Synchronization Protocol for Real Time Multimedia in Mobile Distributed Systems

Miguel A Olmos Bello, Eduardo Lopez Dominguez, Saúl Eduardo Pomares Hernández, Jose Roberto Perez Cruz

Synchronization Protocol for Real Time Multimedia in Mobile Distributed Systems

Miguel A. Olmos Bello [1,*], Eduardo Lopez Dominguez [1], Saul E. Pomares Hernandez [2] and Jose R. Perez Cruz[3]

[1] Laboratorio Nacional de Informática Avanzada (LANIA), Xalapa 91000, México; molmos.mca15@lania.edu.mx, elopez@lania.mx

[2] Computer Science Department, Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE) Puebla 72840, México; spomares@inaoep.mx

[3] CONACYT—FIC, Universidad Michoacana de San Nicolás de Hidalgo (UMSNH), Morelia 58040, México; jrperezcr@conacyt.mx

Abstract

Mobile distributed systems (MDS) deal multimedia data transmissions among geographically distributed mobile sources. In this environment, the preservation of temporal dependencies among exchanged real-time multimedia data is mainly affected by the asynchronous transmissions, the constantly topology changes, unpredictable delays and the lack of global references as memory and perfectly synchronized clocks. Albeit a few works are oriented to fulfill intermedia temporal dependencies, the vast majority of them do not completely support the constraints and characteristics of MDS. This paper presents a causal protocol oriented to satisfy logical and temporal dependencies among real-time multimedia data in a MDS. One of the main aspects of our protocol is the translation of temporal constraints to causal dependencies of the multimedia data using logical mapping, avoiding the use of global references. With the simulations results we demonstrate that our protocol is effective diminishing the synchronization error. Furthermore, the protocol is efficient as far as processing and storage costs at the mobile hosts, and in the overhead attached per message with a reduced usage of bandwidth across the wired and wireless channels in comparison to the Real-time Transport Protocol (RTP).

1 Introduction

Technological advances in wireless networks have allowed the development of Mobile Distributed Systems (MDS) such as mobile telemedicine applications [3], streaming video services [32] and interactive multiuser games [16], among others. These applications are characterized mainly by a multiparty multimedia communication among geographically distributed mobile sources (mobile host). In such environments, preservating temporal dependencies of the real-time multimedia (continuous and discrete) data, called as intermedia synchronization, is an open research issue [7].

In this context, we refer to *real-time intermedia synchronization* to the preservation of temporal dependencies at runtime among the multimedia data from the generation to the presentation [22].

In general, the preservation of temporal dependencies among the real-time multimedia data is mainly affected by: the heterogeneous characteristics of data (e.g. temporal constraints), and unpredictable communication delays. The multimedia data is classified in two broad types: continuous media (e.g., audio and video) and discrete media (e.g., text, data and images) [23]. The continuous media events are played in a period of time, while discrete media is executed in a certain point of time [28].

In addition, in a MDS, the preservation of temporal dependencies is affected by its inherent characteristics such as: asynchronous multiparty communication with limited bandwidth in wireless channels, limited storage and processing in mobile hosts, and finally but most important, the lack of global references (e.g. wall clock and shared memory).

Although several protocols in the state of the art [20, 2, 5, 10, 19, 26, 25, 9] are intended to fulfill intermedia temporal dependencies, most of them do not completely support the previous depicted environment. These can be categorized according to two design principles: end-to-end and asymmetric. The first category is based on end-to-end design principle which consider each device as equal. However, this category omits main characteristics of mobile distributed systems (MDS): a big set of mobile hosts connected on the network, absence of global references, mobility, limited bandwidth on wireless communication channels and limited storage and processing capabilities in mobile hosts. The second category follows an asymmetric design, the main approach is that most of processing is divided between the physical or logical entities on the network, with better capabilities than mobile hosts [14]. Moreover, in this category existing protocols for MDS do not include heterogeneous data synchronization.

This paper presents a distributed protocol for multimedia synchronization, designed for the real-time multimedia exchanging in MDS group communications. The proposed protocol accomplishes intermedia temporal constraints without the usage of global temporal and memory references. This is achieved by integrating a Delta-Causal algorithm to the Logical Mapping model [22] for intermedia synchronization. By the one hand, following the Logical Mapping model, the intermedia temporal dependencies are translated into causal dependencies, which allows using the well known concept of logical time [24]. On the other hand, the adoption of a decentralized Delta-causal delivery method [20] allows to calculate messages deadline by only using relative time points. The protocol was formally analysed, demonstrating that it is efficient in terms of communication, processing and storage; accomplishing common constraints of mobile devices, as well as wired and wireless communication channels. The feasibility and usefulness of the proposed protocol are shown by implementing it in a simulated MDS, composed by a group of mobile hosts, performing real-time multimedia data transmissions over a cellular network. The results show that the proposed protocol reduces the synchronization errors, without requiring neither previous knowledge of the system and global references. Furthermore, it is demonstrated that the overhead per message is lower than the used for Real-time Transport Protocol (RTP) for wired and wireless channels.

2 Materials and Methods

This section depicts the system model, concepts and definitions implemented in the asymmetric protocol that we present in this work.

2.1 System Model

A cellular network is formed by a set of mobile hosts $P=\{p_1, p_2, ..., p_n\}$ and a set of base stations $BSG=\{BS_1, BS_2, ..., BS_s\}$ [13]. Where the mobile hosts are connected on a wireless network to its corresponding base station (BS) which covers a geographic area called a cell. On the other hand, a BS works as a data terminal equipment (DTE), this way the mobile hosts can communicate with other mobile hosts only through its BS. The base stations are interconnected in a static network by using wired channels, see figure 1.

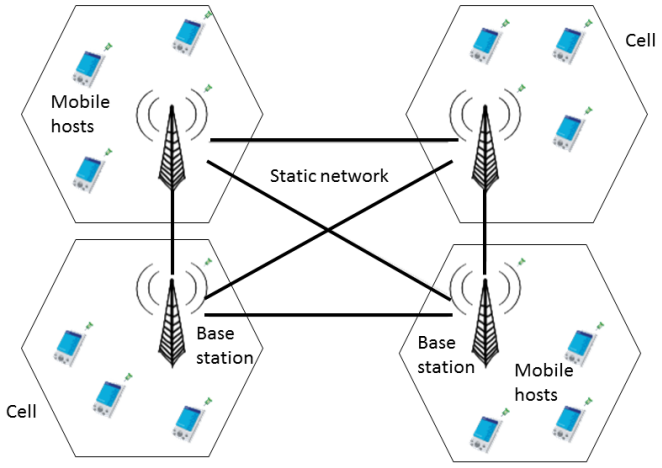

Figure 1: Cellular network architecture for a mobile distributed system.

To achieve the communication between mobile hosts using logical mapping [22], the next elements are defined:

- Messages: A finite number of messages M, where each $m \in M$ is distinguished by a tuple $m = (p,x)$, where the mobile host $p \in P$ is the mobile host sender of m and x is the logical clock or counter of messages sent by p. The set of destinations of a message m is P, which represents every mobile host in the communication group.
- Events: When m is created by a mobile host p, a $send(m)$ event is triggered and a $delivery(p,m)$ event is generated to deliver m in P.
- Intervals: Considering a set of intervals I, where each interval $A \in I$ is a set of messages $A \subset M$ sent by a mobile host $p = Part(A)$, defined using the mapping $Part$: $I \rightarrow P$. Where a^- y a^+ are *begin* and *end* messages respectively belonging to the interval A, and according to

the sequential order of the elements in *Part(A)*, for all $m \in A : a^- \neq m$ and $a^+ \neq m$ implies that $a^- \rightarrow m \rightarrow a^+$.

- Points: Discrete messages sent by a mobile host p. Where a discrete message m is threated as an endpoint of type *begin* and *end*, defined by the relation $a^- = a^+$.

Furthermore, there are two communication levels on a cellular network. The first one is called inter-base and is formed by the base stations communicating each other. The second level is intra-base and it is formed by the mobile hosts communicating with its corresponding base station. At this level, mobile hosts count with limited processing and storage capabilities, and of course the bandwidth on wireless channels is limited [14].

According to the architecture of a cellular network [1], synchronization protocols have to consider the following characteristics:

- Processing and storage cost for mobile hosts should be minimum
- The overhead sent on wireless channels should be low and
- The protocols should be scalable with respect to the number of mobile hosts

In this work we consider reliable communication channels, this means a message can not be dropped neither altered. Instead of that, we take into account unpredictable random delays to reach the challenge of message synchronization.

2.2 Background and Definitions

Causal ordering is implemented to assure that a message sent gets delivered according to the time it was created with respect to other messages, even in a scenario with unpredictable random delays in the communication channels. It is based on the "happened before" relation introduced by Lamport [24].

Definition 1. The causal relation denoted by $\rightarrow$, satisfying the following conditions:

1. If a and b are events of the same process, y a was created before b, then $a \rightarrow b$.
2. If a is the message sent by a process, and b is the receipt of the same message by another process, then $a \rightarrow b$.
3. If $a \rightarrow b$ y $b \rightarrow c$, then $a \rightarrow c$.

By using Definition 1, we conclude that a pair of events is simultaneous "$a \| b$", if and only if $\neg(a \rightarrow b \vee b \rightarrow a)$.

Definition 2. The Inmediate Dependecy Relation [21] denoted by "$\downarrow$" (IDR):

$$e \downarrow e' \Leftrightarrow [(e \rightarrow e') \wedge \forall e'' \in E, \neg(e \rightarrow e'' \rightarrow e')]$$

Thus, an event e has an inmediate dependency relation with another event e', if and only if there is not another event e'' that is part of the set of events E, such as e precedes e'' and e'' at the same time precedes e'.

Broadcast Causal Delivery. The causal broadcast delivery for a communication group (*broadcast*) based on IDR, is defined according to the following:

If $\forall$ *m,m' $\in$ M, send(m)$\downarrow$send(m')*$\Rightarrow$ $\forall p \in P$: *delivery(p,m)$\rightarrow$delivery(p,m')*
then
$\forall$ *m,m' $\in$ M, send(m)$\rightarrow$send(m')*$\Rightarrow$ $\forall p \in P$: *delivery(p,m)$\rightarrow$delivery(p,m')*

The Parcial Causal Relation (**PCR**) [15]. Having a subset $M' \subseteq M$. PCR induced by M' considers the subset of events $E' \subseteq E$ according to the *send* or *delivery* events in M'. PCR is defined according to the the following:

Definition 3. The **causal relation** '$\rightarrow_{E'}$' is the minor partial order relation that satisfies the next properties:

1. $\forall$ participants $p \in P$, the restrictions of $\rightarrow_{E'}$ and $\rightarrow$ to the events E'_p coincide: $\forall e,e' \in E'_p$: e $\rightarrow$ e' $\Leftrightarrow$ $e \rightarrow_{E'} e'$
2. $\forall$ messages $m \in M'$ and $j \in P$, the sending of m precedes its delivery to j: $j \in P \Rightarrow$ *send(m)$\rightarrow_{E'}$ delivery(j,m).*

The Happened-Before Relation (HBR) on intervals. This is, the causal and the simultaneous relations [22]. The causal relation for intervals is defined as it follows:

Definition 4. The relation **HBR** denoted by "$\rightarrow_I$", is given if the next conditions are accomplished:

1. $A \rightarrow B$ if $a^+ \rightarrow_{E'} b^-$
2. $A \rightarrow B$ if $\exists C | (a^+ \rightarrow_{E'} c^- \wedge c^+ \rightarrow_{E'} b^-)$

This means a^+ and b^- are the endpoints (*end* and *begin*) of A and B, while c^- and c^+ (*begin* and *end* respectively) correspond to C. On the other hand, $\rightarrow_{E'}$ is the partial causal order of $E' \subseteq E$. In this case, E', is a composed subset by the end messages of an interval I.

Definition 5. Two intervals A and B are considered simultaneous and denoted by "|||", if the next condition is accomplished:

$$A \;|||\; B \Rightarrow a^- \;||\; b^- \wedge a^+ \mid b^+$$

Thus, an interval A is considered to be emitted "at the same time" in relation to the interval B. The last definition of causal delivery based on endpoints for intervals is defined as it follows:

Definition 6. Causal Broadcast Delivery for intervals based on the endpoints:

If $(a^+,b^-) \in A \times B$, $send(a^+) \rightarrow_{E'} send(b^-) \Rightarrow \forall p \in P$, $delivery(p, a^+) \rightarrow_{E'} delivery(p, b^-)$ then
$\forall p \in P$, $delivery(p,A) \rightarrow_I delivery(p,B)$

In addition, our protocol uses a model called logical mapping presented in [22], which allows to implement a synchronization mechanism in runtime, considering continuous and discrete data. This work translates a temporal scenario of multimedia data according to the identification of logical precedence relations, in specific in terms of the happened-before relation. By using this model, our protocol avoids the need of a global reference.

With the objective of identifying the temporal relations between interval-interval, interval-point and point-to-point, the next logical mappings are implemented as it follows in table 1.

