Temporal Alignment Model for Data Streams in Wireless Sensor Networks Based on Causal Dependencies
Jose Roberto Perez Cruz, Saúl Eduardo Pomares Hernández

To cite this version:

HAL Id: hal-01096593
https://hal.archives-ouvertes.fr/hal-01096593
Submitted on 19 Dec 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Temporal alignment model for data streams in wireless sensor networks based on causal dependencies

Jose Roberto Perez Cruz\textsuperscript{1} and Saul E. Pomares Hernandez\textsuperscript{1,2,3}

\textsuperscript{1}Department of Computer Science, Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Luis Enrique Erro 1, C.P. 72840, Tonantzintla, Puebla, Mexico.
\textsuperscript{2}CNRS, LAAS, 7 avenue du Colonel Roche, F-31400 Toulouse, France
\textsuperscript{3}Univ de Toulouse, LAAS, F-31400 Toulouse, France

E-mail addresses: jrpc@inaoep.mx (J.R. Perez Cruz), spomares@inaoep.mx (S.E. Pomares Hernandez).

New applications based on wireless sensor networks (WSN), such as person-locator services, 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 \textit{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 paper, we propose a temporal alignment model for data streams that identifies temporal relationships and which does not require: the use of synchronized clocks, global references, centralized schemes or additional synchronization signals. The identification of temporal relationships without the use of synchronized clocks is achieved by translating temporal dependencies based on a timeline to causal dependencies among streams. Finally, we show the viability and the effectiveness of the model by simulating it over a sensor network with multi-hop communication.

1 Introduction

Emerging applications based on wireless sensor networks (WSN) such as remote monitoring of biosignals, multimedia surveillance systems and person locator services \cite{1} ubiquitously\textsuperscript{1} harvest a large amount of continuous time-based data, as audio or video, that is generated from several sensor nodes in a distributed and concurrent way \cite{10, 6, 18}. Specifically, the adopted transmission paradigm in such environments is called \textit{event-streaming}, which represents the generation and the transmission of data as a continuous stream of events reported by multiple sources \cite{11}. In order to be useful to the application, all the collected data require a certain degree of post-processing (e.g. data fusion\textsuperscript{2}).

For example, suppose that there is a network of fixed cameras along an area, which aim to monitor a person (see Figure 1). As each camera has a limited vision field, the resultant single video must be formed from multiple possible video sequences, collected by different cameras. Thus, all the collected video sequences will be processed/fused to produce useful information (see Figure 1).

To perform some kind of analysis or processing, all the data originated through the event-streaming must be temporally aligned in some way to make them functional and coherent. To achieve this, the sensors in the network may need to agree on some common temporal references to the whole system \cite{8, 4, 10, 14}. Unfortunately, the characteristics and restrictions of a WSN make it difficult to establish such references. This is mainly due to: 1) the resources’ constraints, 2) the channel variability, 3) the dynamicity in the topology, 4) the lack of perfectly synchronized physical clocks, 5) the absence of shared memory, and 6) the asynchronous nature of the event-streaming \cite{6}.

In order to avoid the use of synchronized clocks, a clock-free alignment approach for data streams was proposed in \cite{14}. This solution is based on the fact that in most sensor networks, 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 commu-
unication scheme, the approach has shown that 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.

In this paper, we propose a new model called Event-Streaming Logical Mapping (ES-LM), for the temporal data alignment in WSNs. The ES-LM model is based on the event-streaming paradigm, the logical mapping model [20] and the data alignment approach described in [17]. The ES-LM uses pairwise interactions between nodes based on the happened-before relation [12]. The ES-LM performs the data alignment by translating temporal dependencies among streams, based on a time-line, to causal dependencies. Such translation allows us to construct a virtual time-line. 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. Finally, we show the viability and the effectiveness of the model by simulating it over a sensor network with multi-hop communication.

This paper is structured as follows. Section 2 presents the system model and the background, including our definition of the event-streaming as an abstract data type. Section 3 presents a description of the proposed temporal data alignment model for event-streaming. In Section 4 we present the analysis of the model and the simulation results. Finally, Section 5 presents the conclusions.

2 Preliminaries

2.1 System model

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

- **Processes.** Each entity associated to the WSN (sensors or sink) is represented as an individual process. Hence, a WSN is a set of processes $P = \{p_1, p_2, \ldots\}$ that communicate with each other by message passing. A process can only send one message at a time.

- **Messages.** We consider a finite set of messages $M$ sent by a process $p \in P$. Such messages contains the samples of audio, video or many other physical signals that each sensor collects. Henceforth, we will refer to a sample as a message. For a message, the sample time $x$ is the time instant at which a process $p$ conducts such a sample. Thus, each message $m \in M$ can be identified as $m(p, x)$.

