Towards a calculus for distributed, real-time and mobile systems

Toufik Messaoud Maarouk
University Center of Khenchela, Algeria
MISC Laboratory, Mentouri University of Constantine, Algeria
Email : toumaarouk@yahoo.fr

Djamel Eddine Saidouni
MISC Laboratory, Mentouri University of Constantine, Algeria
Email : saidounid@yahoo.fr

Mohamed Khergag
Ferhat Abbes University, Setif, Algeria
Email : m_khergag@esi.dz

Abstract— **This paper introduces a model for modeling real-time and mobile systems, which extends the DD-LOTOS language by the mobility nature of processes. Our model allows processes to move between distributed sites or localities, *i.e.* mobility of processes. Two types of communication are present in our model, local communication, *i.e.* the exchange of information between two processes in the same location, and remote communication, *i.e.* the exchange of information between two different localities, the latter is ensured by message exchange.**
We propose a syntactic and structured operational semantics based on true-concurrency semantics, expressing parallel behaviors and supporting at the same time temporal constraints, explicit actions durations, structural and temporal non-atomicity of actions and urgency. We also propose a semantic model for automatic verification tools, this model expresses aspects of the language DD-LOTOS, and adds mobility of process.

Index Terms— **True concurrency semantics, Mobility, Real time systems, Distributed systems, LOTOS.**

I. Introduction

The design, specification and analysis of real time and mobile systems reveal a great challenge today. This design and analysis requires methods and tools for formal specification. Process algebras are languages designed with the aim to specify this type of system, and many models have been proposed, particularly CCS[13][14], $\pi - calculus$[16], despite all the properties that can be studied in CCS, this calculus requires a fixed communication structure , on the other hand π-calculus resolves this difficulty by crossing a big step for creating new channels and to communicate channel names on a channel. LOTOS[7], is a formal description technique promoted to the rank of standard ISO, it relies on language CCS and CSP[6].

Despite the formal framework of the theoretical model, these languages suffers from some insufficiencies and ignore one or several characteristics of distributed, mobile and real-time systems. Several formalisms appeared to argue about such systems, in particular the models which take into account the time aspect in the model such as TCCS(Timed CCS)[17], RT-LOTOS[4], ET-LOTOS[11] and D-LOTOS[23], other models support the mobility and distribution of computation as Mobile ambients[9] based on the notion of locality, the Distributed π-calculus[5], distributed Join-calculus[3] is an asynchronous calculus with mobility and distribution. In [8], it was proposed bigraphs allowing unifying the mobile ambients and the π-calculus.

The main objective of these models is the modeling of aspects: mobility, distribution and real time. In $\pi - calculus$ mobility is expressed by the links move, in the virtual space of linked processes[15]. This mobility allows the dynamic reconfiguration of the topology of the interconnections between processes. The notion of locality is often associated that of migration allowing the mobility of entities between the various domains of a system. A distributed system can be described as a collection of computational activities spread among different sites or localities, which can be physical or logical. eg Dπ-calculus offers construct expressing migration and communication purely local, DJoin-calculus also provides primitives for explicit localities, migration, remote communication, and join-patterns for local communication.

The other important aspect is the time, the models CCS, LOTOS, $\pi - calculus$, and Join-calculus does not allow to specify system whose behavior depends strictly on time. Most of the protocols contain timing mechanisms essential for the safety of functioning. Most of the works concern the extension of existing models and in particularly the process algebras: TCCS, RT-LOTOS, ET-LOTOS, D-LOTOS, $\pi RT - calculus$[10].

In previous works, a language for modeling distributed real-time systems named DD-LOTOS[12] was proposed, extends the language D-LOTOS to support distribution, which incorporating both temporal constraints and durations of actions. DD-LOTOS has true concurrency semantics, in which we escape the assumption of actions structural and temporal atomicity, the idea is based on maximality-based semantics[22].

In this paper, we present a model for modeling distributed, mobile and real time systems. The main advantage of our approach is the explicit support of the following characteristics:

- Concurrent and parallel behaviors : our model is defined using a true concurrency semantics(maximality-based semantics),
- Distribution of calculus : to take into account the distributed aspect of distributed systems, localities are defined explicitly in the model,
- Mobility : using a primitive explicit processes can migrate from an locality to another,
- Real-time : the aim of introducing time in the model, it is to allow the description of the quantitative aspect of the moments which events are really occurs.

The paper is organized as follows : in section 2 we present maximality semantics and the syntax and operational semantic of DD-LOTOS, in section 3 we present the syntax and semantic of our language(Mobile DD-LOTOS). In section 4 we present mobile semantic model for mobile real time systems based on communicating automata which provides a formal basis to reasoning about the desired properties of such systems. Section 5 gives an example of generation of model Mobile C-DATA from a Mobile DD-LOTOS specification. Related work is exposed in section 6, finally in section 7 we conclude.

II. Distributed calculus

A. Maximality semantics

In this section, we introduce maximality semantics of BASIC LOTOS[7], as defined in [23].

1) Principle of semantics of maximality

Semantics of a concurrent system can be characterized by all the states of the system and the transitions through which the system passes from a state to another. In the approach based on maximality, the transitions are events which represent only the beginning of the execution of the actions.

To illustrate the principle of maximality let us consider the expressions of the behavior E and F as follow:

$E = a; stop|||b; stop$ and $F = a; b; stop[]b; a; stop$

In the initial state, no action was yet executed, thus the set of the maximal events is empty, hence from the following initial configurations associated with E and F: ${}_{\phi}[E]$ and ${}_{\phi}[F]$. By applying semantics of maximality, the following transitions are possible:

$${}_{\phi}[E] \xrightarrow{\phi^{a x}}{}_{m\{x\}}\ [stop]\ |||\ {}_{\phi}[b; stop] \xrightarrow{\phi^{b x}}{}_{m\{x\}}\ [stop]\ |||_{\{y\}}[stop]$$

x (resp y) being the name of the event identifying the beginning of the action 'a' (respectively ' b '). Given that nothing can be concluded about the ending of both actions 'a' and 'b ' in the configuration ${}_{\{x\}}[stop]|||_{\{y\}}[stop]$, x and y are there then maximal in this configuration. It should be noted x is also maximal in the intermediate state represented by the configuration ${}_{\{x\}}[stop]|||_{\phi}[b; stop]$. For the initial configuration, associated with the expression of behavior F, the following transition is possible :

$${}_{\phi}[F] \xrightarrow{\phi^{a x}}{}_{m\{x\}}\ [b; stop]$$

As previously, x identifies the beginning of the action 'a' and it is the name of the only maximal event in the configuration ${}_{\{x\}}[b; stop]$. It is clear that, in view of semantics of the operator of prefixing, the beginning of the execution of the action ' b ' is possible only if the action 'a' ended its execution. Consequently, x is no longer maximal when the action ' b ' begins its execution; the unique maximal event in the resultant is thus the one identified by y which corresponds to the beginning of the execution of the action ' b '. All the names of the maximal events were thus modified by the deletion of x and the addition of y, which justify the following derivation:

$${}_{\{x\}}[b; stop] \xrightarrow{\{x\}^{a y}}{}_{m\{y\}}\ [stop]$$

The configuration ${}_{\{y\}}[stop]$ is different from the configuration ${}_{\{x\}}[stop]\ |||\ {}_{\{y\}}[stop]$, because the first one possesses only a single maximal event (identified by y), while the second possesses two (identified by x and y).