Logical mapping	**Expressed on endpoints**
Precedes: $A \rightarrow_I B$	$a^+ \rightarrow b^-$
Simultaneous: $C \mid\mid\mid D \rightarrow_I B$	$c^- \mid\mid d^-,\ c^+ \mid\mid d^+$
Ends: $A \rightarrow_I (C \mid\mid\mid D$	$a^+ \rightarrow c^-,\ a^+ \rightarrow d^-,$ $c^- \mid\mid d^-,\ c^+ \mid\mid d^+$
Starts: $(C \mid\mid\mid D) \rightarrow_I B$	$c^- \mid\mid d^-,\ c^+ \mid\mid d^+,$ $c^+ \rightarrow b^-,\ d^+ \rightarrow b^-$
Overlaps: $A \rightarrow_I (C \mid\mid\mid D) \rightarrow_I B$	$a^+ \rightarrow c^-,\ a^+ \rightarrow d^-,$ $c^- \mid\mid d^-,\ c^+ \mid\mid d^+,$ $c^+ \rightarrow b^-,\ d^+ \rightarrow b^-$

Table 1: Logical mappings expressed on endpoints.

With the five relations identified in table 1 (*precedes*, *simultaneous*, *ends*, *starts* and *overlaps*), we can represent all the possible temporal relations for multimedia data. The resulting logical mappings are translated into synchronization specification units in order to reach our objective for multimedia synchronization.

3 Results

In this section we present our protocol that is based on the *Synchronization Protocol for Continuous Media* presented by Lopez et al. [14]. Our work is an extension of the cited protocol that carries out the synchronization of continuous data only; we extend this implementation to create

a protocol that takes into account the synchronization of heterogeneous data (continuous and discrete) at runtime in mobile distributed systems. In the next subsections we describe the protocol composition and how it works, as well as the simulations we carried out to analyze our protocol in terms of efficiency and effectiveness with respect to the control information and synchronization error generated.

3.1 Protocol Composition

3.1.1 Structures of data

Every mobile host p initialized, storages and uses the next structures:

- *mes_received(p)* used as a counter and incremented when p receives a causal or FIFO message.
- *mes_sent(p)* used as a counter and incremented when p sends a causal or FIFO message.
- *CausalMesDM(p)* used as a counter and incremented when p receives a causal message.
- *FIFOMesDM(p)* used as a counter and incremented when p receives a FIFO message.
- $\Phi(p)$ is a structure of bits. Where the bits in $\Phi(p)$ are the set of messages M in relation to p.

Every base station (BS) initialized, storages and uses the following structures:

- *mes_sent(BS)* used as a counter and incremented when a base station sends a new causal or FIFO message in its cell.
- *causalMes_sent(BS)* used as a counter and incremented when a base station sends a causal message.
- *fifoMes_sent(BS)* used as a counter and incremented when a base station sends a FIFO message.
- *VT(BS)* is a vector time. It controls the number of messages received for each mobile host *VT(BS)[i]* identified by i, in the communication group.
- *CI(BS)* is a structure of tuples *(i, t, d)*; where i is the identifier of the mobile host, t is the logical clock of i and *d=causalMes_sent(BS)*.
- *last_fifo(BS)* is a structure that storages tuples *(i, t, f)* for FIFO messages, where *f=fifoMes_sent(BS)*.

The messages m, count with a different structure depending on the scenario where they are sent, from mobile hosts or base stations.

- A message sent by a mobile host on wireless communication channels to its base station, is marked as m and has the following structure *m=(i, t, mes_received(p), TP, data, h(m))*, where:
  - i, is the mobile host p sender.
  - t, is a logical clock or a counter for messages sent by p.
  - *TP* is a field to identify the type of message (*begin, end, cut, fifo* and *discrete*)
  - $h(m)$ is a structure of bits created before sending a message m by p_i, this structure is a copy of $\Phi(p)$

- A message transmitted among base stations is marked as *bs(m)*, and has the following structure *bs(m)=(i, t, TP, data, H(m))*, where:
  - *data* is the content of a specific message
  - *H(m)* is a tuple, composed by (*i, t*), which represents the causal past of p_i. This structure is generated at the moment a message is transmitted by a base station.
- A message received at a base station BS_l from a mobile host $p \in BS_l$, and transmitted by the same BS_l in its cell, is marked as *intra*(m) which has the following structure *intra(m)=(i, t, TP, data, h'(m))*, where:
  - *h'(m)* is a structure of bits, generated before sending a message m by a base station.
- A message *bs(m)* received at a BS_l from another base station. Which is transmitted in its cell to the mobile hosts, consists in a tuple with the following structure *inter(m)=(i, t, TP, data, h'(m))*.

3.1.2 Specification

In order to initialize mobile hosts and base stations, we define data structures for each node to ensure causal ordering, see table 2.

Table 2: Initialization of entities in the communication group

For each mobile host p_i, i: 1...n;	**For each base station BS_l, l:1...s;**
$\Phi(p_i) \leftarrow \emptyset$	$VT(BS_l)[i]=0 \forall i : 1...n$;
$bit \leftarrow$1b	$CI(BS_l) \leftarrow \emptyset$
$mes\_received(p_i)=0$	$last\_fifo(BS_l) \leftarrow \Phi$
$mes\_sent(p_i)=0$	$H(m) \leftarrow \Phi$
$CausalMesDM(p_i)=0$	$mes\_sent(BS_l)=0$
$FIFOMesDM(p_i)=0$	$causalMes\_sent(BS_l)=0$
	$fifoMes\_sent(BS_l)=0$

There are four main events that occur in our scenario, which are used to synchronize messages at sending and reception by mobile hosts and base stations. These events are triggered when a mobile host p_i creates and sends a new message m to the communication group. We describe the procedures that each entity carry out in the next listings.

The first event is the sending of a message m by a mobile host to its base station. It is necessary to mention that m is delivered to the others mobile hosts in the communication group, this is achieved by retransmitting m to every base station from the local base station, we use the procedure in listing 1 to do this.

Listing 1: Creation and emission of message *m=(i, t, TP, data, h(m))* by mobile host p_i

```
1  The first step is the creation of a message TP, for multimedia data (begin, fifo, cut, end or discrete)
2  The counter mes_sent(p_i) is incremented in one
3  The structure of bits Φ(p_i) is copied to h(m)
4  In the case TP equals to end or cut, do:
5          Construction of m=(i=id_mobile,t=mes_sent(p_i),TP,CausalMesDM(p_i),data,h(m))
6          If TP is equals to end, then mark this mobile host as Active = false
7          Mark Φ(p) as empty, Φ(p) ← ∅
8          and the counter CausalMesDM(p_i) is incremented in one
9  In the case TP equals to begin or discrete, do:
```

```
10          Construction of m=(i=id_mobile,t=mes_sent(p_i),TP,CausalMesDM(p_i),FIFOMesDM(p_i),data,h(m))
11          If TP is equals to begin, then mark this mobile host as Active = true
12          Mark Φ(p) as empty, Φ(p) ← ∅
13          and the counter CausalMesDM(p_i) is incremented in one
14  In the case TP equals to fifo, do:
15          Construction of m=(i=id_mobile,t=mes_sent(p_i),TP,data,h(m)← ∅)
16  After this, send the message m to the base station of p_i
```

The second event is the reception of m at the local base station. When a base station receives a message m from a mobile host belonging to its cell, we use the procedure in listing 2 to order and retransmit m.

Listing 2: Reception of *m=(i, t, TP, CausalMesDM(p_i), data, h(m))* and sending of messages *intra(m)* and *bs(m)* by BS_l

```
1   First, we check if the incoming message belongs to a mobile host in the cell, if this is true then proceed
2   Check if the logical clock t of m is the consecutive number of VT(BS_l)[i]+1
3   If t is different to VT(BS_l)[i]+1, then the message is put in a wait queue and delayed until is deliverable at BS_l
4   Else if t is equals to VT(BS_l)[i]+1, then we apply different cases according to the type of message TP:
5           In the case TP equals to end or cut, do:
6                   for each bit in h(m):
7                           if exists the tuple (k,t,d) in CI(BS_l), such that d is equals to CausalMesDM(p), then
8                                   Add a bit to the structure h'(m) and
9                                   Add the tuple (k,t) to the structure H(m)
10                  Increment causalMes_sent(BS_l) in one, and
11                  Add a new tuple (i,t,causalMes_sent(BS_l)) to CI(BS_l)
12          In the case TP equals to begin or discrete, do:
13                  for each bit in h(m):
14                          if exists (s,r,d) in last_fifo(BS_l), such that d≤FIFOMesDM(i) and s!=j, then
15                                  Add a bit to the structure h'(m) and
16                                  Add the tuple (k,t) to the structure H(m)
17                  for each tuple (k,t,d) in CI(BS_l):
18                          if exists (s,r) in H(m) such that k=s and d≤CausalMesDM(p) then
19                                  Add the tuple (s,r) to the structure H(m)
20                  Increment causalMes_sent(BS_l) in one, and
21                  Add a new tuple (i,t,causalMes_sent(BS_l)) to CI(BS_l)
22          In the case TP equals to fifo, do:
23                  Mark H(m) and h'(m) as empty, H(m) ← h'(m) ← ∅
24                  Increment fifoMes_sent(BS_l) in one, and
25                  Add a new tuple (i,t,fifoMes_sent(BS_l)) to the structure last_fifo(BS_l)
26  Create intra(m) = (i,t=mes_sent(BS_l),TP,data,h'(m)) and bs(m) = (i,t,TP,data,H(m))
27  After this, send intra(m) to the mobile hosts in the cell of BS_l and send bs(m) to every base station in BSG.
```

The third event is the reception of a message *bs(m)* sent by a base station to another base station and subsequently the creation of a message *inter(m)*, we use the code in listing 3 to perform this.

Listing 3: Reception of *bs(m)=(i, t, TP, data, H(m))* and sending of message inter(m) by BS_l

```
1   Now, we check if the incoming message does not belong to a mobile host in the cell, if true then proceed
2   If t is different to VT(BS_l)[i]+1, then bs(m) is put in a wait queue and delayed until is deliverable at BS_l
3   Also, for each tuple (s,x) in H(m) such that x is greater than VT(BS_l)[s], bs(m) is put in the wait queue
4   When the message is delivered, the vector time VT(BS_l)[i] is incremented in one
5   In the case TP equals to begin, cut, end or discrete, do:
6           Increment causalMes_sent(BS_l) in one
7           For each tuple (x,y) in H(m):
8                   if exists (k,t,d) in CI(BS_l) or in last_fifo(BS_l), such that x==k and y==t, then
9                           Add a bit to h'(m)
10                  Add a new tuple (i,t,causalMes_sent(BS_l)) to CI(BS_l)
11  In the case TP equals to fifo, do:
12          Mark H(m) and h'(m) as empty, H(m) ← h'(m) ← ∅
13          Increment fifoMes_sent(BS_l) in one, and
14          Add a new tuple (i,t,fifoMes_sent(BS_l)) to the structure last_fifo(BS_l)
15  Increment mes_sent(BS_l) in one
16  Create inter(m) = (i,t=mes_sent(BS_l),TP,data,h'(m))
17  After this, send inter(m) to the mobile hosts in the cell of BS_l
```

In the fourth event we describe what happens when the mobile hosts receive a message *intra(m)* or *inter(m)* from the local base station, we use the following procedure in listing 4.

Listing 4: Reception of messages *intra(m)* or *inter(m)=(i, t, TP, data, h'(m))* at mobile host p_j

```
First, check if the logical clock t of the incoming message is equals to mes_received(p_j)+1
If the condition is false, then the incoming message is put in a wait queue and delayed until is deliverable at p_j
Else, the message is delivered and the counter mes_received(p_j) is incremented in one
In the case TP equals to begin, cut, end or discrete, do:
        Increment CausalMesDM(p_j) in one
        For each bit in h'(m):
                Remove a bit from Φ(p_j)
        Add a bit to Φ(p_j)
        If Active is true (mobile transmitting continuous data) and TP is different from cut or begin, then
                Send the next message as cut
In the case TP equals to fifo, do:
        Increment FIFOMesDM(p_j) in one
With this, a message sent by a mobile host p_i can be delivered to p_j and all the mobile hosts P in the group
```

3.1.3 Description of the Protocol

In order to describe how our synchronization protocol works, the scenario in figure 2 is presented. There are two base stations BS_1 and BS_2 and three mobile hosts. The mobile host p_1 is in the cell of BS_1, while p_2 and p_3 are in the cell of BS_2.