- **Events.** An event represents an instant execution performed by a process. For our problem of data alignment, we only consider the send and delivery events.

1. The *send* event refers to the emission of a message executed by a process.
2. The *delivery* event 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 $\text{send}(m)$ the emission event and by $\text{delivery}(p, m)$ the delivery of $m$ to the process $p$. The whole set of events in the system is the finite set: $E = \{\text{send}(m) : m \in M\} \cup \{\text{delivery}(p, m) : m \in M, p \in P\}$.

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

- **Transmission delay.** For a message $m \in M$ there is a time period to contend for the transmission media 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.

2.2 Background and Definitions

A suitable way to order events in an asynchronous distributed system is the happened-before relation (HBR) defined by Leslie Lamport [12]. This relation establishes the rules to determine whether an event is the cause or the effect of another event, without using global references.

The HBR is a strict partial order on events, defined as follows:

**Definition 1** The happened-before relation, $\rightarrow$, 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 reception of the same message in 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 $a \rightarrow b$ and $b \rightarrow a$; it is denoted by $a \parallel b$ [12].

**Immediate dependency relation (IDR).** The IDR is the transitive reduction of the HBR [19]. The IDR is defined as follows:

**Definition 3** Two events $a, b \in E$ have an immediate dependency relation (denoted by “$\downarrow$”) if:

$$a \downarrow b \text{ if } a \rightarrow b \land \forall c \in E, \neg(a \rightarrow c \rightarrow b)$$
Intervals. An interval is a set of events which occur during a period of time. If the events that compose an interval satisfy a certain order, then such interval is called ordered interval. The works of Shimamura et al. [21] and Pomares et al. [20] define the following ordered interval composition.

**Definition 4** Let X be an interval of sequentially-ordered events at a process pi; X \( \subseteq E \), and \( x^-, e, x^+ \in X \); where \( x^- \) is the left endpoint and \( x^+ \) is the right endpoint of interval X, such that \( \forall e \in X, x^- \rightarrow e \rightarrow x^+ \) 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.

**Happened-before relation for intervals.** Lamport establishes in [13] 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.

**Definition 5** 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 4 and Pomares et al. [20], 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 sets of sequentially-ordered events. The set of events A occurs before the set of events B if any of the following conditions are satisfied:

1. \( A \rightarrow B \) if \( a^+ \rightarrow b^- \)
2. \( A \rightarrow B \) if \( \exists C, \text{such that } a^+ \rightarrow c^- \wedge c^+ \rightarrow b^- \)

**Definition 6** 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 [20]:

\[
A||B \text{ if } a^- \parallel b^- \wedge a^+ \parallel b^+
\]

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

**2.2.1 The logical mapping model**

The logical mapping model introduced in [20] is useful to represent pairwise interactions between processes. Such model expresses temporal relations between sets of events in terms of the happened-before relation for intervals.

<table>
<thead>
<tr>
<th>Logical mappings</th>
<th>expressed by endpoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>precedes: ( A \rightarrow B )</td>
<td>( a^+ \rightarrow b^- )</td>
</tr>
<tr>
<td>simultaneous: ( A</td>
<td></td>
</tr>
<tr>
<td>ends: ( A \rightarrow (C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( c^- \parallel d^- \wedge c^+ \parallel d^+ )</td>
</tr>
<tr>
<td>starts: ( (C</td>
<td></td>
</tr>
<tr>
<td>overlaps: ( A \rightarrow (C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( c^- \parallel d^- \wedge c^+ \parallel d^+ )</td>
</tr>
<tr>
<td></td>
<td>( c^- \rightarrow b^-, d^+ \rightarrow b^- )</td>
</tr>
</tbody>
</table>

The logical mapping translation involves every pair X, Y of intervals of a temporal relation. Each pair is segmented into four subintervals: \( A(X, Y), C(X, Y), D(X, Y), \) and \( B(X, Y) \), as shown in Table 1.

The logical mapping model identifies five logical mappings, which are sufficient to represent all possible temporal relations between continuous media (interval-interval relations [2]), discrete media (point-to-point relations), and discrete-continuous media relations [15].

**2.2.2 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.

**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 a portion of data has been collected from the environment.

**Definition 7** An atomic event is a tuple \( e(p_i, m(p_i, x)) \), where \( p_i \) refers to the process where the event is executed and \( m(p_i, x) \) is a message \( (m \in M) \) originated by process \( p_i \).

As we stated above, we consider only two types of events: send and delivery. We denote the atomic delivery event by \( delivery(p_i, m(p_i, x)) \), 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.