B. Introduction of durations and of the temporal constraints

Let $\mathcal{D}$ be a countable set, the elements of $\mathcal{D}$ indicate temporal values. Let $\Im$ be the set of all duration functions $\tau : Act \rightarrow \mathcal{D}$ such as $\tau(i) = \tau(\delta) = 0$. τ_0 is the constant function defined by $\tau_0(a) = 0$ for all $a \in Act$. The duration function τ being fixed, let us consider the behavior expression of $G = a; b; stop$. In the initial state, no action is complying, thus the associated configuration is ${}_{\phi}[a; b; stop]$; starting from this state, the transition ${}_{\phi}[a; b; stop] \xrightarrow{\phi^{a x}}{}_{\{x\}}\ [b; stop]$ is possible. The resulting state interprets the fact that the action a is potentially in execution. According to the maximality semantics, we cannot know if the action a has terminated its execution, except if the action b starts its execution (the beginning of b depends on the end of a); thus, if b starts its execution, we can deduce that a has been terminated. We can thus note that the durations of actions are present in an intrinsic but implicit way in the maximality approach; their explicit consideration will enable us to reason on the quantitative properties of system behaviors.

By taking into account the duration of the action a, we can accept the transition: ${}_{\phi}[a; b; stop] \xrightarrow{\phi^{a x}}{}_{\{x:a:\tau(a)\}}$ $[b; stop]$. The obtained configuration shows that the action b can begin its execution only if a duration equal to $\tau(a)$ has passed, this value represents the necessary time for the execution of action a. Of course, we can consider the intermediate states representing the flowing of a lapse of time $t \leq \tau(a)$ by ${}_{\{x:a:\tau(a)\}}[b; stop] \xrightarrow{t}{}_{\{x:a:\tau(a)-t\}}$ $[b; stop]$; such configurations will be called there after temporal configurations, which leads us to note that a configuration generated by the maximality semantics represents in fact a class of temporal configurations. The explicit consideration of the durations of action in the process algebras alone does not allow to specify real-time systems. Compensate for this insufficiency, the classic operators of delay similar those introduced into temporal extensions of LOTOS are used, such as ET-LOTOS or RT-LOTOS, semantics of these operators being naturally expressed in the context of maximality.

Because the actions are not atomic, the temporal constraints concern, in this context, the beginning of execution of the actions and not the complete execution of the actions.

C. Distributed D-LOTOS language

The DD-LOTOS[12] language represents an extension of D-LOTOS language to support the distribution and communication between the localities, it was enriched with the following features:

- The explicit distribution and remote communication.

Distribution is ensured by introducing the notion of locality. Localities exchange information by the principle message exchange. The syntax of DD-LOTOS is defined as follows:

$$
\begin{array}{ll}
E ::= & \textbf{Behaviors} \\
 & stop \mid exit\{d\} \mid \Delta^d E \mid X[L] \mid \\
 & g@t[SP]; E \mid i@t\{d\}; E \mid hide\, L\, in\, E \mid \\
 & E[]E \mid E|[L]|E \mid \quad E \gg E \mid E\,[> E \\
 & \mid a!v\{d\}; E \\
 & \mid a?xE \\
S ::= & \textbf{Systems} \\
 & \phi \mid \quad S \mid S \;\mid\; l(E)
\end{array}
$$

Fig. 1: *Syntax of DD-LOTOS*

Let PN, ranged over by $X, Y...$, be an infinite set of process identifiers, and let $\mathcal{G}$, ranged over by g, the set of gates (observable actions). $i \notin \mathcal{G}$ is the internal action and $\delta \notin \mathcal{G}$ is the successful termination action. $Act = \mathcal{G} \cup \{i, \delta\}$, ranged over by α, is the set of actions. L denotes any finite subset of $\mathcal{G}$. The terms of DD-LOTOS are named behavior expressions, $\mathcal{B}$ ranged over by $E, F, ...$ denotes the set of behavior expressions.

Let $\mathcal{D}$ be a domain of time. $\tau : Act \rightarrow \mathcal{D}$ is the duration function which associates to each action its duration. We assume $\tau(i) = \tau(\delta) = 0$. Let g be an action, E a behavior expression and $d \in \mathcal{D}$ a value in the temporal domain.

The main syntax concerns the syntax of systems S and behavior expression E. The informal semantics of syntactic items is the following:

- Informally $a\{d\}$ means that action a has to begin its execution in a temporal interval $[0, d]$. $\Delta^d E$ means that no evolution of E is allowed before the end of a delay equal to d. In $g@t[SP]$; E (resp. $i@t\{d\}; E$) t is a temporal variable recording the time taken after the sensitization of the action g (resp. i) and which will be substituted by zero when this action ends its execution.
- The basic operators of process algebras as : nondeterministic choice $E[]E$, parallel composition $E|[L]|E$, the interiorization $hide\, L\, in\, E$, sequential composition $E \gg E$, and preemption $E\,[> E$.
- The expression $a!v\{d\}; E$, specifies the emission message v via the communication channel a. This emission operation must occur in the temporal interval $[0, d]$.
- On the other side, the behavior expression $a?xE$ specifies the message receiving on channel a. The received message substitutes the variable value x. This variable is used in the behavior expression E.
- A system may be either :
  - Empty, expressed by ϕ,
  - The composition of sub systems $S \mid S$, or
  - A behavior expression E in a locality l expressed by $l(E)$.

Definition 1: The set of event names is a denumerable set noted $\mathcal{M}$. This set is ranged over $x, y, ...$ $M, N, ...$ denotes a finite subsets of $\mathcal{M}$. The set of atoms of support Act is $Atm = 2^{\mathcal{M}}_{fn} \times Act \times \mathcal{M}$, $2^{\mathcal{M}}_{fn}$ is the power set of $\mathcal{M}$. For $M \in 2^{\mathcal{M}}_{fn}$, $x \in \mathcal{M}$ and $a \in Act$, the atom (M, a, x) be noted ${}_M a_x$. The choice of an event name can be deterministic by using any function $get : 2^{\mathcal{M}} - \{\phi\} \rightarrow \mathcal{M}$ satisfying $get(M) \in M$ for all $M \in 2^{\mathcal{M}} - \{\phi\}$.