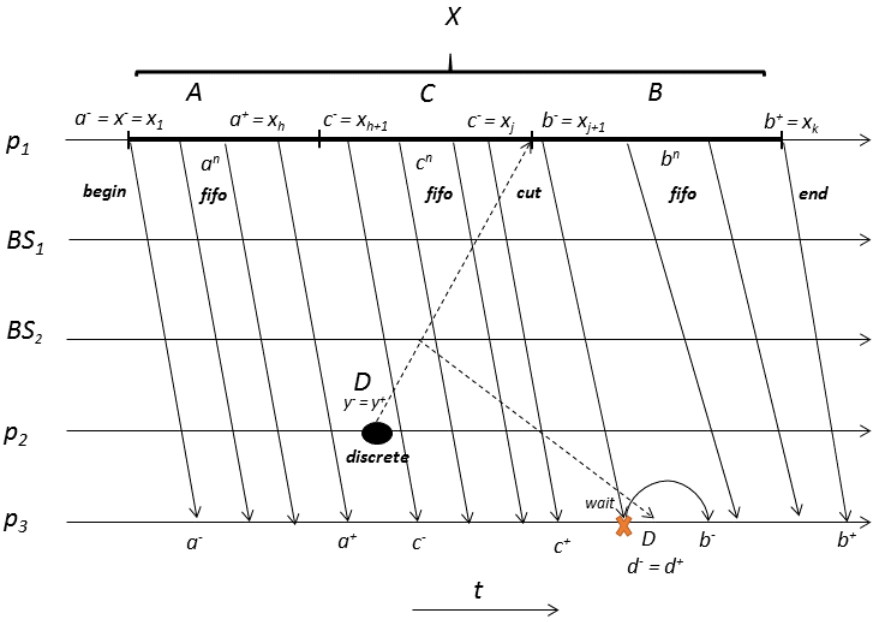

Figure 2: Example of the synchronization protocol for heterogeneous data transmitted at runtime.

In the example, p_1 starts sending continuous data, an interval denoted by X and the first message marked as *begin* with left causal limit a subinterval A: a^- equals to $x^- = x_1$, the next messages sent by p_1 are *fifo* messages denoted by a^n. The right causal limit of subinterval A is denoted by a^+ equals to x_h, this endpoint of A, is determined at the reception of the *fifo* message (x_h) in the mobile host p_2 before the sending event of the *discrete* message D equals to $y^- = y^+$ or $d^- = d^+$. With this last event we can determine the set of messages that compose the subinterval

$A = \{x_1, x_2, ..., x_h\}$. After A, the next message sent by p_1, is denoted by a subinterval C which left causal limit $c^- = x^{h+1}$, the next *fifo* messages are denoted by c^n till message $c^+ = x_j$ that precedes the delivery of d coming from p_2 to p_1, with this last message we can determine the elements that compose the subinterval $C = \{x_{h+1}, x_{h+2}, ..., x_j\}$, with the reception of D in p_1, this mobile host creates a new message of *cut* type, that is the first element of subinterval B denoted by $b^- = x_{j+1}$, the next messages are marked as *fifo* and denoted by b^n till the creation of $b^+ = x_k$ that is the last message of B which is an *end* message of the interval X in messages sent by p_1. With this, we can determine the set of messages that compose the subinterval $B = \{x_{j+1}, x_{j+2}, ..., x_k\}$. The messages received in the mobile host p_3 are synchronized according to the the temporal dependencies, in example, the message sent by p_1 and marked as *cut* also denoted by b^- can not be delivered in p_3 since this mobile host did not receive the *discrete* message denoted by D that immediately precedes b^-. This is $D \rightarrow b^-$. For this reason, the message b^- is placed in the wait queue and delivered till D is received in p_3.

Hence, the resultant logical mapping obtained in the example is $A \rightarrow_I (C \,|||\, D) \rightarrow_I B$. This means that to carry out the delivery of the interval X and the discrete message denoted by D equals to $y^- = y^+$ or $d^- = d^+$, in mobile host p_3, and based on the causal dependencies corresponding to each message, we must ensure that:

- $delivery(p_3, a^+) \rightarrow_{E'} delivery(p_3, c^-)$
- $delivery(p_3, a^+) \rightarrow_{E'} delivery(p_3, d^-)$
- $delivery(p_3, c^+) \rightarrow_{E'} delivery(p_3, b^-)$ and
- $delivery(p_3, d^+) \rightarrow_{E'} delivery(p_3, b^-)$

3.1.4 Correction of Synchronization Error

We implemented a mechanism which can deal with all the messages transmitted considering the synchronization error generated through a correlation of each message. When a base station receives a causal message, two corrective actions are performed. The first one is to add to a wait queue all those messages that violate the causal dependencies, and the second action is to discard the messages that exceed the maximum synchronization error permitted. To achieve this, the following function is defined:

wait(bs(m)): $\forall$ *m' = (k,s')* $\in$ *H(m),*{

- $s' \leq VT(BS_l)[k]$ or
- $deadline_l(m') \leq current\_time(BS_l)$ }

Where $deadline_l(m')$ returns the time when m' expires at the base station that manages this message. This function applies for each message $m" = (k,s")$ such that $s" > VT(BS_l)$ and $s" < s'$ then is also discarded. The time limit for message delivery is determined using a distributed method [20], in this work we do not show how deadline of a message is calculated, but in the cited work you can check the details.

3.1.5 Overhead Analysis

This protocol is also based on the Immediate Dependency Relation protocol presented in [21], in our implementation the size of the control information (overhead) for each message that is sent through communication channels derives according to the number of parallel messages that immediately precede a causal message m. This means a *fifo* type message does not content overhead. Since the structure *H(m)* in the wired channels only uses recent messages that immediately precede a message m, the cardinality for the overhead generated is given by the cardinality of the number of concurrent messages in the communication group, which vary between 0 and n-1 ($0 \leq | H(m) | \leq n$-1), where n is the number of mobile hosts in the communication group. In an optimal case, the overhead is equal to 0 and in the worst case the overhead is n-1. In the case of the wireless communication channels among base stations and mobile hosts, the overhead vary between 0 and n-1 *bits* ($0 \leq | h(m) | \leq n$-1), as well as the best and the worst case are 0 and n-1 *bits* respectively.

3.2 Simulations

In order to evaluate our protocol, we simulate different scenarios of a cellular network. The scenario we use is a mobile distributed system running over a cellular network. We propose a generic scenario composed by four base stations and n mobile hosts distributed proportionally (see Figure 1 in section 2). The communication is set through messages passing. The mobile hosts perform group communication, hence, each message generated and sent by a mobile host is delivered in all the mobile hosts belonging to the communication group. Additionally, this scenario does not take into account a global reference neither shared resources because each mobile host is independent.

The simulations are performed in Sinalgo [29]. This framework, allow us to configure parameters in the network such as mobility, reliability, asynchronous channels, transmission delays, and creation of base stations

and mobile hosts. Furthermore it is possible to implement a synchronization protocol for each node in the network, as well as modify the behavior of sending and reception of messages.

3.2.1 Measuring the synchronization error

One important aspect to evaluate in a scenario of group communication that transmits heterogeneous data in runtime, is the synchronization error experienced by mobile hosts. For example, in a videoconference the maximum synchronization error permitted between sources is 400 ms [4], while in a more restrictive scenario with an application using video with overlayed text and images the error permitted is 240 ms in order to maintain Quality of Service (QoS) [18]. We define two equations to measure the synchronization error, which are presented below:

$$receive\_sync\_error_l(m) = \frac{\Sigma_{(k,s)\epsilon H(m)} receive\_time_l(m) - last\_rcv_l(k)}{\mid H(m) \mid} \tag{1}$$

$$delivery\_sync\_error_l(m) = \frac{\Sigma_{(k,s)\epsilon H(m)} delivery\_time_l(m) - last\_rcv_l(k)}{\mid H(m) \mid} \tag{2}$$

The equation 1 *receive_sync_error$_l$(m)* calculates the synchronization error when a message m is received at a base station BS_l. The equation 2 *delivery_sync_error$_l$(m)* calculates the synchronization error at the delivery after applying our synchronization protocol.

Where the functions *last_receive$_l$(k)* and *last_deliver$_l$(k)* return the time when the last message of a mobile host k was received or delivered. The functions *receive_time$_l$(m)* and *delivery_time$_l$(m)* return the time when m was received or delivered, respectively.

3.2.2 Results Analysis

In order to show the effectiveness and efficiency of our protocol, eighteen experiments have been carried out. We consider scenarios with four base stations and 10, 20, 50, 100, 200, 300, 400, 500 and 600 mobile hosts distributed proportionally at base stations; the number of mobile hosts was selected according to an example of a congested 3G network with UMTS technology, which has support for 126 active users using a bandwidth of 500 kbits/s [31], while for a congested 4G LTE network, the supported active users is about 200 using a bandwidth of 5 MHz, with a download speed of 100 Mbits/s [33]. Each scenario is composed by two configurations, the first one considering a transmission delay between 50-150 ms, which represents a non-congested cellular network. For the second configuration we set a transmission delay of 50-450 ms, which represents a heavy network load [6]. We consider the same transmission delay for the communication channels. In all the experiments, we measure the synchronization error at the reception and delivery of a message, the overhead attached per message in the communication channels and storage overhead.

The mobile hosts only transmit one type of data (continuous or discrete). For continuous data, we use MPEG-4 and H.263 encoded videos from the TKN-Project [17]. We start sending a single *begin* message using the first frame denoted by I, followed by limited number *fifo* messages and finally a single *end* message when another frame I is found, and again. In this case we consider sending frames at a rate of 25 frames per second. For discrete data we use plain text strings with a sending rate of one message every 500 ms to 1 s. In our scenarios fifty percent of the total mobile hosts

send continuous data and the other fifty percent send discrete data. The results obtained are presented below.

Results of synchronization error The results of the first experiment are presented in figure 3, we analyze the scenario with 10 mobile hosts considering a transmission delay of 50-150 ms (non-congested network). The synchronization error at message reception is measured by using equation 1, see figure 3 the points in blue color. On the other side, after applying our protocol, the synchronization error at message delivery is measured by using equation 2, see figure 3 the points in green color. With this we demonstrate that the synchronization error allowed in an ideal scenario (240 ms) for a non-congested network, is maintained below the limit in all the messages and reduced by an important amount for each message with our synchronization protocol.

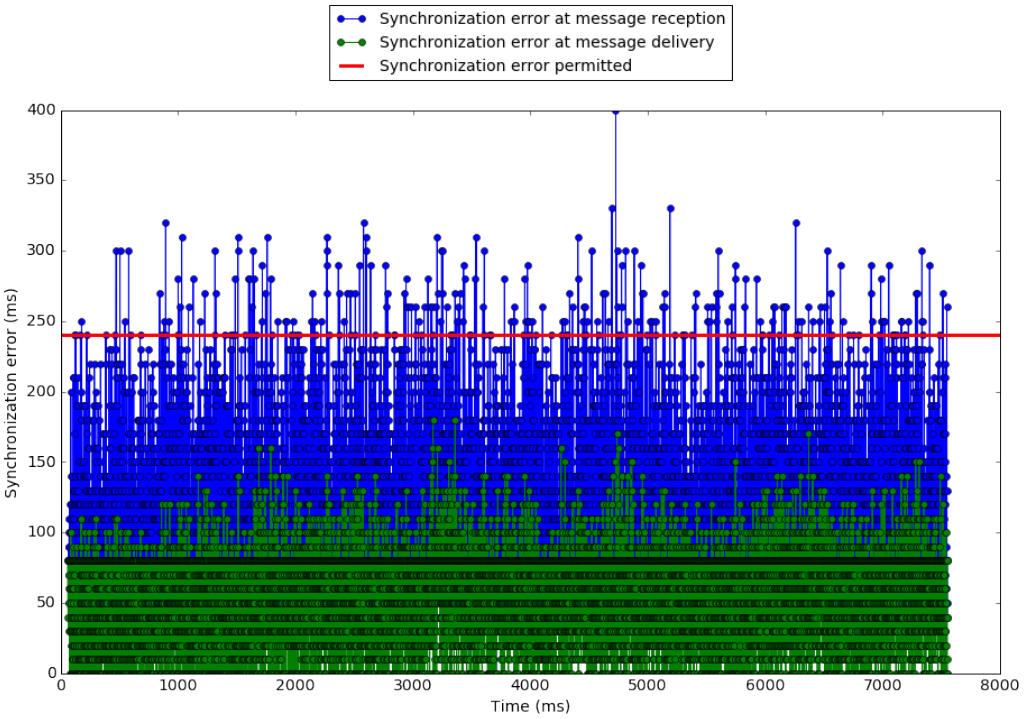

Figure 3: Synchronization error at reception and delivery of messages considering a transmission delay of 50-150 ms in a scenario with 10 mobile hosts.

For the second experiment in figure 4, the synchronization error at reception is above 240 ms in 2% of the messages. After applying our protocol, we correct all the messages below 240 ms.