**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 se-
quently 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 8** A local-stream is a poset $(S_i, \rightarrow_i)$ where $S_i$ is a finite set of atomic events $S_i = \{e_1, e_2, \ldots, e_n\}$ 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 $S_i = \{e_1, e_2, \ldots, e_n\}$, 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.

**Event-streamings.** An event-streaming is a collection of subsets of local-streams generated by different processes. Such subsets of local-streams are grouped and arranged according to their causal dependencies into sets denoted as $Q^R_q$, where $R_q$ denotes the set of identifiers of the processes that generated the events in $Q^R_q$. Thus, in a generic way, an event-streaming $ES_\Theta$ can be viewed as the general causal structure:

$$ES_\Theta = Q^R_1 \rightarrow Q^R_2 \rightarrow \cdots \rightarrow Q^R_{m-1} \rightarrow Q^R_m$$

We formally define an event-streaming as follows:

**Definition 9** An event-streaming is a poset $(ES_\Theta, \rightarrow)$ where $ES_\Theta$ is a finite set of subsets of events $ES_\Theta = \{Q^R_1, Q^R_2, \ldots, Q^R_m\}$ arranged according to the causal relation $\rightarrow$; $\Theta$ is the set of the identifiers of the processes that generated the events; and each $Q^R_q \in ES_\Theta$ is a subset of events generated by the processes whose identifiers form the set $R_q$.

Within an event-streaming each set $Q^R_q$ is a collection of subsets of local-streams $\Omega_1, \Omega_2, \ldots, \Omega_k$, where $\Omega_i \subseteq S_i$, $\Omega_j \subseteq S_j$, and $\Omega_k \subseteq S_k$ such that $\Omega_i \cup \Omega_j \cup \cdots \cup \Omega_k$ (i, j, ..., k \in $R_q$). Furthermore, the events in any set $\Omega_i$ are arranged according to the local causal relation $\rightarrow_i$, which form the poset $(\Omega_i, \rightarrow_i)$. For example, suppose that three processes $p_i$, $p_j$ and $p_k$ generated three local-streams as depicted in Figure 2. When a process collects such local-streams, it constructs four subsets: $Q_1^{(i)}$ containing events exclusively generated by $p_i$, $Q_2^{(i,j)}$ containing events generated by $p_i$ and $p_j$, $Q_3^{(i,j,k)}$ containing events generated by $p_i$, $p_j$ and $p_k$, and $Q_4^{(i,j,k)}$ containing events generated by $p_j$ and $p_k$.

**Figure 2:** Representation of the subsets of an event-streaming.

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

3 Temporal data alignment for event-streaming

3.1 The problem of data alignment for event-streaming

Based on the definition of the data stream alignment problem given in [14], we define the problem of data alignment for event-streaming as follows:

**Definition 10** Data alignment problem for event-streaming: Given a set of local-streams: $\{S_1, S_2, S_3, \ldots\}$, 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 $Q^R_q$. In this sense, the synchronization error establishes the temporal distance between the execution of a pair of messages of interest.

3.2 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. However, for our problem this covers only the base case, which is the alignment of streams whose events have all been generated by a single process. In the following sections, we present an extension to the native logical mapping called event-streaming logical mapping model (ES-LM). The ES-LM establishes how an event-streaming $ES_\Theta$ (a stream composed by events generated by several processes) 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 $Q^R_q \in ES_\Theta$ is related to a local-stream $Y_k$. We note that any subset $Q^R_q$ is considered as a sub-event-streaming. In our ES-LM model, without loss of generality, it is assumed that $\neg Q^R_q \rightarrow Y_k$ or $\neg Q^R_q \parallel Y_k$. Therefore, when a subset $Q^R_q$ is aligned to a local-stream $Y_k$, a new sub-event-streaming is generated according to the left column of Table 2. The resultant sub-event-streaming has the following general causal structure:
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 2).

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

3.3 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 \( Y_d \) \((X_c = Y_d \lor X_c \not\equiv Y_d)\). Applying the native logical mapping, we generate the first event-streaming \( ES_0 \) as follows.

We construct a first subset \( Q_1^{(c)} \) with the first non-concurrent events in \( X_c \) (see Table 3). 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 3).

Then, according to Table 3, 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 bounded by the beginning of \( Y_d \) and the end of any of the two local-streams (see Figure 4).

![Figure 3: Aligning the first subset of the first event-streaming.](image)

![Figure 4: Aligning the second subset of the first event-streaming.](image)

The last subset \( Q_3^{(w)} \) is constructed depending upon which local-stream finishes first. If \( X_c \) finishes first, the last subset will contain the remaining events of \( Y_d \) (see Table 3). Otherwise, the last subset will contain the remaining events of \( X_d \) (see Table 3). These two cases are illustrated in Figure 5.