Definition 2: The set $\mathcal{C}_t$ of temporal configurations is defined by :

- $\forall E \in \mathcal{B}, \forall M \in 2^{\mathcal{M} \times Act \times \mathcal{D}}_{fn}$: $\quad {}_M[E] \in \mathcal{C}_t$
- $\forall P \in PN, \forall M \in 2^{\mathcal{M} \times Act \times \mathcal{D}}_{fn}$: $\quad {}_M[P] \in \mathcal{C}_t$
- if $\mathcal{E} \in \mathcal{C}_t$ then hide L in $\mathcal{E} \in \mathcal{C}_t$
- if $\mathcal{E} \in \mathcal{C}_t$ et $F \in \mathcal{B}$ then $\mathcal{E} \gg F \in \mathcal{C}_t$
- if $\mathcal{E}, \mathcal{F} \in \mathcal{C}_t$ then $\mathcal{E}$ op $\mathcal{F} \in \mathcal{C}_t$ $\quad op \in \{\ [] \ ,\ ||| \ ,\ || \ ,\ |[L]| \ ,\ [> \ \}$
- if $\mathcal{E} \in \mathcal{C}_t$ et $\{a_1, ..., a_n\}$, $\{b_1, ..., b_n\} \in 2^{\mathcal{G}}_{fn}$ then $\mathcal{E}\,[b_1/a_1, ..., b_n/a_n] \in \mathcal{C}_t$
- $\forall \mathcal{E} \in \mathcal{C}_t, \forall d \in \mathcal{D} : \Delta^d \mathcal{E} \in \mathcal{C}_t$
- ${}_{\{x:g:d\}}[E(t)] \in \mathcal{C}_t$

Definition 3: (actions) The actions in global system are :

- The set of communication actions between localities : are emission or receiving messages through a communication channel $Act_{com} ::= a!m \mid a?x \mid \tau$ (output actions, input actions and the silent action).
- The set $Act = \mathcal{G} \cup \{i, \delta\}$ previously defined.

Definition 4: (Localities and channels) : The set $\mathcal{L}$ ranged over by l, denotes the set of localities. ϑ an infinite set of channels defined by users ranged over by *a*,*b*,...channels are used for communication message between localities.

D. Structured operational semantics :

The temporal maximality transition relation between the temporal configurations is noted $\rightarrow \subseteq \mathcal{C}_t \times Atm \cup \mathcal{D} \times \mathcal{C}_t$. This semantic is given as follows :

Process $a!v\{d\}; E$: Let us consider the configuration ${}_M[a!v\{d\}; E]$, the emission of the message v begins once the actions indexed by the set M, have finished their execution, conditioned by the condition $Wait(M)$ which must be equal to $false$ in rule (1). The predicate $Wait : 2_{fn}^{\mathcal{M}\times Act\times\mathcal{D}} \longrightarrow \{true, false\}$ defined on every $M \in 2_{fn}^{\mathcal{M}\times Act\times\mathcal{D}}$ as follows: $Wait(M) = \exists x : a : d \in M$ such that $d > 0$. Intuitively, $Wait(M) = true$ if there is at least an action referred in M which is yet in execution. Rules (2) and (3) express the fact that the time attached to the process of emission cannot begin to elapse only if all the actions referenced by M are finished. Rule (4) imposes that the occurrence of the action of emission takes place for the period d, otherwise the process is transformed to $Stop$.

$$(1)\quad \frac{\neg Wait(M)}{{}_{M}[a!v\{d\};E] \overset{M^{a!v}x}{\longrightarrow} {}_{\{x:a!v:t\}}[E]} \qquad x = get(\mathcal{M})$$

$$(2)\quad \frac{Wait(M^{d'})\ or\ (\neg Wait(M^{d'})\ and\ \forall\varepsilon{>}0.\ Wait(M^{d'-\varepsilon}))\ d'{>}0}{{}_{M}[a!v\{d\};E] \overset{d'}{\longrightarrow} {}_{M^{d'}}[a!v\{d\};E]}$$

$$(3)\quad \frac{\neg Wait(M)}{{}_{M}[a!v\{d'{+}d\};E] \overset{d}{\longrightarrow} {}_{M}[a!v\{d'\};E]}$$

$$(4)\quad \frac{\neg Wait(M)\ and\ d'{>}d}{{}_{M}[a!v\{d\};E] \overset{d'}{\longrightarrow} {}_{M}[stop]}$$

Process $a?xE$: Let us consider the configuration ${}_{M}[a?xE]$, the following rule expresses that the receiving starts once the action indexed by the set M have finished its execution.

$$\frac{\neg Wait(M)}{{}_{M}[a?xE] \overset{M^{a?x}y}{\longrightarrow} {}_{\{y:a?x:0\}}[E]}$$

Remote communication

Distributed activities exchange messages between them, the expression $l(a!v\{d\})$, expresses that the message v is offered for a duration d. By an activity at the locality l, the message v should be sent on channel a. On the other side, $k(a?xE)$ specifies that activity E in locality k is ready to receive a message on channel a. The following rule defines the remote communication between two distributed activities via the channel a. In this case communication will be specified by silent (τ) evolution as follows :

$$\frac{-}{{}_{M}[l(a!v\{d\};E1)] \mid {}_{M'}[k(a?xE2)] \overset{\tau}{\rightarrow} {}_{M}[l(E1)] \mid {}_{M'}[k(E2\{v/x\})]}$$

Time evolution on system

$$\frac{E \overset{d}{\longrightarrow} E'}{l(E) \overset{d}{\longrightarrow} l(E')}$$

$$\frac{S_1 \overset{d}{\longrightarrow} S_1' \qquad S_2 \overset{d}{\longrightarrow} S_2'}{S_1 \mid S_2 \overset{d}{\longrightarrow} S_1' \mid \ S_2'}$$

III. THE MOBILE DD-LOTOS LANGUAGE

In this section we propose a calculus for distributed, real time computing with mobility. In DD-LOTOS, distribution is ensured by the presence of localities in the calculus.

The mobile and distributed applications are naturally dynamic, which means that the number of processes involved in the system is not fixed. All any time there can be a new creation of processes, or a deletion of another process. To be able to study this type of applications, it is necessary to use dynamic models, among other models, which formally supply tools for this characteristic.

Our calculus introduces the dynamics by allowing the processes to migrate from a locality to an other one. Although we require a new primitive to enable migration between localities. Thus we augment DD-LOTOS with the construct : $go(l, E)\{d\}$, intuitively this means : migrate the behavior E to location l, this migration is offered to the environment for a period d. Another dynamic aspect of our calculus is the possibility of creating of new locality, or deletion of localities.

Let us illustrate our approach with an example represented by the following system:

$$l(F \ ||| \ (creat(k, E) >> go(k, E')\{d\}))$$

It is constituted of a single locality l. This locality contains two parallel processes, the first one F, and the second is composed by the sequential composition operator (>>), which is composed of the process of creating the locality k ($creat(k, E)$) and followed by the migration process ($go(k, E')\{d\}$).