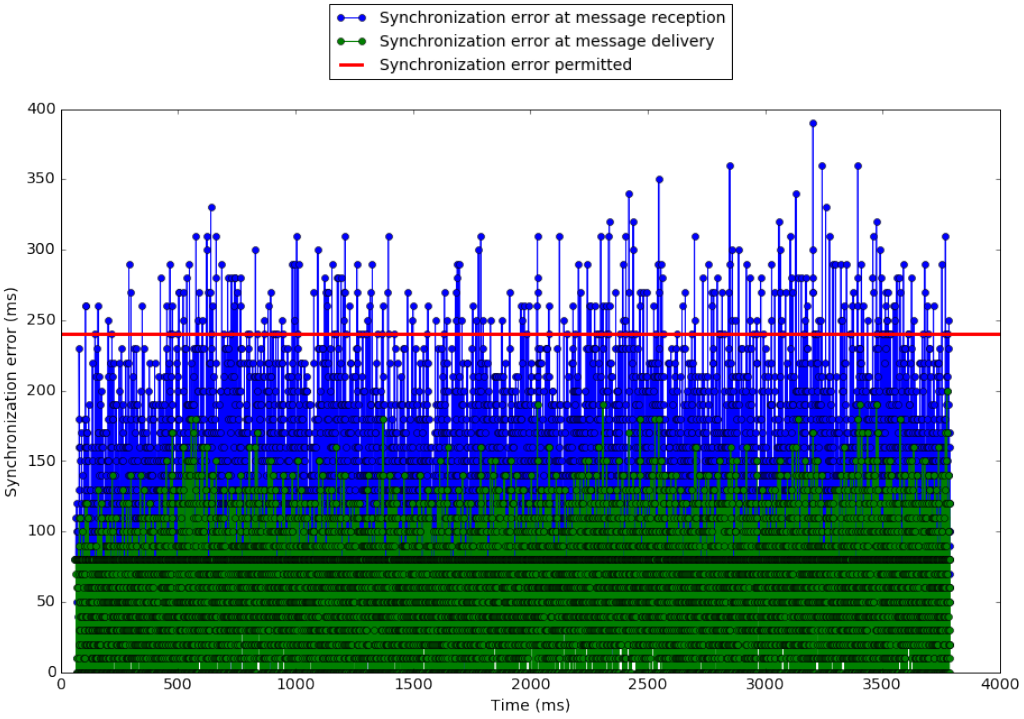

Figure 4: Synchronization error at reception and delivery of messages considering a transmission delay of 50-150 ms in a scenario with 20 mobile hosts.

For the next set of experiments in figures 5 to 11, identified by number of mobile hosts, and a transmission delay of 50-150 ms. We notice that the same behavior explained before is performed for each scenario. If the transmission delay is set according to a non-congested network, the points (messages) above the synchronization error limit (240 ms) at message reception are a total of 1-3%, after applying our protocol at message delivery, we correct the synchronization error for all the messages according to the limit permitted in this restricted scenario.

Furthermore, when applying our protocol the number of messages that exceed 80 ms is about 7-12 % which is feasible in terms of QoS, if we consider a scenario that requires lip synchronization for audio and video where the synchronization error permitted is 80 ms [18]. In another scenario using loosely coupled audio (e.g. dialog mode with various participants) and audio synchronization for correlated video and animation, the limit permitted is 120 ms [18], with our protocol for a non-congested network, the synchronization error of the messages above this limit is about 1-2%.

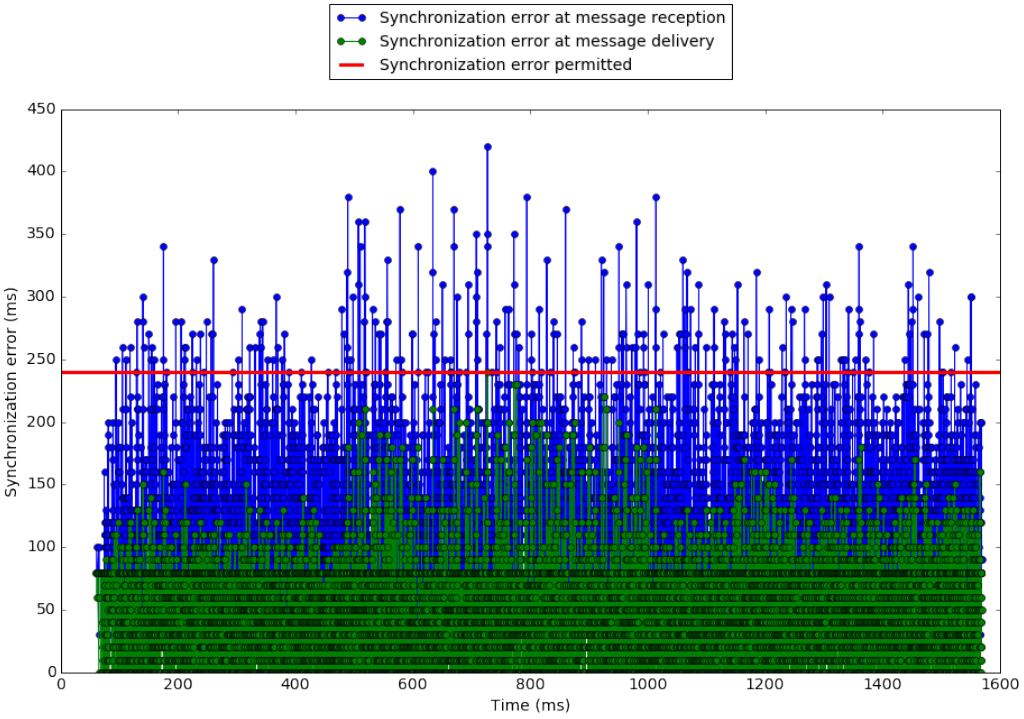

Figure 5: Synchronization error at reception and delivery of messages considering a transmission delay of 50-150 ms in a scenario with 50 mobile hosts.

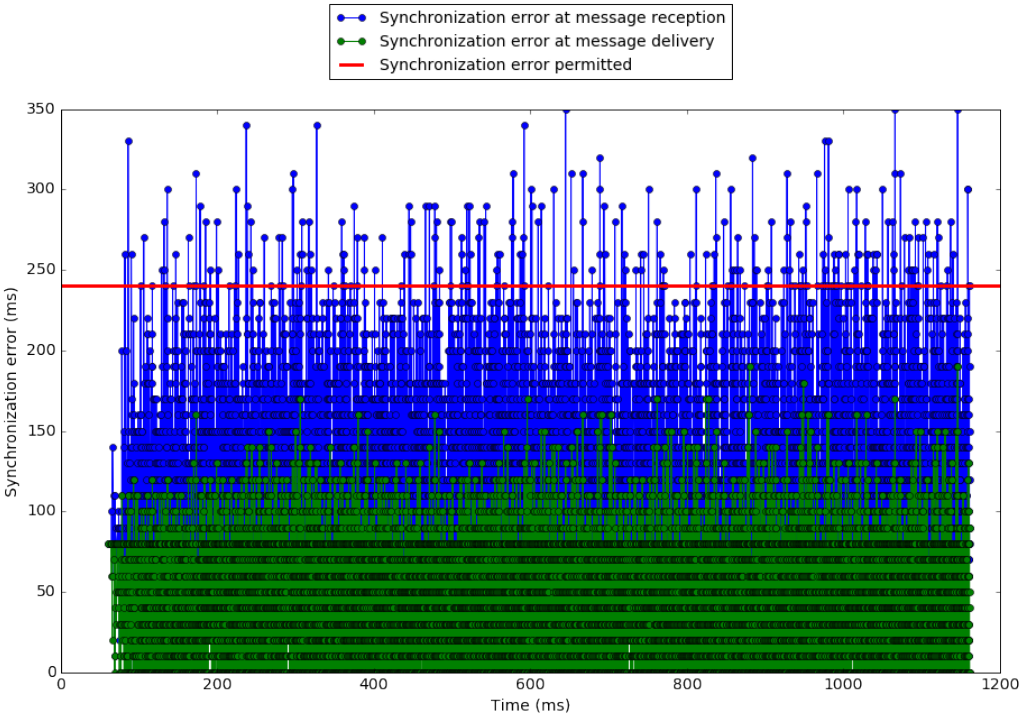

Figure 6: Synchronization error at reception and delivery of messages considering a transmission delay of 50-150 ms in a scenario with 100 mobile hosts.

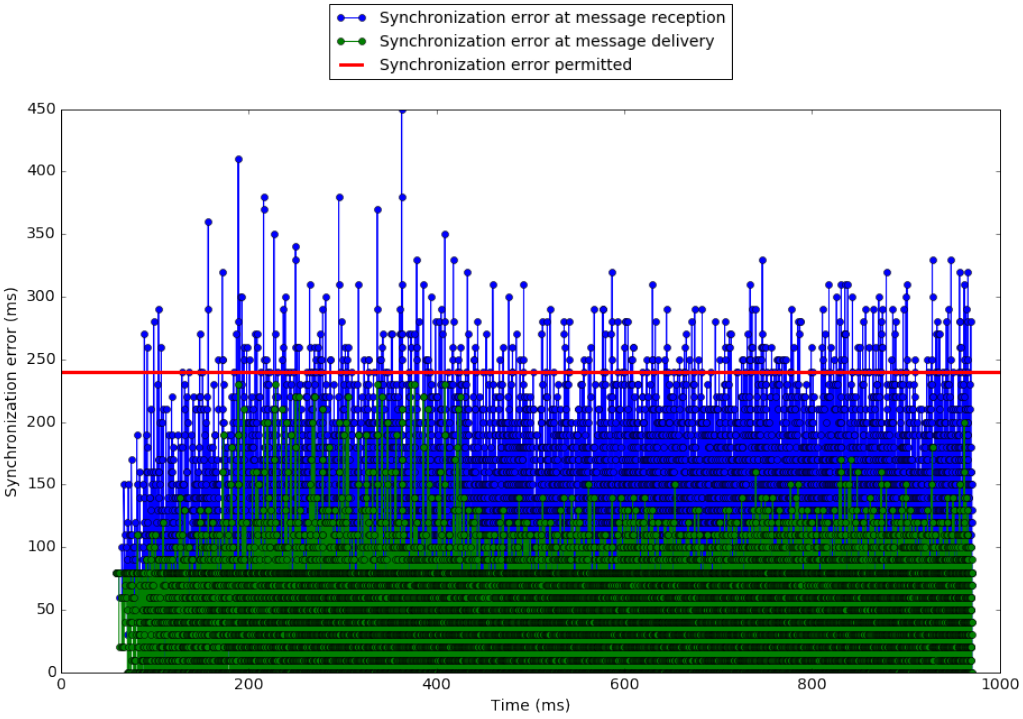

Figure 7: Synchronization error at reception and delivery of messages considering a transmission delay of 50-150 ms in a scenario with 200 mobile hosts.

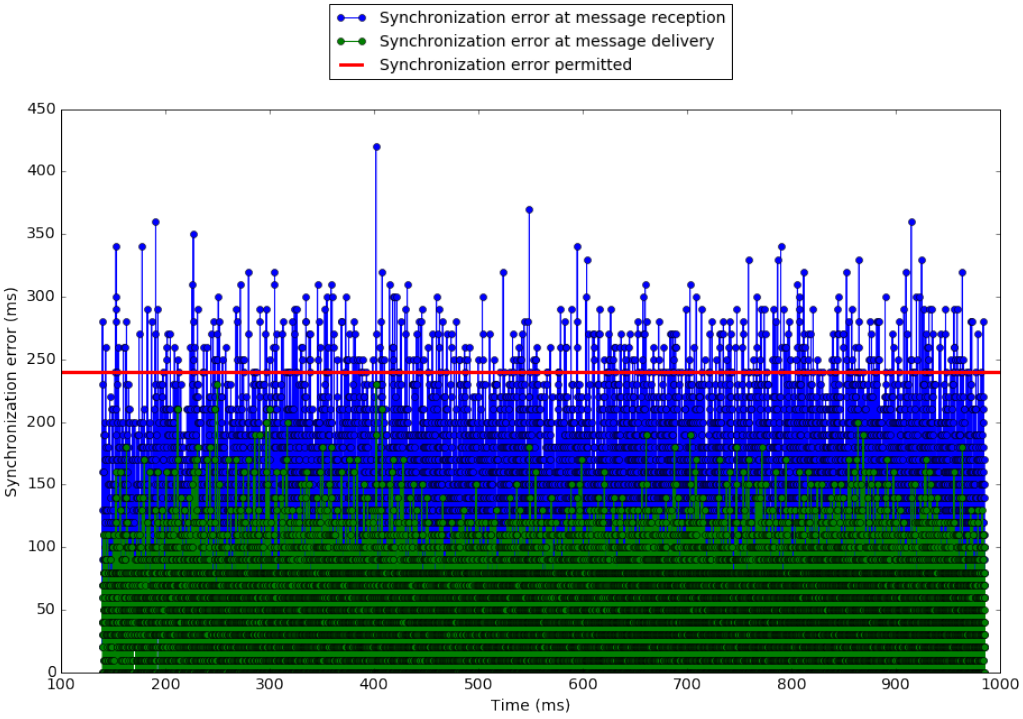

Figure 8: Synchronization error at reception and delivery of messages considering a transmission delay of 50-150 ms in a scenario with 300 mobile hosts.