![Figure 5: Aligning the last subset of the first event-streaming.](image)

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 \( d \).

<table>
<thead>
<tr>
<th>Table 3: A.1 Alignment of two local-streams</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Q_1^{(c)} )</td>
</tr>
<tr>
<td>( \bullet ) if ( x' \in X_c, \ y' \in Y_d : x' \rightarrow y' ),</td>
</tr>
<tr>
<td>( {x \in X_c : delivery(p_d, x) \rightarrow send(y')} )</td>
</tr>
<tr>
<td>( \bullet ) otherwise, ( \emptyset )</td>
</tr>
<tr>
<td>( Q_2^{(c, d)} )</td>
</tr>
<tr>
<td>( \bullet ) if ( x' \in X_c, \ y' \in Y_d : x' \rightarrow y' ),</td>
</tr>
<tr>
<td>( {x \in X_c : delivery(p_d, x') \rightarrow send(y)} )</td>
</tr>
<tr>
<td>( \bullet ) if ( x' \in X_c, \ y' \in Y_d : x' \rightarrow y' ),</td>
</tr>
<tr>
<td>( {x \in X_c : send(x) \rightarrow delivery(p, y')} )</td>
</tr>
<tr>
<td>( \bullet ) otherwise, ( X_c \setminus Q_1^{(c)} \cup Y_d )</td>
</tr>
<tr>
<td>( Q_3^{(w)} )</td>
</tr>
<tr>
<td>( \bullet ) if ( y' \in Y_d : \gamma \in Y_d : delivery(p, y') \rightarrow send(y) )</td>
</tr>
<tr>
<td>( \bullet ) otherwise, ( \emptyset )</td>
</tr>
</tbody>
</table>

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 \( X_\beta \), where \( \beta \) 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 \( X_\beta \).

Using \( X_\beta \) and \( Y_k \), we construct a new event-streaming forming the subsets \( Q_\gamma^{(n)} \) of the general causal structure \( ES_\Theta = Q_1^{(c)} \rightarrow Q_2^{(c, d)} \rightarrow \cdots \rightarrow Q_{n-1}^{(c, d)} \rightarrow Q_n^{(w)} \) by detecting the concurrences between \( X_\beta \) and \( Y_k \).

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 \( Q_\gamma^{(n)} \in X_\beta \) that precede the local-stream \( Y_k \) to form the first subsets of the new event-streaming (see stage B.1 of Table 4). 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 \( Q_\gamma^{(n)} \in ES_\Theta \) (see Figure
Table 2: Event-streaming logical mapping

<table>
<thead>
<tr>
<th>Data alignment for event-streaming</th>
<th>ES logical mappings</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q^{E5}_a \leftrightarrow$</td>
<td>$s$-starts: $Q^E_a \rightarrow Q^E_w$</td>
</tr>
</tbody>
</table>
| [ ] if $\forall x^e \in + \text{ } T_a^E, \exists y^e \in Y_k : x^e \rightarrow y^e$,  
  $\{ x \in Q^E_a : \text{delivery}(p_k, x) \rightarrow \text{send}(y^e) \}$  
| [ ] otherwise, $\emptyset$ |
| $Q^{E5}_b \leftrightarrow$ | $s$-simultaneous: $\emptyset \rightarrow Q^E_b$ |
| [ ] if $y^e \in Y_k : x^e \rightarrow y^e$,  
  $(Q^E_b - Q^E_a) \cup (Y_k - \{ y \in Y_k : \text{delivery}(p_k, x^e) \rightarrow \text{send}(y) \})$  
| [ ] if $\forall x^e \in + \text{ } T_a^E, \exists y^e \in Y_k : y^e \rightarrow x^e$,  
  $(\{ x \in Q^E_a : \text{delivery}(p_k, x) \rightarrow \text{send}(y^e) \} - Q^E_b) \cup Y_k$  
| [ ] otherwise, $(Q^E_b - Q^E_a) \cup Y_k$ |
| $Q^{E5}_b \leftrightarrow$ | $s$-ends: $Q^E_b \rightarrow Q^E_b$ |
| [ ] if $\forall x^e \in + \text{ } T_a^E, \exists y^e \in Y_k : y^e \rightarrow x^e$,  
  $Q^E_b \rightarrow \{ x \in Q^E_b : \text{delivery}(p_k, x) \rightarrow \text{send}(y^e) \}$  
| [ ] otherwise,  
  $\{ y \in Y_k : \text{delivery}(p_k, x) \rightarrow \text{send}(y^e) \}$ |

6).