After one calculus step, we obtain the system :

$$l(F \ \ ||| \ go(k, E')\{d\}) \mid k(E)$$

This step consists of creating the locality k, so the system decomposes now of two localities l and k. We assume that the migration will be activated in the time interval $[0, d]$, we obtain the following system :

$$l(F) \mid k(E'|||E)$$

If the process F completes its behavior, so it becomes the process $stop$, then we obtain :

$$l(stop) \mid k(E'|||E)$$

In our calculus a locatity which contains that the process $stop$ will be deleted, we obtain the following system :

$$k(E'|||E)$$

A. Syntax

In this section we introduce our calculus of mobile processes. We extend the DD-LOTOS to model mobile systems, processes can be distributed into several localities. The proposed model represents both local computation on different localities and remote communication between localities. We introduce structures to account for creation of localities and migration of behaviors.

Definition 5: (actions) The actions in global system are of the following :
- The set $Act = \mathcal{G} \cup \{i, \delta, go, create\}$: specific processes within a locality
- And the set of communication actions between localities : are emission or receiving messages through a communication channel $Act_{com} ::= a!m \mid a?x \mid \tau$ (output actions, input actions and the silent action).

The syntax of our model is an extension of that of DD-LOTOS, by introducing aspects of mobility.

The syntax of calculus is given in figure 2. The main syntactic category is that of a system S and behaviors E. Intuitively, a system consists of a set of localities running independently in parallel, which localities can communicate via channels. The system can be empty or contains behaviors for execution .

If the expression $go(k, E)$ is currently residing at locality l it can migrate to the locality k and then continue the execution of E. The set $\mathcal{B}$ ranged over by $E, F, ...$ denotes all the expressions of behaviors.

$E ::=$		**Behaviors**
	...	
	$\mid go(l, E)\{d\}$	**Migration**
	$\mid create(l, E)$	**Creation of locality**
$S ::=$		**Systems**
	$\phi \mid \; S \mid S \mid \; l(E)$	

Fig. 2: *Syntax of Mobile DD-LOTOS*

The informal meaning of the various builders so defined is the following :

- A process $go(l, E)\{d\}$ provokes the migration of the behavior E to location l, this migration is offered to the environment for a period d. The migration must be realized in the space of the time 0 to d, if the time taken exceeds d, the action is no available to the environment. In a system without temporal constraints, this process behaves as follows: The locality which contains the migration, migrates to the locality described in the primitive.

- A process $create(l, E)$ provokes the creation of the locality l and start the execution of the behavior E in this locality.

Example

Let be the following system:

$$l(go(k, E)\{d\} \;|||\; F) \;\; \mid k(E')$$

The system is composed of two localities. The locality l contains two parallel processes : the action of migration and the process F. The locality k that contains the process E'.

In this system we have several cases:

Case 1 :

The migration can take place before the time taken does not exceed d units, which gives :

$$l(F) \;\; \mid k(E' \;|||\; E)$$

Case 2 :

There is a duration d', such as $d = d' + d''$, and the time taken after the action of migration is made sensitive is d”, which gives :

$$l(go(k, E)\{d'\} \;|||\; F) \;\; \mid k(E')$$

Case 3 :

The time taken exceeds the duration d, thus the migration will never occur, which gives :

$$l(F) \;\; \mid k(E')$$

In that case we have to make the action of migration take place within the period d, otherwise the migration will never occur.

B. *Structured operational semantics of calculus :*

In[12] we studied two aspects of distributed systems: distribution and communication. In distribution we introduced the notion of locality, for communication we have adopted two types: local communication in the same locality on the gates of synchronization and remote communication by sending / receiving messages communication channels. The semantics of basic operators (stop, exit, preemption, delay operator, action with predicate, Interiorization, nondeterministic choice, parallel composition, sequential composition) remains the same as in D-LOTOS.

Definition 6: The relation of transition of maximality $\rightarrow \subseteq \mathcal{C} \times Atm \cup D \times \mathcal{C}$ is defined as the smallest relation satisfying the following rules:

- Process of creation of localities : $l(create(k, E))$

Let us consider the configuration $_M[l(F \mid create(k, E))]$, this configuration represents potential evolutions according to the actions indexed by the set M. The creation of localities can not occur until the actions indexed by the set M have completed their execution, that is $wait(M) = false$ in rule (1). If the action of creation is not sensitized and we have d units of time passed, then it is expressed by rule (2).

$$(1) \quad \frac{\neg wait(M)}{_M[l(F \mid create(k,E))] \overset{M^{create_x}}{\longrightarrow} \; _\phi[l(F)] \mid_{\{x:create:0\}} [k(E)]}$$

$$(2) \quad \frac{wait(M^d) \qquad d>0}{_M[l(F \mid create(k,E))] \xrightarrow{d} \; _{M^d}[l(F \mid create(k,E))]}$$

- Process of deletion of localities **:**

If the set of processes in current execution in a locality is $stop$, we delete this locality.

$$(1) \quad \frac{\neg wait(M) \qquad and \; [l(stop)]}{_M[l(stop)] \xrightarrow{\delta} \; _\phi[\phi]}$$

$$(2) \quad \frac{\neg wait(M) \qquad and \qquad _M[l(stop)] \;|||\; _N[k(E)]}{_M[l(stop)] \;|||\; _N[k(E)] \xrightarrow{\delta} \; _N[k(E)]}$$

- Process of migration : $_M[k(go(l, E)\{d\})]$

We consider the configuration $k(F \mid go(l, E)\{d\})$, the process of migration can not occur until the actions indexed by the set M have completed their execution, which is expressed by rule (1). Rule (2) requires that the occurrence of the action go has for the period d, in the contrary case the migration will never. Rule (3), expresses the passage of time.

(1) $\dfrac{\neg wait(M)}{{}_M[k(F \mid go(l,E)\{d\})] \overset{M^{go_x}}{\rightarrow} {}_{\phi}[k(F)] \mid_{\{x:go:0\}} [l(E)]} x = get(\mathcal{M})$

(2) $\dfrac{\neg wait(M) \ and \ d' > d}{{}_M[k(go(l,E)\{d\})] \overset{d'}{\rightarrow} {}_M[k(Stop)]}$

(3) $\dfrac{\neg wait(M) \ and \ d' > 0}{{}_M[k(go(l,E)\{d+d'\})] \overset{d}{\rightarrow} {}_{M^d}[k(go(l,E)\{d'\})]}$

IV. Behavioral model for mobile, real time systems

In[12] we discuss a further extension of DATA[1], who take in account the remote communication with the intuition that a distribution of calculus is explicit (each automaton represents a locality), so this model is still static and he does not allow dynamic evolution of calculus, to represent the notions recently defined for our calculus, we have to extend our model C-DATA[12], to allow the constructs defined for the purposes: Migration, Creation of localities and also Deletion theme.

Our model distinguishes between two types of calculus, local and global, for the local calculation we can synchronize the communications between the processes, description can be done with the C-DATA.

We propose an extension of C-DATA for the support and to express the mobility of processes, creation and destruction of localities expresses the dynamic aspect of the calculus.