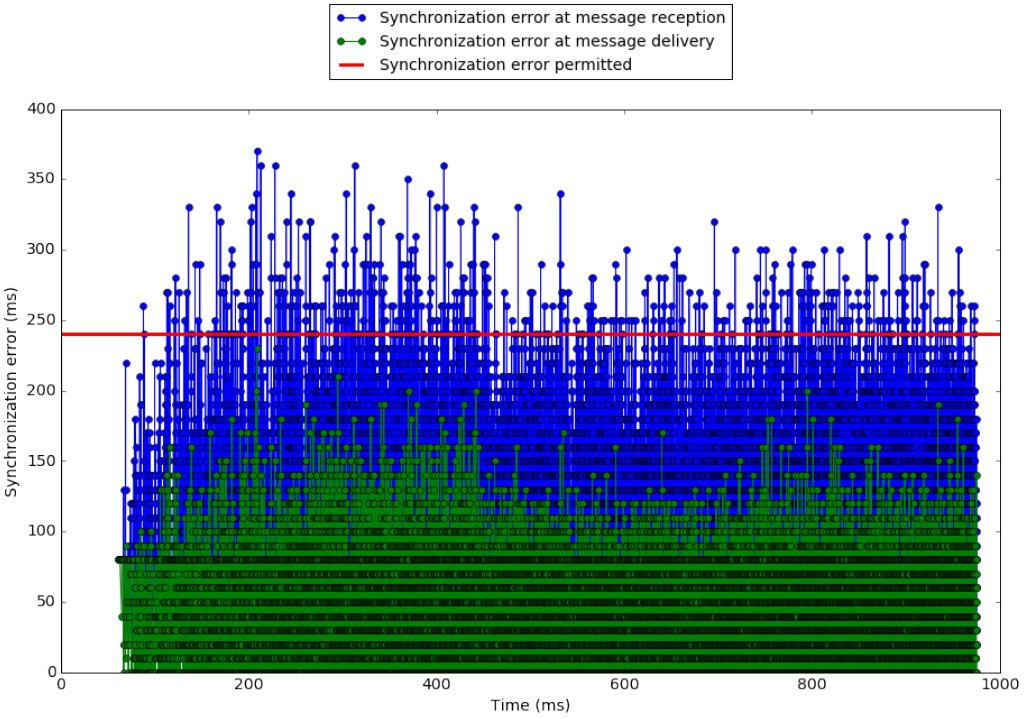

Figure 9: Synchronization error at reception and delivery of messages considering a transmission delay of 50-150 ms in a scenario with 400 mobile hosts.

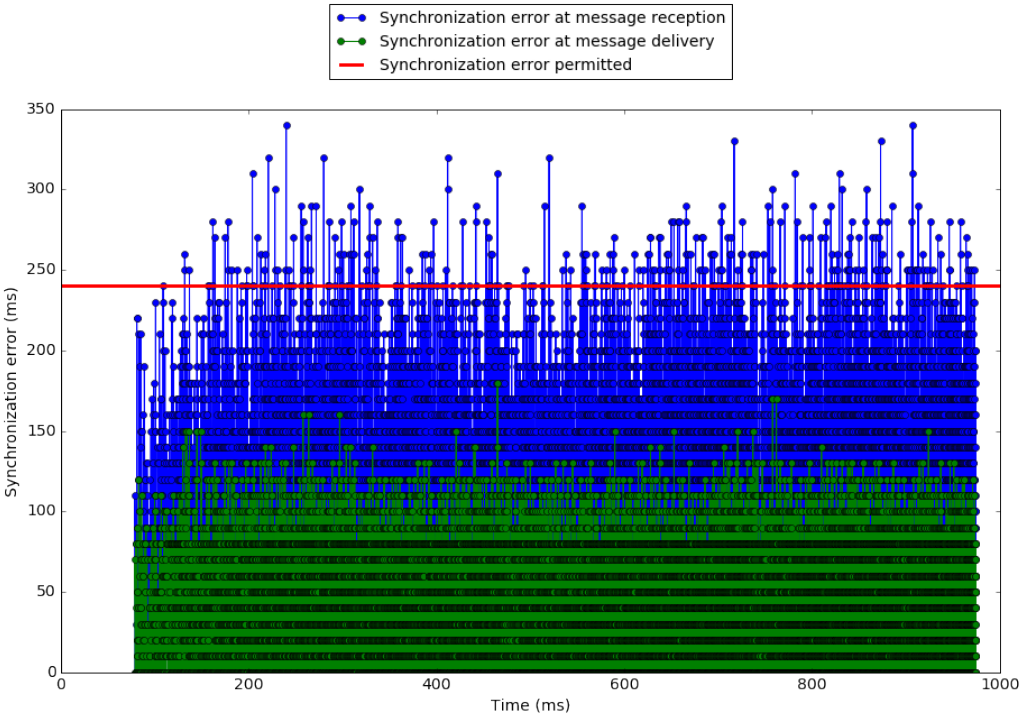

Figure 10: Synchronization error at reception and delivery of messages considering a transmission delay of 50-150 ms in a scenario with 500 mobile hosts.

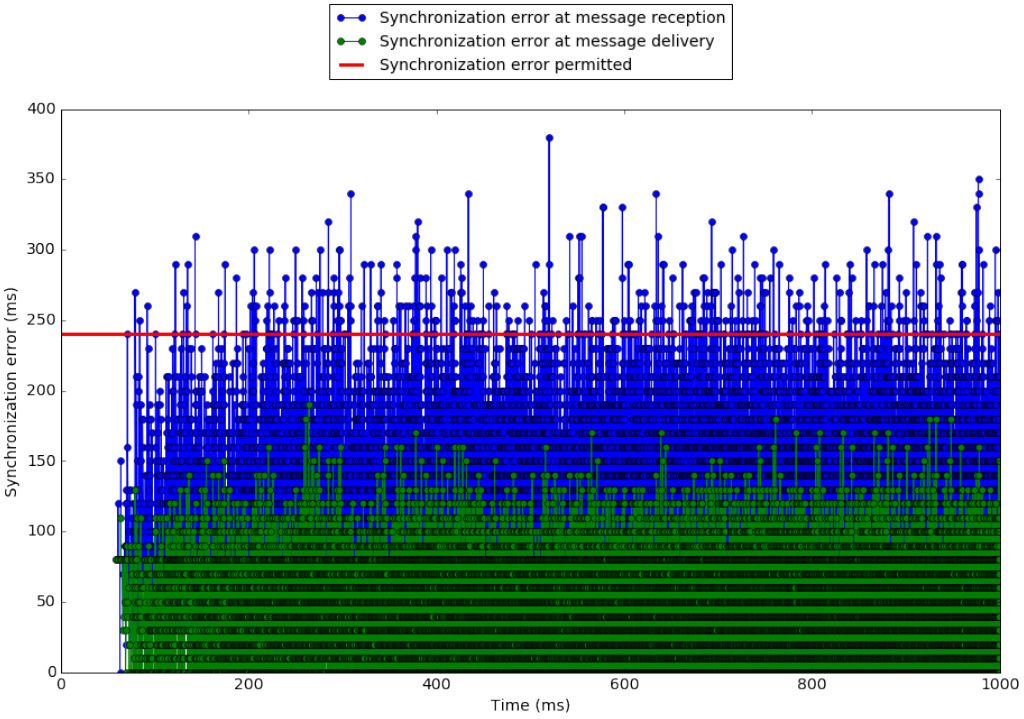

Figure 11: Synchronization error at reception and delivery of messages considering a transmission delay of 50-150 ms in a scenario with 600 mobile hosts.

When configuring a heavy network load with a transmission delay of 50-450 ms. In the experiment 10 that we present in figure 12 with 10 mobile hosts, the synchronization error at message reception measured by using equation 1, exceeds the maximum permitted of 400 ms in 26% of the messages, this limit is indicated by a horizontal red line. This delay for each message can produce the loss of coherency among the data transmitted in a videoconference [4]. After applying our protocol, the synchronization error measured by using equation 2, is maintained below 400 ms in 99% of transmitted messages.

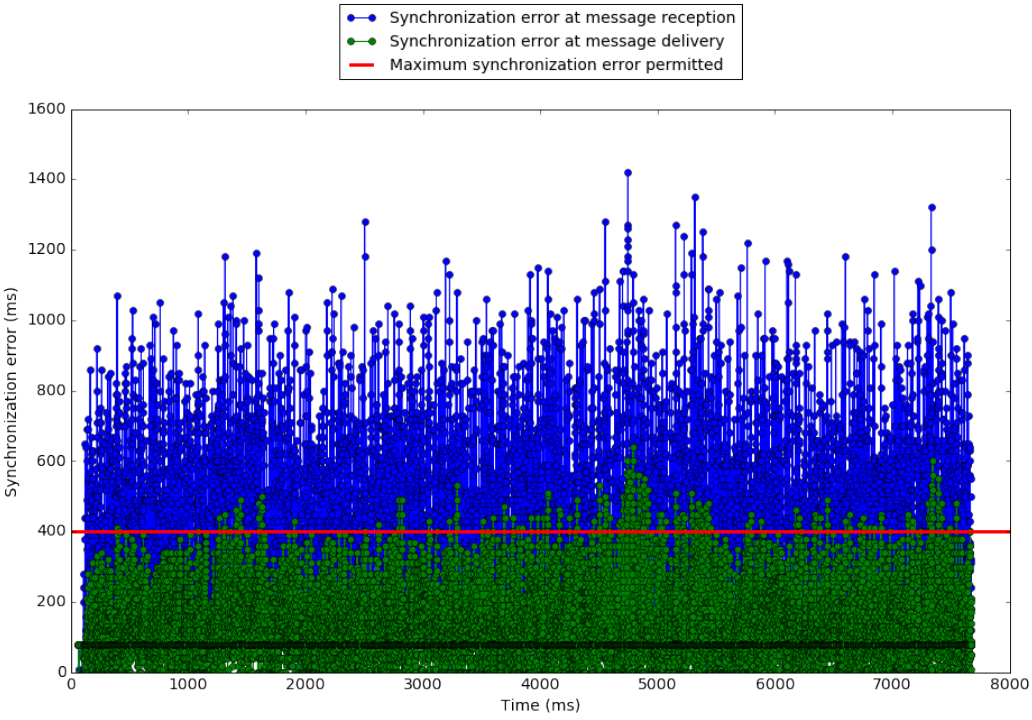

Figure 12: Synchronization error at reception and delivery of messages considering a transmission delay of 50-450 ms in a scenario with 10 mobile hosts.

For the next experiment in figure 13, 36% of the messages received by the base station BS_l are above the maximum synchronization error permitted. With our protocol, we reduce the amount of messages above the limit to 3%.

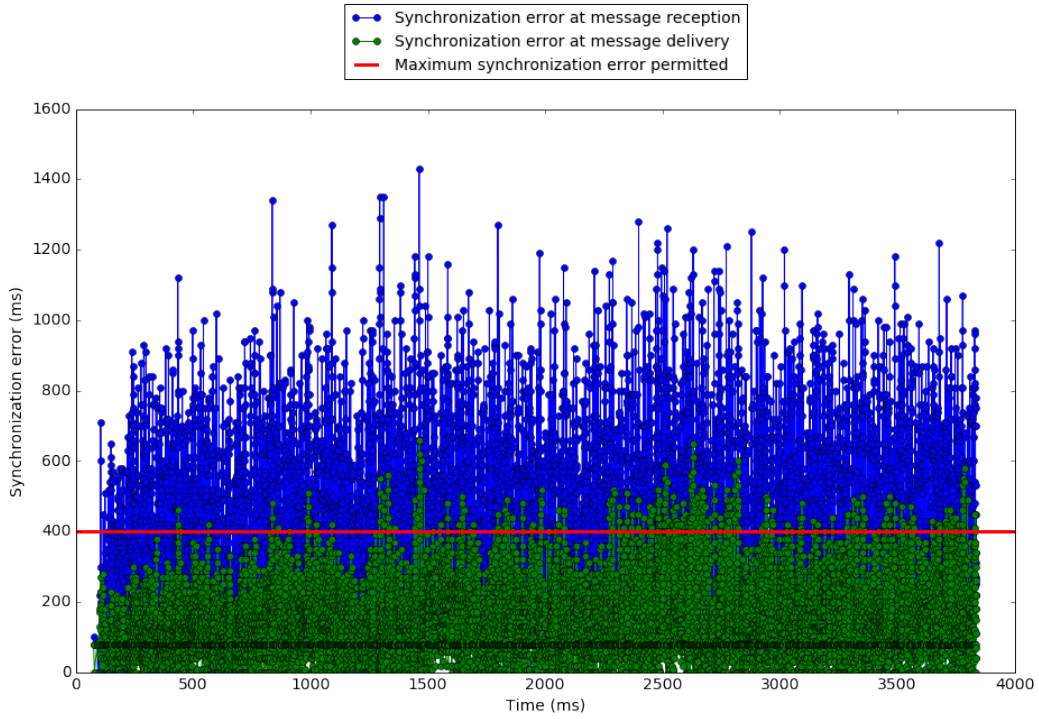

Figure 13: Synchronization error at reception and delivery of messages considering a transmission delay of 50-450 ms in a scenario with 20 mobile hosts.