![Figure 6: Aligning the first subsets of events of an event-streaming](image)

If a subset $Q^E_a \in X_β$ 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 $ES_\emptyset$. The new subset $Q'^E_a$, the first of the two new subsets, will contain the part of $Q^E_a$ whose events have no concurrence. To determine the events without concurrences, it is necessary to identify the event $x \in Q^E_a$ that immediately precedes the beginning of the local-stream $Y_k$, (see line 1.2 of Table 4). For the example depicted in Figure 6, the new subset $Q'^E_a$ corresponds to the subset $Q'^E_3$. The remaining events of $Q^E_a$ are aligned as stated in the following stage.

B.2 Aligning the subsets of events with concurrences between an event-streaming and a local-stream If during stage B.1 a subset $Q^E_a$ 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 $Q^E_a$ with concurrent events to form a new subset $Q'^E_a$ (see line 2.1 of Table 4).

Once the beginning of the concurrent parts of both streams are detected, according to stage B.2 of Table 4 (lines 2.2 and 2.3), the all subsequent subsets $Q^E_b \in X_β$ are attached with the corresponding concurrent events of $Y_k$, until one of the two streams finishes. This means that for each subset $Q^E_b$ in the concurrent part of $X_β$, a new subset $Q'^E_b$ will be constructed for the new event-streaming $ES_\emptyset$ (see Figure 7).

![Figure 7: Aligning the subsets of events with concurrences.](image)

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 ($y^e \rightarrow X_β^*$), the last concurrent subset $Q'^E_b \in X_β$ can contain some events that are concurrent with $Y_k$ and other events that have no concurrence (see Figure 8). If this is the case, $Q^E_b$ needs to be segmented to construct two new subsets for the new event-streaming $ES_\emptyset$. The new subset $Q'^E_b$, the first of the two new subsets, will contain the concurrent part of $Q^E_b$ and the concurrent events of $Y_k$. To determine such concurrent events, it is necessary to identify the event $x \in Q^E_b$ 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 4). For the example depicted in Figure 8, the new subset $Q'^E_b$ corresponds to subset $Q'^E_3$. The remaining events of $Q^E_b$ are aligned as stated in the following stage.
Table 4: Alignment process between an event-streaming and a local-stream

<table>
<thead>
<tr>
<th>B.1</th>
<th>If $\forall x^+ \in X^+<em>B, \exists y^+ \in Y_k : x^+ \to y^+$ a new set $Q^*</em>{a}$ is created as follows:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$Q^*_{a} \leftarrow$</td>
</tr>
<tr>
<td></td>
<td>- for each $Q^<em>_{b} \in X_B : -(3x \in Q^</em>_{b} : \text{delivery}(p_k, x) \downarrow \text{send}(y^-))$</td>
</tr>
<tr>
<td></td>
<td>- if $\exists x \in Q^*_{b} : \text{delivery}(p_k, x) \downarrow \text{send}(y^-),$</td>
</tr>
<tr>
<td></td>
<td>$Q' = { x \in Q^*_{b} : \text{delivery}(p_k, x) \to \text{send}(y^-) }$</td>
</tr>
<tr>
<td></td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td>Case A. If $Q^+<em>c - Q' \neq \emptyset$, a new set $Q^*</em>{c}$ is created by:</td>
</tr>
<tr>
<td></td>
<td>$Q^<em>_{c} \leftarrow (Q^+_c - Q') \cup (Y_k - { y \in Y_k : \text{delivery}(p_k, Y^</em>_{c}) \to \text{send}(y^-) })$</td>
</tr>
<tr>
<td></td>
<td>2.1</td>
</tr>
<tr>
<td></td>
<td>Case B. If $\forall x^+ \in X^+<em>B, \exists y^+ \in Y_k : x^+ \to y^+$ a new set $Q^*</em>{c}$ as follows:</td>
</tr>
<tr>
<td></td>
<td>$Q^*_{c} \leftarrow$</td>
</tr>
<tr>
<td></td>
<td>- for each $Q^*_{b} \in X_B : -(\text{delivery}(p_k, x) \downarrow \text{send}(y^-)),$</td>
</tr>
<tr>
<td></td>
<td>$Q^<em>_{b} \cup { (Y_k - { y \in Y_k : y \in Q^</em><em>{c} \cup \ldots \cup Q^*</em>{c} }) - { y \in Y_k : \text{delivery}(p_k, Y^*_{c}) \to \text{send}(y^-) } }$</td>
</tr>
<tr>
<td></td>
<td>- if $\exists x \in Q^*_{b} : \text{delivery}(p_k, x) \downarrow \text{send}(y^-),$</td>
</tr>
<tr>
<td></td>
<td>$Q' = { x \in Q^*_{b} : \text{delivery}(p_k, x) \to \text{send}(y^-) }$</td>
</tr>
<tr>
<td></td>
<td>2.2</td>
</tr>
<tr>
<td></td>
<td>Case C. If $\forall x^+ \in X^+<em>B, \exists y^+ \in Y_k : y^+ \to x^+$ a new set $Q^*</em>{c}$ as follows:</td>
</tr>
<tr>
<td></td>
<td>$Q^*_{c} \leftarrow$</td>
</tr>
<tr>
<td></td>
<td>- for each $Q^*_{b} \in X_B : -(\text{delivery}(p_k, x) \downarrow \text{send}(y^-)),$</td>
</tr>
<tr>
<td></td>
<td>$Q^<em>_{b} \cup { (Y_k - { y \in Y_k : y \in Q^</em><em>{c} \cup \ldots \cup Q^*</em>{c} }) - { y \in Y_k : \text{delivery}(p_k, Y^*_{c}) \to \text{send}(y^-) } }$</td>
</tr>
<tr>
<td></td>
<td>- if $\exists x \in Q^*_{b} : \text{delivery}(p_k, x) \downarrow \text{send}(y^-),$</td>
</tr>
<tr>
<td></td>
<td>$Q' = { x \in Q^*_{b} : \text{delivery}(p_k, x) \to \text{send}(y^-) }$</td>
</tr>
<tr>
<td></td>
<td>2.3</td>
</tr>
<tr>
<td></td>
<td>Case A. If $\forall x^+ \in X^+<em>B, \exists y^+ \in Y_k : y^+ \to x^+$ a new set $Q^*</em>{d}$ as follows:</td>
</tr>
<tr>
<td></td>
<td>$Q^*_{d} \leftarrow$</td>
</tr>
<tr>
<td></td>
<td>- if $Q^<em>_{b} - Q' \neq \emptyset$, $Q^</em>_{b} - Q'$</td>
</tr>
<tr>
<td></td>
<td>- otherwise, for each $Q^<em>_{b} \in { X_B - Q^</em><em>{c} \cup \ldots \cup Q^*</em>{c} }$</td>
</tr>
<tr>
<td></td>
<td>3.1</td>
</tr>
<tr>
<td></td>
<td>Case B. If $\forall x^+ \in X^+<em>B, \exists y^+ \in Y_k : x^+ \to y^+$ a new set $Q^*</em>{d}$ is created by:</td>
</tr>
<tr>
<td></td>
<td>$Q^<em>_{d} \leftarrow Y_k - { y \in Y_k : y \in Q^</em><em>{c} \cup \ldots \cup Q^*</em>{c} }$</td>
</tr>
<tr>
<td></td>
<td>3.2</td>
</tr>
</tbody>
</table>