A. Formalization

Definition 7: A Mobile Communicating DATA (Mobile C-DATA) $A(S, L_S, s_0, \vartheta, H, \Pi, T_D)$ represents a subsystem with :

- S is a finite set of states,
- $L_S : S \to 2^{\Phi_t(H)}_{fn}$ is a function which corresponds to each state s the set F of ending conditions(duration conditions) of actions possibly in execution in s,
- $s_0 \in S$ is the initial state,
- ϑ is the alphabet of the channels on which messages flow between the subsystems.
- H is a finite set of clocks,
- $\Pi = Act_{com} \cup Act$, is the set of actions of A, and
- $T_D \subseteq S \times 2^{\Phi_t(H)}_{fn} \times 2^{\Phi_t(H)}_{fn} \times \Pi \times H \times S$ is the set of transitions.
  A transition $(s, G, D, \alpha/(a(!/?)v)/\tau, z, s\prime)$ represents switch from state s to state $s\prime$, by starting execution of action $\alpha \in Act$ (creating, deletion or migration of a locality), actions of communication(emission or Receiving) or synchronization for the accomplishment of communication (silent action) and updating clock z.
  G is the corresponding guard which must be satisfied to fire this transition.
  D is the corresponding deadline which requires, at the moment of its satisfaction, that action α must occur.
  $(s, G, D, \alpha/(a(!/?)v)/\tau, z, s\prime)$ can be written $s \xrightarrow{G,D,\alpha/(a(!/?)v)/\tau),z} s'$.

Definition 8: **(System)** A system of n $Mobile$ $C\text{-}DATA$ is a tuple $S = (A_1, \dots, A_n)$, with $A_i = (S_i, L_{S_i}, s_{0_i}, \vartheta, H_i, \Pi_i, T_{iD})$ a $Mobile\ C-DATA$.

Definition 9: 1) **States** : $GS(S) = (s_1, v_1) \times \dots \times (s_n, v_n) \times (\vartheta^*)^p$, is the set of states.
2) **Initial state** The initial state of S is :$q_0 = ((s_{01}, 0), \dots, (s_{0n}, 0) : \epsilon_1, \dots, \epsilon_p)$ such as ϵ is the empty word on the alphabet ϑ
3) **System states** Let $S = (A_1, \dots, A_n)$ a system of n $Mobile$ $C\text{-}DATA$, $A_i = (S_i, L_{S_i}, s_{0_i}, \vartheta, H_i, \Pi_i, T_{iD})$:
A global state of S is defined by the state of each subsystem and the states of each channel, a state of S is an element of
$(s_1, v_1) \times \dots \times (s_n, v_n) \times (\vartheta^*)^p$ such that $v_i(h)$ are valuations on H.

Definition 10: Let $S = (A_1, \dots A_n)$ be a system of n $Mobile$ $C\text{-}DATA$, $A_i = (S_i, L_{S_i}, s_{0_i}, \vartheta, H_i, \Pi_i, T_{iD})$. The semantics of a system S is given by the following rules. A transition T between the state $s = ((q_1, v_1), \dots, (q_n, v_n) : x_1, \dots, x_p)$ and the state $s\prime = ((q\prime_1, v\prime_1), \dots, (q\prime_n, v\prime_n) : x\prime_1, \dots, x\prime_p)$ is a rule of emission (RE) or reception (RR) or executing a silent action (RA) or passage of time (RP) or creation rule (RC) or migration rule (RM) or deletion rule (RD) or synchronization rule (RS):

(1) (RE rule of emission)
$$\frac{\big((q_i,v_i),a!v,\ \big(q'_i,v'_i\big)\big)\in T_D}{(\dots,(q_i,v_i),\dots:\dots,x_g,\dots)\overset{a!v}{\rightarrow}\big(\dots,\big(q'_i,v'_i\big),\dots:\dots,x_g.a,\dots,\big)}$$

(2) (RR rule of reception)
$$\frac{\big((q_i,v_i),\ a?x,\ \big(q'_i,v'_i\big)\big) \in T_D}{(\dots,(q_i,v_i),\dots:\dots,x_g.a,\dots)\overset{a?x}{\rightarrow}\big(\dots,\big(q'_i,v'_i\big),\dots:\dots,x_g,\dots\big)}$$

(3) (RA rule of executing an internal action)
$$\frac{\big((q_i,v_i),\ a,\ G,\ D,\ z,\ \big(q'_i,v'_i\big)\big) \in T_D \qquad v\models G}{(\dots,(q_i,v_i),\dots:\dots,x_i,\dots)\overset{\alpha}{\rightarrow}\big(\dots,\big(q'_i,v'_i\big),\dots:\dots,x_i,\dots\big)}$$

(4) (RP passage of time)
$$\frac{d\in R^+ \qquad \forall d'\leq d \qquad (v_i+d)\ \not\models D}{(\dots,(q_i,v_i),\dots:x_1,\dots,x_p)\overset{d}{\rightarrow}(\dots,(q_i,v_i+d),\dots:x_1,\dots,x_p)}$$

(5) (RC creation rule)
$$\frac{\big(Q(\dots,(q_n,v_n)),\ create(l,E),\ Q'(\dots,(q_{n+1},v_{n+1}))\big)\in T_D}{(\dots,(q_n,v_n):x_1,\dots,x_p)\overset{create}{\rightarrow}(\dots,(q_{n+1},v_{n+1}):x_1,\dots,x_{p+1})}$$

(6) (RM migration rule)

$$\frac{\big((q_i,v_i),\ go(l,E),\ (q'_i,v'_i)\big)\in T_D\ \ v_i\models G}{(\ldots,(q_i,v_i),\ldots:x_1,\ldots,x_p)\xrightarrow{go}(\ldots,\big(q'_i,v'_i\big),\ldots:x_1,\ldots,x_p)}$$

(7) (RD deletion rule)

$$\frac{\big(Q(\ldots,(q_n,v_n)),\ \delta,\ Q'(\ldots,(q_{n-1},v_{n-1}))\big)\in T_D}{(\ldots,(q_n,v_n):x_1,\ldots,x_p)\xrightarrow{\delta}(\ldots,(q_{n-1},v_{n-1}):x_1,\ldots,x_{p-1})}$$

(8) (RS for this rule the channels x_i and x_j do not contain any messages, because if we synchronize the emission with the reception, we must consume the message from channel)

$$\frac{\big((q_i,v_i),\ a?x,\ \big(q'_i,v'_i\big)\big)\in T_D\quad \big((q_j,v_j),\ a!v,\ \big(q'_j,v'_j\big)\big)\in T_D\ i\neq j}{(\ldots,(q_i,v_i),\ldots,(q_j,v_j),\ldots:x_1,\ldots,x_p)\xrightarrow{\tau}\big(\ldots,\big(q'_i,v'_i\big),\ldots,\big(q'_j,v'_j\big),\ldots:x_1,\ldots,x_p\big)}$$

where G is a temporal constraint or $guard$, D is the corresponding deadline which requires, at the moment of its satisfaction, that action a must occur, z the clock is to be reset.