For the next set of experiments considering a heavy network load, we measure the maximum synchronization error in figures 14 to 20. We analyzed the results for the different scenarios in number of mobile hosts and we conclude that in a heavy network load, then the maximum synchronization error permitted (400 ms) exceed by the messages at reception is between 25-40% of the total messages generated, when applying our protocol we reduce the maximum synchronization error limit exceed by messages to 1-5% of the total messages.

In a scenario for multimedia synchronization, with an application using video, loosely coupled audio (e.g. background music), with non-overlay text and loosely coupled images (e.g. slide show), the synchronization error permitted in order to maintain QoS is 500 ms [18]. Without using our protocol the messages above this limit are about 25-30% of the total, after applying our protocol to correct the synchronization error, the messages above the limit of 500 ms are 1% at most.

On the other side, if we consider a more restrictive scenario where the synchronization error limit allowed is 240 ms in order to maintain QoS [18], the points above this limit at message reception are between 40-60% without our protocol. With our protocol, the messages above this synchronization error are between 10-20% depending on the scenario, where the maximum error reached correspond to 300 or more mobile hosts.

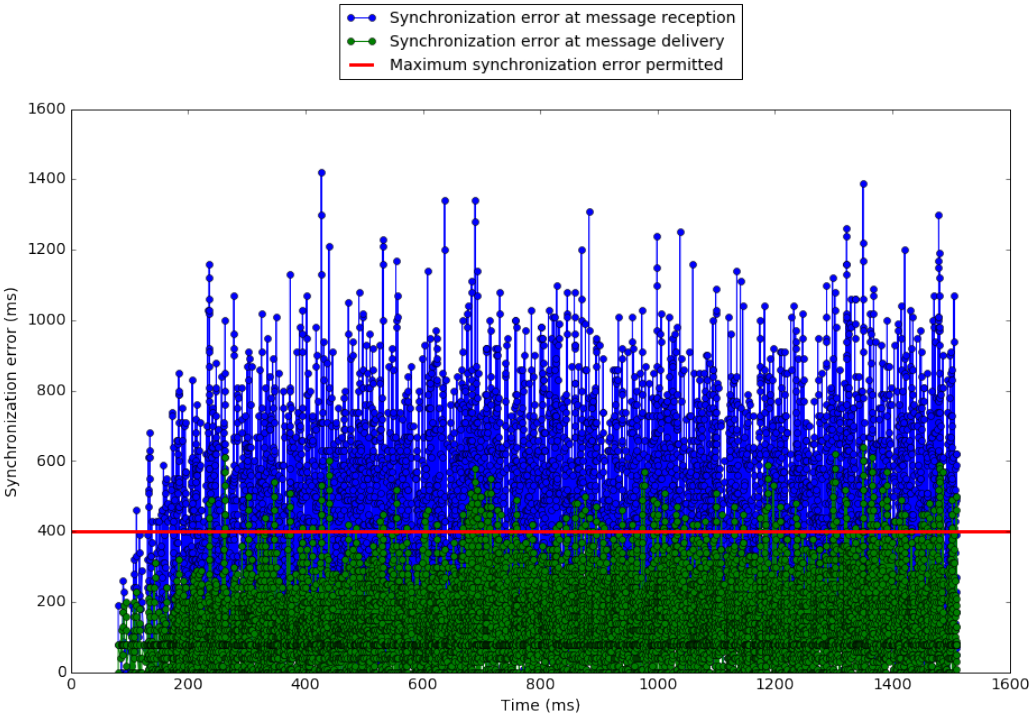

Figure 14: Synchronization error at reception and delivery of messages considering a transmission delay of 50-450 ms in a scenario with 50 mobile hosts.

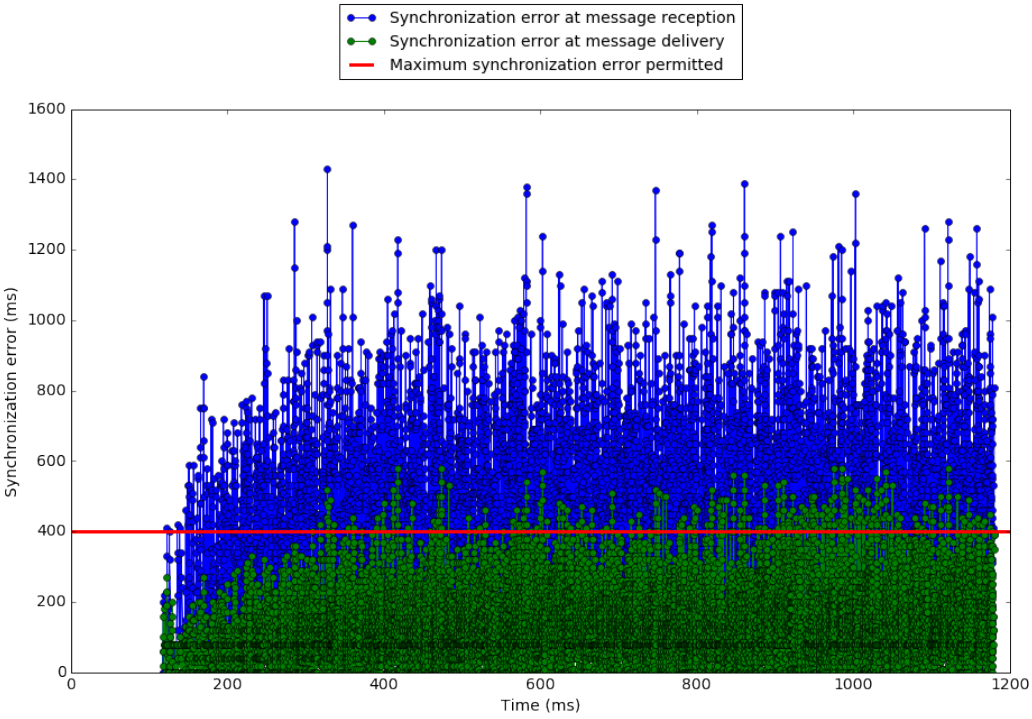

Figure 15: Synchronization error at reception and delivery of messages considering a transmission delay of 50-450 ms in a scenario with 100 mobile hosts.

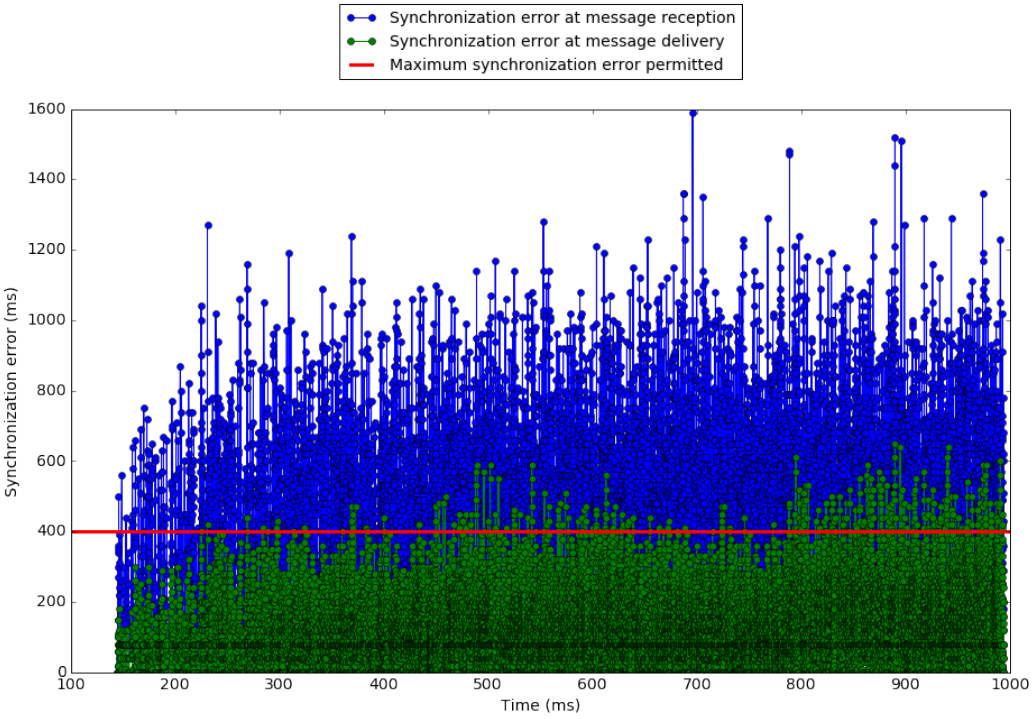

Figure 16: Synchronization error at reception and delivery of messages considering a transmission delay of 50-450 ms in a scenario with 200 mobile hosts.

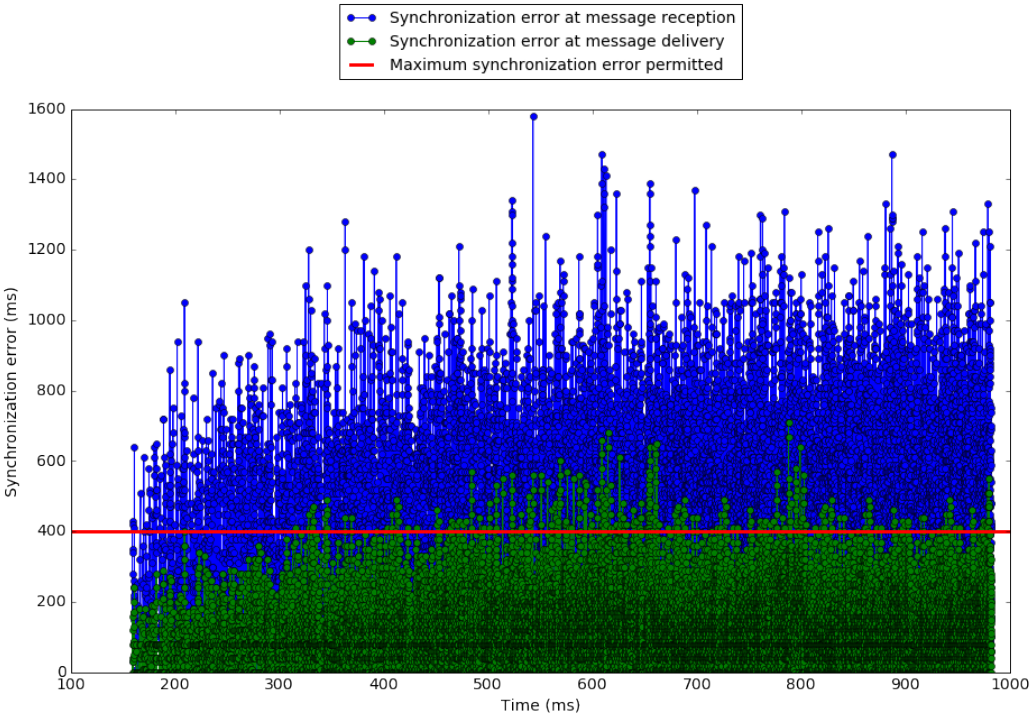

Figure 17: Synchronization error at reception and delivery of messages considering a transmission delay of 50-450 ms in a scenario with 300 mobile hosts.

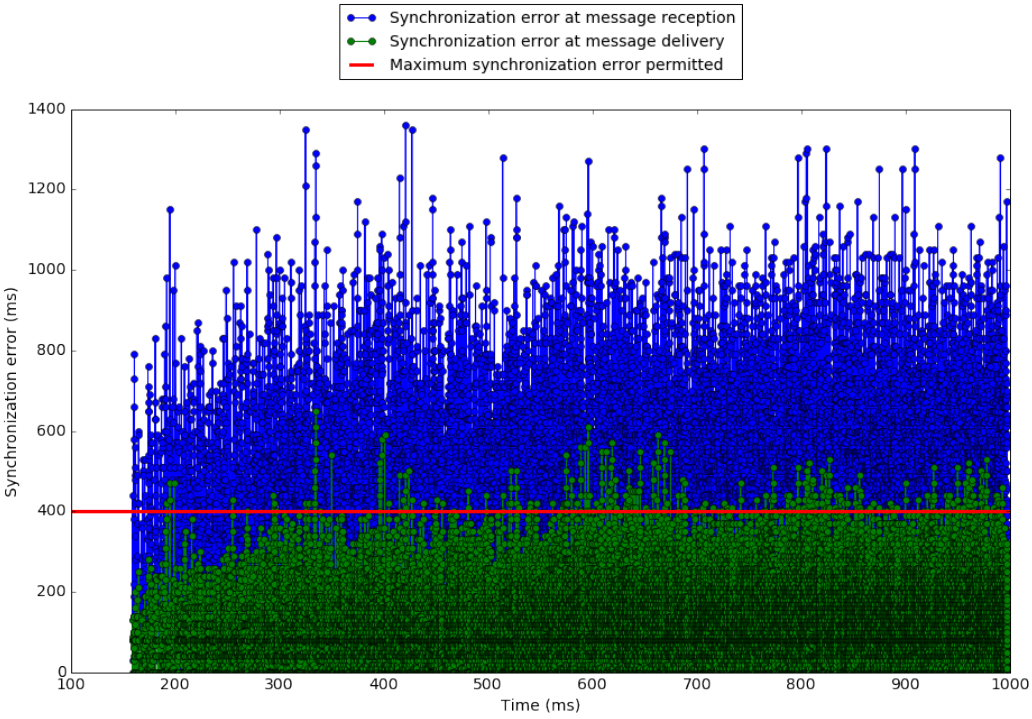

Figure 18: Synchronization error at reception and delivery of messages considering a transmission delay of 50-450 ms in a scenario with 400 mobile hosts.