### B.3 Aligning the last subsets of events without concurrency

This stage is explained through two cases.

**Case A. $Y_k$ finishes before $X_B$.** If at the end of stage B.2, the last subset $Q^*_{a}$ was segmented, the second created subset, denoted as $Q^*_{d}$, will contain the remaining non-concurrent events of $Q^*_{b}$ (see line 3.1 of Table 4). In the example of Figure 8, such subset $Q^*_{d}$ corresponds to the subset $Q^*_{a}$.

The fact that the local-stream $Y_k$ finishes first ($y^+ \to X^+_B$) implies that the concurrent parts of both streams finish along with $Y_k$. Therefore, according to line 3.2 of Table 4, the remaining subsets $Q^*_{c}$ in $X_B$ will become the last subsets $Q^*_{d} \in ES_{\emptyset}$. In the example of Figure 8, the last subsets $Q^*_{d}$ are the subsets $Q^*_{d}, Q^*_{d}, Q^*_{d}$. 

**Case B. $X_B$ finishes before $Y_k$.** The fact that the event-streaming $X_B$ finishes first ($X^+_B \to y^+, y^+ \in Y_k$) means that the concurrent parts of both streams finish along with the event-streaming $X_B$. After the last subset $Q^*_{c}$ in $ES_{\emptyset}$ was constructed with the concurrent events of $X_B$ and $Y_k$, only one more subset $Q^*_{d}$ is constructed according to line 3.3 of Table 4. Such subset $Q^*_{d}$ will contain the remaining events of the local-stream $Y_k$. In the example of Figure 9, the last subset $Q^*_{d}$ corresponds to the subset $Q^*_{a}$.