V. EXAMPLES

A. Exemple 1

Let be the process E that receives information(Rq_Data) on the channel a, then it waits for a period of t units of time. The process offers the message m on channel b for a period d units of time, and finishes its execution with the process $exit$.

The behavior expression could be specified as

$$E ::= \quad a?Rq\_Data \\ (\Delta^t(b!m\{d\}; exit))$$

In the initial state no action is complying, which explains why the duration conditions set is empty. We correspond in this state the expression E to form the initial mobile C-DATA configuration $_{\phi}[E]$ where no action was drawn. From this configuration, the first action to be taken by the behavior E is the receipt of information Rq_Data on channel a. A clock x, chosen by the function get form the set $\mathcal{M}$ and having the initial value 0, is associated to this action. We apply the reception rule(RR) of model mobile C-DATA :

$$\underbrace{_{\phi}[E]}_{config0}\ \xrightarrow{\phi,\, a?Rq\_Data,\, x}\ \underbrace{_{\{x\geq\tau_1\}}[\Delta^t((b!m\{d\}\ ;\ exit))]}_{config1}$$

τ_1 : represents the time of reception the message Rq_Data on channel a. When the reception is completed, from the configuration $config1$ the behavior expression $\Delta^t((b!m\{d\}\ ;\ exit))$ cannot start its execution only if t units of time elapsed.

$$config1\ \xrightarrow{t}\ \underbrace{_{\phi}[(b!m\{d\}\ \ ;\ exit)]}_{config2}$$

From this configuration, the emission of the message m on channel b, is possible

$$config2\ \xrightarrow{\phi,\, b!m,\, x}\ \underbrace{_{\{x\geq\tau_2\}}[\ exit]}_{config3}$$

τ_2 : represents the time of emission the message m on channel b. From the semantics of the operator ";", the process $exit$ cannot begin its execution only if the message m was sent.

We can have a different situation from the $config2$, if the time elapsed since the sensitization of the action of emission exceeds the deadline of offer d units of time, thus the process is transformed into the process $stop$.

$$\frac{t>d}{config2\xrightarrow{t}\underbrace{_{\phi}[\ stop]}_{config4}}$$

From the configuration $config3$, we obtain :

$$config3\ \ \xrightarrow{\phi,\,\delta,\,x}\ \ \underbrace{_{\{x\geq 0\}}[stop]}_{config5}$$

B. Exemple 2

In this section, we give how to generate mobile C-DATA from a mobile DD-LOTOS specification. Let us take the example given in [12] of the behavior of simplified communications protocol. In this protocol we have assumed that the system is composed of one sender and two receivers, this system may be defined by :

$$l\ (E)|k\ (P)\ |\ n(Q)$$

where :

- l is the locality of the sender, and E its behavior,
- k is the locality of the receiver R1, and P its behavior,
- n is the locality of the receiver R2, and Q its behavior.

We suppose that the source communicates with the receiver $R1$ via the channel a, and with the receiver $R2$ via the channel b.

The behavior of the sender is as follows:

1) Messages are sent each t units of time. Each message is sent to the receivers via the channels a and b.
2) When the sender receives a negative acknowledgement from a receiver, then resend the lost message to it.
3) Creation of new localities (new receivers)

The behavior of a receiver is as follows:

1) When receiving a message v, concatenate it with received messages,
2) When detecting a lost of a message, then send a negative acknowledgement to the sender.
3) Process migration

In locality l, the activity E is composed of three sub activities, which are specified respectively by the behavior expressions $(a!v\{d\}\ |\ b!v\{d\}), c?xE'$ and $create(m,F)$. The expression $(a!v\{d\}\ |\ b!v\{d\})$ describes the fact that the sender may send the message v on channels a and b, with the constraint that this message offering is during d units of time.

In the expression $c?xE'$ (c may be a or b), identifies the loste message. c is the channel on which negative acknowledgement is received. E' specifies the resending operation.

And the expression $create(m, F)$, describes the creation of localities.

Thus the behavior expression E may be defined by:

$$E ::= E1 \;\;|||\;\; E2 \;\;|||\; E3$$

$$E1 ::= (a!v\{d\}|b!v\{d\}) >> \;\; \Delta^t E1$$

$$E2 ::= (c?xE') >> E2$$

$$E3 ::= create(m, F)$$

P and Q are respectively the behaviors associated to localities k and n *i.e.* the receiver R1 and the receiver R2.

$$P ::= ((a?xB \;[]\; \Delta^t i; c!Nack\{d\}) >> go(l, G)\{d'\}) >> P$$

$$Q ::= ((b?xB \;[]\; \Delta^t i; c!Nack\{d\}) >> go(l, G)\{d'\}) >> Q$$

The expression $go(l, G)\{d'\}$ describes the migration of the behavior G towards the sender. B specifies the concatenation operation of the received messages.

The complete specification in mobile DD-LOTOS language is given by figure 3:

Specification $Sender - Receivers[a, b, c]$
behavior
$l\ (E) \mid k\ (P) \mid n(Q).$

Where

process $E[a, b, c, m] ::= E1 \;\;|||\;\; E2 \;\;|||\; E3$
Where
$E1 ::= (a!v\{d\}|b!v\{d\}) >> \;\; \Delta^t E1$
$E2 ::= (c?xE') >> E2$
$E3 ::= create(m, F) >> E3$
Endproc

process $P[a, c] ::=$
$((a?xB \;[]\; \Delta^t i; c!Nack\{d\}) >> go(l, G)\{d'\}) >> P$
Endproc

process $Q[b, c] ::=$
$((b?xB \;[]\; \Delta^t i; c!Nack\{d\}) >> go(l, G)\{d'\}) >> Q$
Endproc

Endspec.

Fig. 3: *Specification of Sender and receivers*

The system consists of three subsystems, ie three different localities. The communication between the activities of different localities is ensured by the model C-DATA.

