events, either continuous and/or discrete.

For such applications, the distributed ordering of events, oriented to intermedia synchronization or data alignment, requires to identify and to represent cause-effect dependencies among multimedia data, besides to only represent multimedia elemental temporal relations based on causal dependencies. Under such requirements, the concurrency among events not only must be treated as the opposite of the causal relation, but it may be treated as a partial affection among events that co-occur at an instant of time. This last in order to reduce the uncertainty of the system.

In this sense, the research could be focused on how the fuzzy-causal relation, defined in this work, can be extended to determine certain order among the concurrent events. Based on the concept of fuzzy-causal closeness, it can be inferred a degree of affection among concurrent events by considering some temporal and spatial patterns.

- **Data alignment and association of event-streaming in mobile ad hoc networks.** In some applications such as emergency services for disaster recovery there is a need for networks where devices can move and be free to dynamically associate with any other devices in their communication range. In this dissertation, we tackled the data alignment and association for event-streaming by considering a network of fixed devices with a pre-existing infrastructure. Future extensions of this work will tackle problems that arise when considering the restrictions imposed by the devices' mobility and there is no infrastructure for the network deployment.

- **Event-streaming processing for the self-managing in Machine-to-Machine (M2M) environments.** Ubiquitous computing environments based on M2M, such as Environment-to-Environment (E2E) connection, has been developed as a new form of communication that allows users to interact among people and objects by connecting their natural physical environments. The goal is to equip many devices into a person environment to focus on natural human-interaction. Thus, users would not need worry about staying within proximity, field of view, or audible distance of an output device, because the system should be responsible to find the most appropriate input and output devices to support communication. For this, the data harvested by the devices must be processed to produce information that enables the devices to react intelligently to their environment.

In this sense, future directions of this work can be focused on the development of solutions oriented to:

- Data aggregation for event-streaming. This concerns in future extensions of the ES-LM and the fuzzy-causal relation to support handling of atomic events that can describe actions occurred within the physical environment at a certain instant of time. By handling this kind of events, the subsets that compose an event-streaming would represent *composite events*, which are complex temporal relationships among atomic events and could be useful to describe complex happenings of the physical environment.
- Distributed detection of predicates. Considering the concept of *composite events*, another extension for the ES-LM concerns to detect the temporal dependencies among streams under a rich palette of time

modalities. Thus, the subsets that compose an event-streaming would represent predicates of interest.

Bibliography

- [All83] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843, November 1983.
- [AMC07] Ian F. Akyildiz, Tommaso Melodia, and Kaushik R. Chowdhury. A survey on wireless multimedia sensor networks. *Computer Networks*, 51(4):921–960, 2007.
- [ASSC02] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.
- [BGM08] Jacques M. Bahi, Arnaud Giersch, and Abdallah Makhoul. A scalable fault tolerant diffusion scheme for data fusion in sensor networks. In *Proceedings of the 3rd international conference on Scalable information systems*, InfoScale '08, pages 10:1–10:5, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
- [Bou13] A. Boulis. Castalia: A simulator for wireless sensor networks. <http://castalia.research.nicta.com.au/>, 2013.

- [BPLOBdlC11] Eva Besada-Portas, Jose A. Lopez-Orozco, Juan Besada, and Jesus M. de la Cruz. Multisensor fusion for linear control systems with asynchronous, out-of-sequence and erroneous data. *Automatica*, 47(7):1399–1408, 2011.
- [CK02] Punit Chandra and Ajay D. Kshemkalyani. Detection of orthogonal interval relations. In *Proceedings of the 9th International Conference on High Performance Computing, HiPC '02*, pages 323–333, London, UK, UK, 2002. Springer-Verlag.
- [CK05] Punit Chandra and Ajay D. Kshemkalyani. Causality-based predicate detection across space and time. *IEEE Trans. Comput.*, 54:1438–1453, November 2005.
- [CK08] Punit Chandra and Ajay Kshemkalyani. Data-stream-based global event monitoring using pairwise interactions. *J. Parallel Distrib. Comput.*, 68:729–751, June 2008.
- [ESW⁺05] Jaime Esteban, Andrew Starr, Robert Willetts, Paul Hannah, and Peter Bryanston-Cross. A review of data fusion models and architectures: Towards engineering guidelines. *Neural Comput. Appl.*, 14(4):273–281, 2005.
- [Fid88] Colin J. Fidge. Timestamps in Message-Passing Systems That Preserve the Partial Ordering. In *Proceedings of the 11th Australian Computer Science Conference (ACSC'88)*, pages 56–66, 1988.
- [GYJO10] Mingyan Gao, Xiaoyan Yang, Ramesh Jain, and Beng Chin Ooi. Spatio-temporal event stream processing in multimedia com-

- munication systems. In *Proceedings of the 22nd international conference on Scientific and statistical database management, SSDBM'10*, pages 602–620, Berlin, Heidelberg, 2010. Springer-Verlag.
- [HCB00] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8, HICSS '00*, pages 8020–, Washington, DC, USA, 2000. IEEE Computer Society.
- [HM04] David L. Hall and Sonya A. H. McMullen. *Mathematical Techniques in Multisensor Data Fusion (Artech House Information Warfare Library)*. Artech House, Inc., Norwood, MA, USA, 2004.
- [ITU01] ITU-T Recommendation G.1010. End-user Multimedia QoS Categories, 2001.
- [KC13] Ajay D. Kshemkalyani and Jiannong Cao. Predicate detection in asynchronous pervasive environments. *IEEE Transactions on Computers*, 62(9):1823–1836, 2013.
- [KS08] Ajay D. Kshemkalyani and Mukesh Singhal. Chapter 1. In *Distributed Computing: Principles, Algorithms, and Systems*. Cambridge University Press, New York, NY, USA, 2008.
- [Ksh96] Ajay D. Kshemkalyani. Temporal interactions of intervals in distributed systems. *Journal of Computer and System Sciences*, 52(2):287 – 298, 1996.