![Figure 8: Aligning the last subsets of events without concurrency when $y^+ \to X^+_B$.](image)

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

4 Analysis and results

4.1 Proof of the temporal data alignment

In this section we prove that by following the ES-LM 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 \( Q^R_q \) in an event-streaming establishes a virtual time-line, where each subset \( Q^R_q \) 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 \( Q^R_q \) 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 8 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 \( S_i = \{ e_0 \rightarrow e_{n+1} \rightarrow \cdots \rightarrow e_{n-1} \rightarrow e_n \} \). The fact that the events of \( S_i \) are arranged by \( \rightarrow \) implies that each event happens before another at a different 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 2, 3 and 4) during the data alignment, the subsets of events that compose a new event-streaming \( ES_\beta \), are formed by segmenting two streams (a local-stream and an event-streaming or two local-streams). From Tables 3 and 4 (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 \( Q^R_{u-1} \) and \( Q^R_u \). Let \( Y_i \) be a local-stream and let \( X \) denote a local-stream or an event-streaming such that \( \forall x \in X \), \( \exists y_x \in Y_i : x^* \rightarrow y^* \). Assuming that \( e^* (e^* \in Y_i \) or \( e^* \in X \)) is the endpoint \((e^* = y^* \vee e^* = y^* \wedge e^* = x^*)\), whose identification triggered the segmentation, \( \forall x \in X \), \( Q^R_{u-1} \) and \( Q^R_u \) are constructed according to one of the three following cases:

1. if \( e^* \in Y_i \), \( e^* = y^* \) and \( x^* \rightarrow y^* \) then \( Q^R_{u-1} = \{ e \in X : delivery(p, x) \rightarrow send(e^*) \} \cup \{ y \in Y_i : send(y) \rightarrow delivery(p, x^*) \} \);
2. if \( e^* \in X \), \( e^* = x^* \) and \( x^* \rightarrow y^* \) then \( Q^R_{u-1} = \{ e^* \} \cup \{ y \in Y_i : send(y) \rightarrow delivery(p, x) \) \} \cup \{ y \in Y_i : send(y) \rightarrow delivery(p, e^*) \} \);
3. if \( e^* \in Y_i \), \( e^* = y^* \), and \( y^* \rightarrow x^* \) then \( Q^R_{u-1} = \{ e^* \} \cup \{ y \in Y_i : send(y) \rightarrow send(e^*) \} \}

In each of these three cases, happened-before relationships are established among the left endpoints of \( Q^R_{u-1} \) and the right endpoints of \( Q^R_u \). This means that for a pair of subsets \( Q^R_{u-1} \) and \( Q^R_u \) we have that \( \forall (\omega^*_i, \omega^*_j) \in \Theta \), \( Q^R_{u-1} \times Q^R_u : \omega^*_i \rightarrow \omega^*_j \) \((i, j \in \beta)\).

Thus, let \( \omega^*_i \) and \( \omega^*_j \) be two events such that \( \omega^*_i, \omega^*_j \in \Omega_i \), \( \Omega_i \in Q^R_{u-1} \) and \( \Omega_j \subseteq S_j \) (for any local-stream \( S_j \)) if \( \omega^*_i \neq \omega^*_j \) then \( \omega^*_i \rightarrow \omega^*_j \). By the transitive property of the HBR we have that \( \omega^*_i \rightarrow \omega^*_j, \forall \omega^*_j \in Q^R_u \). Moreover, for any event \( \omega \in Q^R_u \) \((\omega \neq \omega^*_i) \), \( \omega \rightarrow \omega^*_i \); thereby, transitively \( \omega \rightarrow \omega^*_j \). Therefore, by Definition 5 we have \( Q^R_{u-1} \rightarrow Q^R_u \).

Thus the subsets of an event-streaming are chronologically ordered representing a time-line, where each subset \( Q^R_q \) is a time-slot.

Corollary 1 Each subset \( Q^R_q \in ES_\omega \) represents a time-slot.

Part II. To demonstrate that each subset \( Q^R_q \) in an event-streaming represents a unique time-slot, we formulate and prove the following Lemma:

Lemma 2 Each subset \( Q^R_q \) represents a unique time-slot, therefore, any pair of subsets \( Q^R_q \) and \( Q^R_r \) that compose
an event-streaming ES, is disjoint.

$$\forall Q^R_u, Q^R_v \in ES_\emptyset, u \neq v : Q^R_u \cap Q^R_v = \emptyset$$

**Proof of Lemma 2** We prove Lemma 2 by contradiction. Therefore, we suppose that $$\forall Q^R_u, Q^R_v \in ES_\emptyset, u \neq v : Q^R_u \cap Q^R_v \neq \emptyset$$, i.e., $$\exists x_p : x_p \in Q^R_u \wedge x_p \in Q^R_v$$.