In locality l, the behavior expression of the source is a parallel composition of three expression, and is translated by:

${}_{\phi}[E] \rightarrow_{\phi} [E1] \;\;|||\;\; {}_{\phi}[E2] \;\;|||\;\; {}_{\phi}[E3]$

In the initial state no action is complying, which explains why the duration conditions set is empty. We correspond in this state the expression E to form the initial mobile C-DATA configuration ${}_{\phi}[E]$ where no action was drawn. From this configuration, the emission of the message v on channels a and b is possible, also the creation of locality m chosen by the function get form the set $\mathcal{L}$. A clock x, chosen by the function get form the set $\mathcal{M}$ and having the initial value 0, is associated to this emission. We apply the emission rule of model mobile C-DATA :

$$\underbrace{{}_{\phi}[E1] \;|||\; {}_{\phi}[E2] \;|||\; {}_{\phi}[E3]}_{config0}\xrightarrow{\phi, a!v, x}$$

$$\underbrace{{}_{\{x\geq\tau_1\}}[b!v\{d\}) >> \;\Delta^t E1] \;|||\; {}_{\phi}[E2] \;|||\; {}_{\phi}[E3]}_{config1}$$

$$\xrightarrow{\{x \geq \tau_1\}, b!v, y}$$

$$\underbrace{{}_{\{x\geq\tau_1, y\geq\tau_2\}}[\;\Delta^t E1] \;|||\; {}_{\phi}[E2] \;|||\; {}_{\phi}[E3]}_{config2}$$

With τ_1(resp τ_2) represents the time of emission the message v on channel a (resp b). We apply the creation rule of model mobile C-DATA :

$${}_{config2}\xrightarrow{\{x \geq \tau_1, y \geq \tau_2\}, create, z}$$

$$\underbrace{{}_{\{x\geq\tau_1, y\geq\tau_2\}}[\;\Delta^t E1] \;|||\; {}_{\phi}[E2] \;|||\; {}_{\{z\geq\tau_3\}}[E3]}_{config3}$$

With τ_3 represents the time of creation of locality m.

The same reasoning is applied in the following way to the other branch where the emission on channel b begins before the emission on channel a :

$$\underbrace{{}_{\phi}[E1] \;|||\; {}_{\phi}[E2] \;|||\; {}_{\phi}[E3]}\xrightarrow{\phi, b!v, y}$$

$$\underbrace{{}_{\{y\geq\tau_2\}}[a!v\{d\} >> \;\Delta^t E1] \;|||\; {}_{\phi}[E2] \;|||\; {}_{\phi}[E3]}_{config1}$$

$$\xrightarrow{\{y \geq \tau_2\}, a!v, x}$$

$$\underbrace{{}_{\{x\geq\tau_1, y\geq\tau_2\}}[\;\Delta^t E1] \;|||\; {}_{\phi}[E2] \;|||\; {}_{\phi}[E3]}_{config2}$$

$$\xrightarrow{\{x \geq \tau_1, y \geq \tau_2\}, create, z}$$

$$\underbrace{{}_{\{x\geq\tau_1, y\geq\tau_2\}}[\;\Delta^t E1] \;|||\; {}_{\phi}[E2] \;|||\; {}_{\{z\geq\tau_3\}}[E3]}_{config3}$$

We obtain the same configuration($config3$) if the expression of creation that begins the first.

From the configuration $config2$ the behavior expression $[\;\Delta^t E1]$ can not start its execution only if both send on channels a and b finished, in other words, that if the conditions on the durations are quite satisfied, corresponding to the condition $\{x \geq \tau_1 \wedge y \geq \tau_2\}$. The following trasition becomes possible :

$$\underbrace{{}_{\{x\geq\tau_1,y\geq\tau_2\}}[\ \Delta^t E1]\ |||\ {}_{\phi}[E2]\ |||\ {}_{\phi}[E3]}_{config2}$$

$$\xrightarrow{d}\underbrace{{}_{\phi}[\ \Delta^{t-d}E1]\ |||\ {}_{\phi}[E2]\ \ |||\ {}_{\phi}[E3]}_{config3}$$

In the locality $k(n)$, behavior expression of the receiver $R1(R2)$, is a choice between either receiving a message on channel $a(b)$, or if the reception is not performed after t units of time, a negative acknowledgement message sent to the source and this is translated in mobile C-DATA :

$$\underbrace{{}_{\phi}[P]}_{config0}\ \xrightarrow{\phi, a?v, x}$$

$$\underbrace{({}_{\{x\geq\tau_4\}}[B[e/v]>>\ {}_{\phi}[\ go(l,G)\{d'\}])>>{}_{\phi}[P]}_{config1}$$

With τ_4 represents the time of reception the message on channel a.

In the other hand, when the time of receiving(t units of time) is exceeded, the number of lost message($Nack$) is transmitted

$$\frac{d>t}{{}_{\phi}[P]\xrightarrow{d}\underbrace{({}_{\phi}[i;c!Nack\{d\}]>>\ {}_{\phi}[\ go(l,G)\{d'\}])>>{}_{\phi}[P]}_{config2}}$$

Internal action i is urgent and $\tau(i) = 0$.

$$config2\xrightarrow{\phi, i, x}$$

$$\underbrace{({}_{\{x=0\}}[c!Nack\{d\}]>>\ {}_{\phi}[\ go(l,G)\{d'\}])>>{}_{\phi}[P]}_{config3}$$

$$\xrightarrow{\phi, c!Nack, x}\underbrace{{}_{\{x\geq\tau_5\}}[\ \ go(l,G)\{d'\}]>>{}_{\phi}[P]}_{config4}$$

With τ_5 represents the time of reception of the message on channel c.

The information $Nack$ is offered to the environment for a duration of d units of time, if the time (t) elapsed since the sensitization of the action of emissions exceed d units then the action is not offered.

$$\frac{t>d}{config3\xrightarrow{t}\underbrace{{}_{\phi}[\ go(l,G)\{d'\}]>>{}_{\phi}[P]}_{config5}}$$

From the configuration $config4$ the behavior expression [$go(l,G)\{d'\}$] can not start its execution only if emission on channel c finished. In other words, that if the condition on the duration is quite satisfied, corresponding to the condition $\{x \geq \tau_5\}$. So the configuration $config4$ becomes the configuration $config5$.The following trasition becomes possible :

$$config5\xrightarrow{\phi, go, x}\underbrace{{}_{\{x\geq\tau_6\}}[\ go(l,G)\{d'\}]\ >>\ {}_{\phi}[P]}_{config6}$$

With τ_6 represents the time of migration.

The migration go is offered to the environment for a duration of d units of time, if the time (t) elapsed since the sensitization of the action go exceed d units then the action is not any more offered.

$$\frac{t>d}{config5\xrightarrow{t}\underbrace{{}_{\phi}[P]}_{config7}}$$

The same reasoning is applied in the locality n.

VI. Related Works

For modeling concurrent processes, the π-caluclus[16] was introduced by R. Milner. However, it considers a notion of mobility different from our calculus, is translated by the existence of names, which is inseparable from the process of communication, fundamentally in any concurrent system. The existence of names suggests also a space abstracted from connected processes, in which names represent the connections, only the processes which share names are then capable of interacting. The structure of system thus changes in a dynamic way, because the links between processes are ceaselessly created and destroyed.

$\pi_{1l} - calculus$[19][20], is an asynchronous $\pi - calculus$, extended with a notion of locality and related model of failures. Domains in the $\pi_{1l}-calculus$ manifest themselves in two ways, first the model features an explicit notion of site, written $\{P\}a$; representing the process P running at the locality named a. Second, for each locality name a, there is an associated process $Loc(a)$. The topology of sites in $\pi_{1l} - calculus$ the is flat. In the mobility aspect, the migrant entities are the processes, expressed by the instruction $spawan(a, P)$ send the process P to the locality a to run there, it is an instruction asynchronous objective.