- [Ksh05] Ajay D. Kshemkalyani. Predicate detection using event streams in ubiquitous environments. In *Proceedings of the EUC 2005 Workshops: UISW, NCUS, SecUbiq, USN, and TAUES*, pages 807–816, Nagasaki, Japan, 2005. Springer Berlin / Heidelberg.
- [Ksh07] Ajay D. Kshemkalyani. Temporal predicate detection using synchronized clocks. *IEEE Transactions on Computers*, 56(11):1578–1584, 2007.
- [Lam78] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.
- [Lam86] Leslie Lamport. On interprocess communication. part 1: Basic formalism,. *Distributed Computing*, 1(2):77–85, 1986.
- [Lee90] C.-C. Lee. Fuzzy logic in control systems: fuzzy logic controller. parts I and II. *IEEE Transactions on Systems, Man and Cybernetics*, 20(2):404–435, 1990.
- [LS07] Guo-Liang Lee and Chi-Sheng Shih. Clock free data streams alignment for sensor networks. In *13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 2007. RTCSA 2007*, pages 355–362, 2007.
- [MA75] E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, 1975.
- [Mar82] Marc B. Vilain. A System for Reasoning about Time. In *2nd. (US) National Conference on Artificial Intelligence*, pages 197–201. MIT Press, 1982.

- [Mat88] Friedemann Mattern. Virtual Time and Global States in Distributed Systems. In *Proc. Int. Workshop on Parallel and Distributed Algorithms*, pages 215–226, Gers, France, 1988. North-Holland.
- [PCPH14a] Jose Roberto Perez Cruz and Saul E. Pomares Hernandez. Temporal alignment model for data streams in wireless sensor networks based on causal dependencies. *International Journal of Distributed Sensor Networks*, 2014:1–11, 2014.
- [PCPH14b] Jose Roberto Perez Cruz and Saul E. Pomares Hernandez. Temporal data alignment and association for event-streaming in ubiquitous environments based on fuzzy-causal dependencies. In *2014 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, pages 127–134. IEEE Computer Society, February 2014.
- [PCPHM12] Jose Roberto Perez Cruz, Saul Eduardo Pomares Hernandez, and Enrique Munoz de Cote. Data alignment for data fusion in wireless multimedia sensor networks based on M2M. *KSIIT Transactions on Internet and Information Systems*, 6(1):229–240, 2012.
- [PEMR08] Saul Pomares Hernandez, Jorge Estudillo Ramirez, Luis A. Morales Rosales, and Gustavo Rodríguez Gómez. Logical mapping: An intermedia synchronization model for multimedia distributed systems. *Journal of Multimedia*, 3(5):33–41, 2008.
- [PFD04] Saul Pomares Hernandez, Jean Fanchon, and Khalil Drira. The immediate dependency relation: an optimal way to ensure causal

- group communication. In *Annual Review Of Scalable Computing, Editions World Scientific, Series On Scalable Computing*, pages 61–79, 2004.
- [PHLDRGF09] S.E. Pomares Hernandez, E. Lopez Dominguez, G. Rodriguez Gomez, and J. Fanchon. An efficient delta-causal algorithm for real-time distributed systems. *Journal of Applied Sciences*, 9:1711–1718, 2009.
- [RDLT09] Gabriel Robles-De-La-Torre. Virtual reality: Touch / haptics. In E. Bruce Goldstein, editor, *Encyclopedia of Perception*. Sage Publications, Thousand Oaks, CA, USA, 2009.
- [Rei09] Sean Reilly. Multi-event handlers for sensor-driven ubiquitous computing applications. In *Proceedings of the 2009 IEEE International Conference on Pervasive Computing and Communications*, pages 1–2, Washington, DC, USA, 2009. IEEE Computer Society.
- [Ros95] Timothy J. Ross. Properties of membership functions fuzzification and defuzzification. In *Fuzzy Logic with Engineering Applications*. McGraw Hill, New York, NY, USA, 1995.
- [STT01] Kenichi Shimamura, Katsuya Tanaka, and Makoto Takizawa. Group communication protocol for multimedia applications. In *Proceedings of the 2001 International Conference on Computer Networks and Mobile Computing (ICCNMC'01)*, ICCNMC '01, pages 303–308, Washington, DC, USA, 2001. IEEE Computer Society.

-
- [Sug85] Michio Sugeno. An introductory survey of fuzzy control. *Information Sciences*, 36(1-2):59 – 83, 1985.
- [TS85] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-15(1):116–132, 1985.
- [Zad65] Lofti A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.