According to the ES-LM model, the subsets $$Q^R_q$$ 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 $$Q^R_u$$ must be related to another $$Q^R_v$$ ($$u \neq v$$) according to one of the five logical mappings described in Table 2. This means that for any $$e_i \in Q^R_u$$ and any $$c_\mu \in Q^R_v$$, $$e_i \rightarrow c_\mu$$ or $$c_\mu \rightarrow e_i$$ or $$c_\mu \rightarrow e_i$$ such that if there is $$x_p$$ such that $$x_p \in Q^R_u \wedge x_p \in Q^R_v$$, implies that $$x_p$$ 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). □

### 4.2 Simulation results

We have simulated the temporal data alignment model for event-streaming using the Castalia simulator [5]. The simulation scenario is within a set of 50 nodes that were arranged into multi-hop paths, where the nodes are separated by distances between 5 and 10 meters in a field of 200 × 200 meters. Through each multi-hop path, a node reaches the sink helped by up to 10 relay nodes. With this arrangement, each relay node aligns the streams generated by the predecessor nodes in the path in such a way that if there exist $$n_i$$ nodes behind a node $$p_i$$, $$p_i$$ aligns at most $$n_i$$ streams.

For the simulation, we implemented the model using the well-known vector clock structure [16, 7] to identify and preserve the causal relations among events. Therefore, we obtain a computational cost as well as communication and storage overheads of $$O(n_s)$$, where $$n_s$$ is the number of nodes that are related to a certain node during the data transmission.

To transmit a local-stream we use two types of causal messages: *begin* and *end*, and a type of FIFO message: *fifo*, which does not carry any causal information. So, a process $$p_i$$ generates a local-stream $$S_i$$ by sending a *begin* message ($$S_i^*$$) followed by certain number of *fifo* messages. Finally, to notify a process $$p_j$$ that the transmission of the local-streaming has been finished, the process $$p_i$$ sends an *end* message ($$S_i^*$$).

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. Each generated local-stream was composed by a random number of messages between 9 and 100, that were generated using sampling rates between 25 and 1000 milliseconds.

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 ES-LM. The ES-LM use only the causal dependencies between the event-streamings to perform the data alignment and do not use any kind of physical time.

At each hop we took the causal messages *begin* and *end* exchanged during the transmissions of the local-streams to determine the synchronization error between a pair of streams.

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

For example, in the scenario depicted in Figure 10, process $$p_i$$ sends the message $$m(p_j, p_i)$$ to $$p_j$$, which indicates the beginning of the local-stream $$S_i$$. Assuming that $$p_j$$ is generating the local-stream $$S_j$$, the synchronization error between $$S_i$$ and $$S_j$$ is determined by the difference between the sampling time of $$m(p_j, p_i)$$ and the sampling time of $$m(p_j, p_i)$$.

![Figure 10: Example of the alignment of causal messages.](image)

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

$$e_j(e^*_a, e^*_b) = |TS_j(e^*_a) - TS_j(e^*_b)|$$

where $$e_j$$ is the synchronization error measured at process $$p_j$$ and $$TS_j$$ is a sample time at process $$p_j$$.

By using sampling rates between 25 and 1000 milliseconds, we show that the synchronization error can be bounded according to the transmission delay as shown in Figure 11.

### 4.2.1 Analysis of the results

Based on the ES-LM, the data alignment is performed in the intermediate nodes while the streams are propagated through the network. In our case the data alignment is achieved by ensuring at each intermediate node the synchronization error $$e_j(e^*_a, e^*_b)$$ is bounded by the transmission delay for a pair of endpoints $$e^*_a$$ and $$e^*_b$$. This means that the execution of the send events of such pair of endpoints take place at most at $$e_j(e^*_a, e^*_b)$$ 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 ES-LM. In such event-streaming, a pair of endpoints \((e^*_\alpha, e^*_\rho)\) is related at \( p_j \) as shown in Figure 10. 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.

With respect to why the average synchronization error is bounded between sampling rates of 75 ms to 830 ms we present the following analysis (see Figure 11). 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 by our solution 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 Conclusions

A temporal alignment model for data streams in WSNs called event-streaming logical mapping (ES-LM) has been presented. One original aspect of our model is that the data alignment is performed without using synchronized clocks, global references, centralized schemes or additional synchronization signals. This was achieved by translating temporal dependencies based on a time-line to causal dependencies among streams. The ES-LM model constructs a virtual time-line by arranging the transmitted data into causally-ordered sets of events. In terms of the problem of temporal data alignment, it was proven that each ordered set of events determines a specific and unique time slot. An instantiation of the model was simulated over a sensor network with multi-hop communication. The simulation results show that the synchronization error is bounded according to the transmission delay.

A Conflict of interests

The only funding source for this work is mentioned in the acknowledgment section of the paper. The authors declare that there is no conflict of interests regarding the publication of this article.

B Acknowledgments

Jose Roberto Perez Cruz would like to thank to the Council of Science and Technology (CONACYT) of Mexico for the doctoral scholarship with which his studies were supported.

References