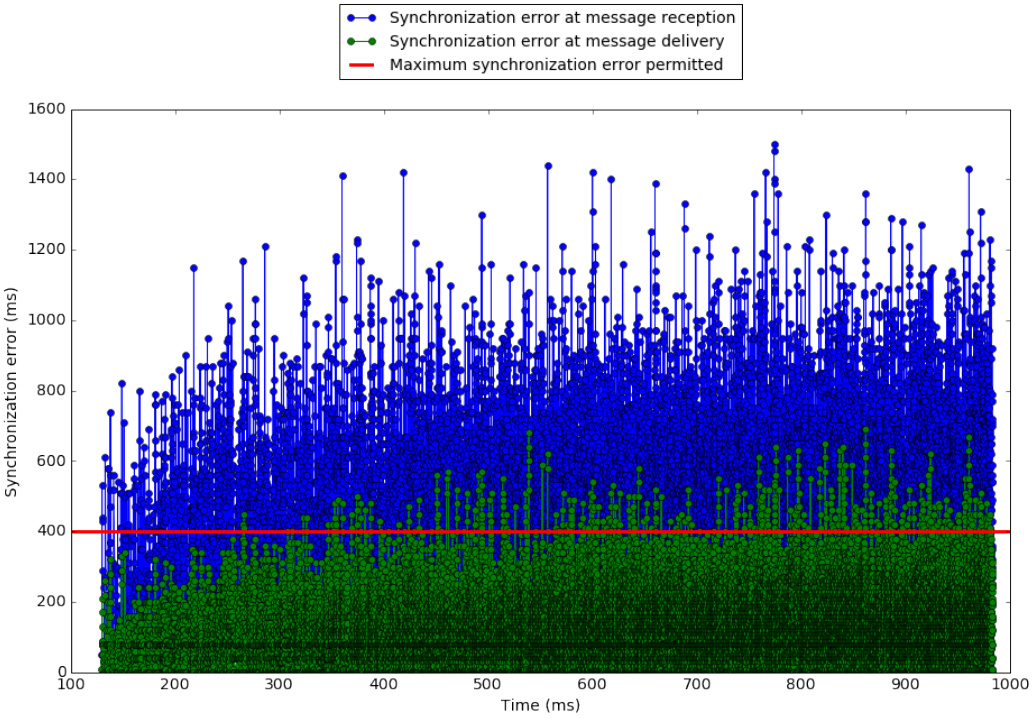

Figure 19: Synchronization error at reception and delivery of messages considering a transmission delay of 50-450 ms in a scenario with 500 mobile hosts.

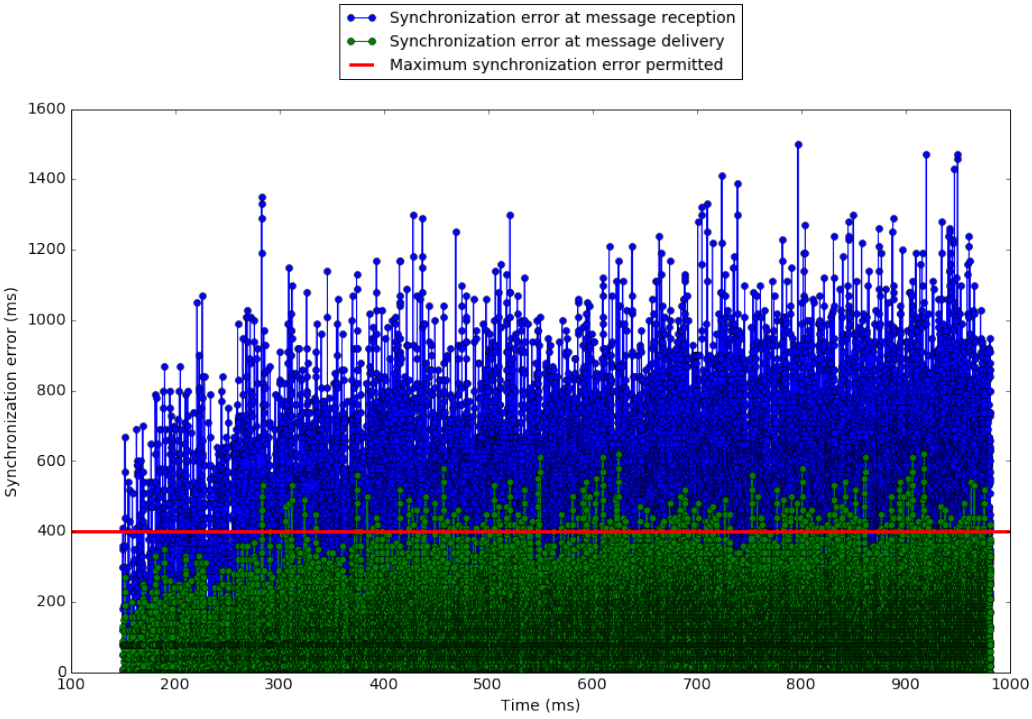

Figure 20: Synchronization error at reception and delivery of messages considering a transmission delay of 50-450 ms in a scenario with 600 mobile hosts.

Results of overhead On the other hand, our protocol implements the logical mapping expressed on *endpoints*. In this work, the endpoints are sent as causal messages (*begin*, *end*, *cut* and *discrete*). As mentioned in section 3.1.5, the control information is only added to causal messages transmitted when is required. We present the results obtained for transmission and storage overhead in in table 3, the table includes the experiments listed before and is divided by the two configurations in the transmission delay. The simulation time for results shown is about 2-3 minutes.

The following results presented in table 3 in terms of structures of data, show that the overhead per message transmitted is below than 39 bytes in wired channels, while in wireless channels is less than 9 bits. This is achieved with our asymmetric synchronization protocol by using the $H(m)$ structure of bytes for messages sent among base stations and a structure of bits $h(m)$ for messages sent by mobile hosts and base stations at wireless channels (see section 3.1.1). In addition, the average storage overhead in mobile hosts is between 9 to 24 bytes depending on the scenario, this is achieved by only using four counters and the structure of bits $\Phi(p)$ to manage the control information which derives on the number of parallel messages that immediately precede a causal message. This is an advantage in order to maintain the scalability feature in communication groups, since we also take into account the capabilities of base stations for the bulk of processing and storage to achieve synchronization. We also highlight that in mobile hosts, the processing cost is very low since we use bitwise operations and basic arithmetic operations.

Table 3: Message and storage overhead in causal messages used for our protocol

Experiment	Number of causal and fifo messages		Average overhead per causal message		Average storage overhead	Total overhead of causal messages	
Number of mobile hosts	Causal messages	fifo messages	Wired channel (bytes)	Wireless channel (bits)	Mobile host (bytes)	Wired channel (bytes)	Wireles channe (bytes)
Transmission delay (50-150ms)							
10	19,069	21,690	1.09	0.45	10.67	20,801	1,073
20	21,368	42,805	2.23	0.76	9.35	47,636	2,030
50	27,340	107,011	2.89	1.30	9.01	79,016	4,443
100	45,451	205,380	4.28	1.51	9.49	194,442	8,579
200	112,419	362,625	8.53	2.40	10.20	959,148	33,721
300	171,036	515,253	18.01	3.98	10.31	3,080,868	92,606
400	222,999	678,837	23.07	4.33	9.66	5,145,492	160,677
500	296,454	809,019	27.52	5.76	10.44	8,159,256	147,457
600	388,789	931,575	35.12	7.12	10.51	13,654,270	210,508
Transmission delay (50-450ms)							
10	18,946	21,524	1.25	0.51	9.55	23,762	1,208
20	21,060	42,842	1.81	0.83	9.81	38,077	2,185
50	30,169	103,536	3.61	1.23	10.09	109,029	4,628
100	43,056	206,831	9.42	2.16	10.35	405,731	11,637
200	109,938	359,646	13.10	3.73	13.08	1,439,818	51,192
300	176,709	507,996	17.65	4.68	12.27	3,118,958	103,277
400	236,067	659,547	27.15	6.36	18.88	6,408,472	187,739
500	269,064	780,552	31.71	7.58	20.76	8,532,483	255,038
600	375,669	921,435	38.85	8.14	23.48	14,594,280	382,214

In addition, we decided to compare our synchronization protocol with the Real-time Transport Protocol (RTP) since the works in the state of the art for multimedia synchronization are based on extensions for RTP, see section 4. We present the table 4 with the overhead required in communication channels for RTP to synchronize data according to its causal dependencies in the same scenarios we presented for our experiments. It is important to mention that the overhead attached per message with RTP is about 12 bytes by default [30]; however, not all the header content in the packet is used to synchronize data in this protocol. For this reason, we only take into account the next fields: *sequence number*, which gives information about the number of received message, *time stamp*, which represents a mark of time of the information (e.g. wall clock) and *SSRC* that is the identifier from the process that sends the message. This information gives a total of 10 bytes of overhead attached per message.

Moreover, RTP works together with the Real-Time Transport Control Protocol (RTCP), this protocol sends periodic information in order to synchronize data among a group of participants. After analyze this protocol, we take into account the next fields: *SSRC*, *NTP timestamp most significant word*, *NTP timestamp least significant word* and *RTP timestamp*, the overhead generated considering these fields is a total of 16 bytes, which represents the static part of the packet RTCP SR. In addition, RTCP adds a report block, this packet creates a structure for each

process belonging to the communication group; for this reason we decided to include the next fields of the report block: *SSRC_1*, *interarrival jitter* and *delay since last SR*, which represents a total of 12 bytes of overhead for each structure added. The equation 3 is used to measure the overhead generated by RTCP:

$$RTCPOverhead = 16 + (12 * (n - 1)) \quad (3)$$

Where 16 is the static part of the packet, 12 is the overhead added per structure and n is the number of mobile hosts in the scenario. We include RTCP packets in the scenario every 5-7.5 seconds according to the number of participants in the communication group [27] and also considering our scenario with a simulation time of 2-3 minutes.

On the other side and with the objective of measuring and comparing our work, we consider all the structures in the fifo and causal messages sent over wireless and wired channels. For fifo messages we include the next fields: *sequence number of the message*, *identifier of the process* and *type of message*, these fields represent a total of 6 bytes per fifo message. In the case of causal messages, apart from the fields listed above, we also include: *counter of causal messages*, this field adds 2 more bytes, resulting a total of 8 bytes per causal message; moreover, in the causal messages we have to include the resulting overhead for the tuples in the structure $H(m)$ and the bits generated in the structure of bits $h(m)$, see table 3. The results of our comparison are presented in table 4.

Table 4: Total overhead of our protocol in comparison to RTP/RTCP

Number of mobile hosts	**Total of messages**	**Total overhead of our Protocol (bytes)**	**Total overhead of RTP (bytes)**	**Total overhead of RTCP (bytes)**	**Percentage of reduction at wireless channels**	**Total percentage of reduction**
Transmission delay (50-150 ms)						
10	40,759	304,566	407,590	44,640	37.25%	32.65%
20	64,173	477,440	641,730	175,680	47.42%	41.59%
50	134,351	944,245	1,343,510	1,087,200	64.40%	61.15%
100	250,831	1,798,909	2,508,310	4,334,400	76.55%	73.71%
200	475,044	4,067,971	4,750,440	17,308,800	85.91%	81.56%
300	686,289	7,633,280	6,862,890	38,923,200	90.06%	83.33%
400	901,836	11,163,183	9,018,360	69,177,600	92.30%	85.72%
500	1,105,473	15,532,459	11,054,730	108,072,000	93.81%	86.96%
600	1,320,364	22,564,539	13,203,640	155,606,400	94.72%	86.63%
Transmission delay (50-450 ms)						
10	40,470	305,682	404,700	44,640	37.26%	31.97%
20	63,902	465,794	639,020	175,680	47.50%	42.83%
50	133,705	976,224	1,337,050	1,087,200	64.23%	59.73%
100	249,887	2,002,802	2,498,870	4,334,400	76.63%	70.69%
200	469,584	4,528,390	4,695,840	17,308,800	85.96%	79.42%
300	684,705	7,683,883	6,847,050	38,923,200	90.03%	83.21%
400	895,614	12,442,029	8,956,140	69,177,600	92.28%	84.08%
500	1,049,616	15,623,346	10,496,160	108,072,000	94.02%	86.82%
600	1,297,104	23,510,456	12,971,040	155,606,400	94.71%	86.05%

The results in the table reflect the overhead reduction in transmission over communication channels, even when the communication group grows up in number of mobile hosts. The main reason of this, is because we only add control information to the causal messages as is explained in subsection 3.1.5. Furthermore, we manage the highest reduction of overhead in wireless communication channels to accomplish the approach of our asymmetric protocol based on the argument that the bandwidth is limited at wireless communication channels. In the case of the RTP and RTCP protocols, the amount of overhead increments in order O(n), this means that when the number of process in the communication group grow up, RTCP generates an important amount of control information according to the equation 3 and the reports should be sent frequently in order to maintain quality of service (QoS) [27]. See table 4, in specific the column about total percentage of reduction of our protocol in comparison to RTP and RTCP.

4 Discussion

The synchronization protocols can be categorized according to the type of data synchronized among a communication group that use multimedia applications. There are three main categories of protocols that we identify: discrete, continuous and heterogeneous [12]. In this section we briefly resume the advantages of each synchronization protocol in the state of the art with respect to the mentioned categories.

4.1 Synchronization Protocols for Discrete Data

The protocols of this category focus on synchronizing discrete data such as images, data and text [12]. Applications such as WhatsApp, Messenger and Skype are examples of this. In specific using the functions to talk by sending/receiving plain text messages, attaching files and emoticons. Several works of the state of the art can be classified in this category, below we present three works related to our protocol.

A Reliable Multicast Protocol for Distributed Mobile Systems [2], uses an asymmetric design, to achieve the synchronization it implements three algorithms: FIFO, causal and total. The control information in this protocol is independent of the number of mobile hosts in the communication group.

An Optimal Causal Broadcast Protocol in Mobile Dynamic Groups [5], also is implemented for group communication. It uses broadcast to deliver messages in the mobile distributed system. The *overhead* attached for each message m is O(1) because it only uses one tuple, the one from the immediate last message that precedes m.

An Efficient Delta-causal algorithm for Real-time Distributed Systems [20], proposes causal ordering for group communication based on domain over an hybrid wireless network. The overhead used for this protocol is given by O(n) where n is the number of mobile hosts in the network.

These protocols do not use a global reference, which is viable for mobile distributed systems [11]. Also [2] and [5] present an asymmetric design to delegate the bulk of processing and storage to the entities with more capabilities (base stations). In our work, we propose the synchronization of discrete and continuous data, this is the main difference in comparison to the listed protocols which can not deal with data streams.

4.2 Synchronization Protocols for Continuous Data

In this category we describe the protocols that carry out about continuous data such as audio and video among a communication group. In example, Skype for videoconferences is an application that uses continuous data, as well as IPTV streaming services [13]. The protocols related to our work are listed below.

Media Synchronization Algorithms for Wireless Systems [10], uses a mechanism called MoSync which is implemented in the application layer. This work follows an asymmetric design. It is focused on lowering the use of buffer, overload, complexity of messages and fluctuation in the transmission. The overhead has order $O(I(K+L))$ where L is the message request by a mobile host, K is the number of servers that process I requests and L is the number of rounds of $L + K$ messages.

An Extension to SDP and RTP for Media Loopback [19], implements monitoring of messages by using media loopback with acknowledgement. This extension adds control information to SDP (Sockets Direct Protocol) and RTP (Real-time Transport Protocol). It gives the possibility to ensure the delivery of continuous data and measure the performance metrics with this implementation. The overhead attached per message is about 12 bytes according to RTP.

Rtp4mux [26], adapts RTP and RTCP for synchronization of continuous data in communication groups. This extension includes new packets to RTCP and carry out the synchronization by using acknowledgment messages. The overhead attached per message is about 12 bytes.

The first work uses an asymmetric design while the other works follow an end-to-end design which is not applicable for mobile distributed systems, also the last two works use a global reference to achieve the synchronization. In spite of the properties of the first protocol, it does not take into account discrete data, this means it cannot be used in applications such as online multiuser videogames, telemonitoring or interactive videoconferences.

4.3 Synchronization Protocols for Heterogeneous Data

In this last category we present two protocols found in the state of the art that carry out about multimedia synchronization for continuous and discrete data. The works are listed below.

Inter-Destination Media Synchronization using RTP [25] and Multimedia Group Synchronization [9] from the same author, present extensions to RTP/RTCP. The overhead attached per message is about 10 bytes according to the standard, in addition to the overhead generated by RTCP according to the description in subsection 3.2.2. The disadvantages of these works are based on the use of global references to synchronize data, the principle of design is end-to-end and the overhead is given in order $O(n)$, where n is the number of process in a communication group. According to the subsection 2.1 these properties are not suitable for mobile distributed systems.

For more related works based on this approach, please refer to the next study about multimedia group and inter-stream synchronization techniques [8].

5 Conclusions

In this paper we presented the synchronization protocol for heterogeneous data to be used at runtime by communications groups in mobile distributed systems. This work ensures temporal constraints by executing temporal dependencies taking into account the causal dependencies of the multimedia data to ensure causal message ordering as viewed by the end mobile hosts. This was achieved by considering the characteristics of mobile distributed systems, such as lack of global references (e.g. wall clock or shared memory), asynchronous sources, communication through messages passing and independent devices. Moreover, we implemented an asymmetric design for our protocol. With this principle, we delegate the processing and storage cost to base stations since the mobile hosts have limited capabilities; In the case of the wireless networks with limited bandwidth, the attached control information is less than in the wired channels.

The simulations results in the different scenarios we presented, based on number of mobile hosts and transmission delay, showed that our protocol is efficient in the control information attached per message in the communication channels. Moreover, it is feasible to implement in a mobile distributed system in terms of processing and storage cost at the mobile devices. In addition, we presented experiments to show the reduction of the synchronization error in the message delivery in all cases, this gives a tolerable error according to the maximum synchronization error supported in multimedia applications. In comparison to RTP and RTCP, our protocol is scalable since the mentioned protocols attach overhead with linear growth depending on the mobile hosts members of the communication group. In our case, we only add overhead in causal messages according to the number of mobile hosts to support the synchronization over discrete and continuous data.

References

[1] Sridhar Alagar and Subbarayan Venkatesan. Causal ordering in distributed mobile systems. *IEEE Transactions on Computers*, 46(3):353–361, 1997.

[2] Giuseppe Anastasi, Alberto Bartoli, and Francesco Spadoni. A reliable multicast protocol for distributed mobile systems: Design and evaluation. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1009–1022, 2001.

[3] Shahab Baqai, M Farrukh Khan, Miae Woo, Seiichi Shinkai, Ashfaq A. Khokhar, and Arif Ghafoor. Quality-based evaluation of multimedia synchronization protocols for distributed multimedia information systems. *IEEE Journal on Selected Areas in Communications*, 14(7):1388–1403, 1996.

[4] Ivano Bartoli, Giovanni Iacovoni, and Fabio Ubaldi. A synchronization control scheme for videoconferencing services. *Journal of multimedia*, 2(4), 2007.

[5] Chafika Benzaid and Nadjib Badache. An optimal causal broadcast protocol in mobile dynamic groups. In *Parallel and Distributed Processing with Applications, 2008. ISPA'08. International Symposium on*, pages 477–484. IEEE, 2008.

[6] Michael S Borella. Source models of network game traffic. *computer communications*, 23(4):403–410, 2000.

[7] Fernando Boronat, Jaime Lloret, and Miguel García. Multimedia group and inter-stream synchronization techniques: A comparative study. *Information Systems*, 34(1):108–131, 2009.

[8] Fernando Boronat, Jaime Lloret, and Miguel García. Multimedia group and inter-stream synchronization techniques: A comparative study. *Information Systems*, 34(1):108–131, 2009.

[9] Fernando Boronat, Mario Montagud, and Juan C Guerri. Multimedia group synchronization approach for one-way cluster-to-cluster applications. In *Local Computer Networks, 2009. LCN 2009. IEEE 34th Conference on*, pages 177–184. IEEE, 2009.

[10] Azzedine Boukerche and Harold Owens Ii. Media synchronization and qos packet scheduling algorithms for wireless systems. *Mobile Networks and Applications*, 10(1-2):233–249, 2005.

[11] George F Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed systems: concepts and design*. pearson education, 2005.

[12] S Ud Din and D Bulterman. Synchronization techniques in distributed multimedia presentation. In *The Fourth International Conferences on Advances in Multimedia (MMEDIA)*, pages 1–9, 2012.

[13] E Lopez Dominguez and S E Pomares Hernandez. *Protocolo Causal de Transporte para Datos Continuos en Redes Celulares*. Editorial Academica Espanola, 2011.

[14] Eduardo Lopez Dominguez, Saul E Pomares Hernandez, Pilar Gomez Gil, Jorge de la Calleja, Antonio Benitez, and Antonio Marin-Hernandez. Intermedia synchronization protocol for continuous media using mpeg-4 in mobile distributed systems. *channels*, 7:8, 2012.

[15] Jean Fanchon, Khalil Drira, and Saul E Pomares Hernandez. Abstract channels as connectors for software components in group communication services. In *Computer Science, 2004. ENC 2004. Proceedings of the Fifth Mexican International Conference in*, pages 88–95. IEEE, 2004.

[16] Stefano Ferretti and Marco Roccetti. Fast delivery of game events with an optimistic synchronization mechanism in massive multiplayer online games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 405–412. ACM, 2005.

[17] Frank HP Fitzek and Martin Reisslein. Mpeg-4 and h. 263 video traces for network performance evaluation. *IEEE network*, 15(6):40–54, 2001.

[18] Anna Haj Hać and Cindy X Xue. Synchronization in multimedia data retrieval. *International Journal of Network Management*, 7(1):33–62, 1997.

[19] K Hedayat, N Venna, P Jones, A Roychowdhury, C SivaChelvan, and N Stratton. An extension to the session description protocol (sdp) for media loopback. *Work in Progress*, 2008.

[20] S E Pomares Hernandez, E Lopez Dominguez, G Rodriguez Gomez, and ZJ Fanchon. An efficient delta-causal algorithm for real-time distributed systems. *Journal of Applied Sciences*, 9(9):1711–1718, 2009.

[21] S E Pomares Hernandez, Jean Fanchon, and Khalil Drira. The immediate dependency relation: an optimal way to ensure causal group communication. *Annual Review of Scalable Computing*, 3:61–79, 2004.

[22] Saul E Pomares Hernandez, Luis A Morales Rosales, Jorge Estudillo Ramirez, and Gustavo Rodriguez Gomez. Logical mapping: An intermedia synchronization model for multimedia distributed systems. *Journal of Multimedia*, 3(5):33–41, 2008.

[23] Zixia Huang, Klara Nahrstedt, and Ralf Steinmetz. Evolution of temporal multimedia synchronization principles: A historical viewpoint. *ACM*, 9(1s):34, 2013.

[24] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *ACM*, 21(7):558–565, 1978.

[25] Mario Montagud, Fernando Boronat, Hans Stokking, and Ray van Brandenburg. Inter-destination multimedia synchronization: schemes, use cases and standardization. *Multimedia systems*, 18(6):459–482, 2012.

[26] Abdelhamid Nafaa, Toufik Ahmed, Y Hadjadj, and Ahmed Mehaoua. Rtp4mux: a novel mpeg-4 rtp payload for multicast video communications over wireless ip. *IEEE*, 2003.

[27] Colin Perkins. *RTP: Audio and Video for the Internet*. Addison-Wesley Professional, 2003.

[28] Jorge Estudillo Ramirez and Saul E Pomares Hernandez. A temporal synchronization model for heterogeneous data in distributed systems. *International Journal of Information Technology*, 4(4), 2008.

[29] DCG Sinalgo. *Simulator for network algorithms (sinalgo)*, accessed on 2016-12-19. Available online: http://www.disco.ethz.ch/projects/sinalgo.

[30] Máté Tömösközi, Patrick Seeling, Péter Ekler, and Frank HP Fitzek. Performance evaluation of network header compression schemes for udp, rtp and tcp. *Periodica Polytechnica Electrical Engineering and Computer Science*, 60(3):151–162, 2016.

[31] UMTS World. *UMTS Network Planning Basics*, accessed on 2017-07-04. Available online: http://www.umtsworld.com/technology/capacity.htm.

[32] Dapeng Wu, Yiwei Thomas Hou, Wenwu Zhu, Ya-Qin Zhang, and Jon M Peha. Streaming video over the internet: approaches and directions. *IEEE Transactions on circuits and systems for video technology*, 11(3):282–300, 2001.

[33] SeungJune Yi, SungDuck Chun, YoungDae Lee, SungJun Park, and SungHoon Jung. *Radio Protocols for LTE and LTE-advanced*. John Wiley & Sons, 2012.