$D\pi - calculus$[5], was designed to be a minor extension of the $\pi - calculus$ by which elementary semantic notion of distribution could be studied. A syntactic category of locations or sites is introduced and all processes now exist, and execute at a specific named location. There is a new mechanism for processes to move from one site to another. The basic entities in $D\pi$ are processes and systems. A system is a set of processes located in parallel. The topology of domains is flat. There is no remote communication: a message for which the corresponding receiver is in a distant domain must migrate towards its destination. In $D\pi - calculus$ migration is expressed by the instruction *goto l.P*, like in $\pi_{1l} - calculus$, processes are migrant entities, and it's an asynchronous and objective instruction.

$Djoin - calculus$[3][2], is a model for mobile programming that includes explicit features for co-localization of receivers and partial failure. The basic entities of $Djoin - calculus$ are messages, definitions, def-processes, pattern of joint message and solutions. Domains are given by localities in the DJoin. Localities constitute a tree structure, and are units of migration and failure. The distant communication is done in two steps : first output current domain and input in the target domain, then consummation of message. In $Djoin-calculus$ only localities can migrate, the instruction of migration has the form $go(a, k)$ where a is the locality of destination and k is a continuation. It is an asynchronous and subjective migration.

Inspired by the π-caluclus, the Ambients Calculus[9] considers processes that are executed in hierarchically nested environment called ambients and that may transferred from an ambient to another. An ambient is a place that is delimited by boundary and where multiple processes execute. Each ambient has a name, a collection of local processes, and a collection of sub-ambients. Ambients can move in and out of other ambients, subject to capabilities that are associated with ambient names. Communication and interaction between different domains can only be obtained through ambient migration and opening. The model describes the migration of process in domains (ambients). An ambient can migrate in any location in the tree. The calculus offers three form of mobility ($in\ n.P$, $out\ n.P$ and $open\ n.P$), the migration is subjective.

M-calculus[24], higher-order extension of the distributed Join-calculus with programmable localities. From the Join-calculus, the M-calculus retains the idea of asynchronous communication, definitions with join patterns of messages for synchronization and of hierarchically organized localities. Basic entities for distribution are localities. The topology of localities are hierarchically. The migration is realized by means of higher-order communication.

KLAIM[21], Extension of the language LINDA. The communication mechanism is asynchronous and is based on the concept of tuple space. KLAIM extends language LINDA with notions of sites (physical locations), localities (logical location) and migration agents. The remote communication is a two time: the tuple to be transmitted is locally assessed and placed in the tuple space target, then it is consumed. Mobile entities are processes, they are static. There are two types of migration (two instructions), both being objective. The first $out(P)@l.Q$ migrates a process P to the locality with the context (mobile context), on the contrary, $eval(P)@l.Q$, migrates P to l, where it is evaluated in the context of l.

PICT[18], is a language based on an asynchronous $\pi - calculus$. NomadicPICT[25], an extension of PICT with notions of locality, agent and migration. Basic entities are the processes and agents. These processes are located and named. The domain space is hierarchical, NomadicPICT offers two types of mobile entities: the migration of processes and the migration of messages. The instruction of migration is expressed by ($migrate\ to\ s \rightarrow P$), this migration is subjective.

VII. Conclusion

In this paper, we proposed a calculus modeling mobility in distributed systems, our model also allows to modeling the real-time systems.

In a previous work[12], we introduced the notion of locality required for modeling the distributed aspect of distributed systems. In the distributed model the communication is assured by exchange of messages. Then, a behavioral model was defined in terms of temporal labeled system of transitions. In this work we expanded the model for two new constructions: the construction of migration processes between distributed localities, and construction of creation of new localities.

The main interest of our approach is the proposal of a language defined on true concurrency semantics: semantics of maximality which allows the explicit expression of durations, and it supports temporal constraints including urgency of actions.

Concerning communication, we have defined local and remote communication, when two processes want to communicate, so are on the same locality, then the communication is ensured through the gates which are defined locally. If both processes are on two different localities then the message exchange is the way of communication.

References

[1] N. Belala and D. E. Saïdouni. Non-Atomicity in Timed Models. In *International Arab Conference on Information Technology (ACIT'2005)*, Al-Isra Private University, Jordan, December 2005.

[2] C.Fournet. *A calculus for Distributed Mobile Programming*. PhD thesis, Ecole Polytechnique, Palaiseau France, 1998.

[3] C.Fournet, G.Gonthier, J.J.Lévy, L.Maranget, and D.Rémy. A calculus of mobile agents. *Montanari and Sassone*, 103:406–421, 1996.

[4] J. Courtiat and R. de Oliveira. On RT-LOTOS and its application to the formal design of multimedia protocols. *Annals of Telecommunications*, 50:11–12, 1995.

[5] M. Hennessy and J. Riely. Resource access control in systems of mobile agents. *Information and Computation*, 173:82–120, 2002.

[6] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[7] ISO8807. *LOTOS, A Formal Description Technique Based on the Ordering of Observational Behaviour*. ISO, November 1988.

[8] O. H. Jensen and R. Milner. Bigraphs and mobile processes(revised). Tech. report, University of Cambridge, Computer Laboratory, 2004.

[9] L.Cardelli and A.D.Gordon. Mobile ambients. *Lecture Notes in Computer Science*, 1378:140–155, 1998.

[10] J. Y. Lee and J. Zic. On modeling Real-time Mobile Processes. In *Twenty-Fifth Australasian Computer Science Conference ACS 2002*, volume 4, Melbourne, Australia, 2002.

[11] L. Léonard and G. Leduc. An Introduction to ET-LOTOS for the Description of Time-Sensitive Systems. *Computer Networks and ISDN Systems*, 29:271–292, 1997.

[12] T. M. Maarouk, D. E. Saïdouni, and M. Khergag. DD-LOTOS : A distributed real time language. In *Proceedings 2nd Annual International Conference on Advances in Distributed and Parallel Computing (ADPC 2011) Special Track: Real Time Embedded Systems (RTES 2011)*, pages 45–50, Singapore, 2011.

[13] R. Milner. Calculus for synchrony and asynchrony. *TCS*, 25:267–310, 1983.

[14] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[15] R. Milner. *Communicating and mobile systems : the pi-calculus*. Cambridge University Press, May 1999.

[16] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes,Parts I and II. *Information and Compution*, pages 100–140,41–77, September 1992.

[17] F. Moller and C. Tofts. A temporal calculus of communicating systems. In J. C. Baeten and J. W. Klop, editors, *CONCUR*, volume 458 of *LNCS*, pages 401–415. Springer-Verlag, 1990.

[18] B. Pierce. Programming in the Pi-Calculus : A Tutorial Introduction to Pict (Pict version 4.1). Technical report, Indiana University, 1998.

[19] R.Amadio. An asynchronous model of locality, failure, and process mobility. In *Proceedings COORDINATIO 97*, volume 1282 of *LNCS*, 1997.

[20] R.Amadio. On modelling mobility. *Theoretical Computer Science*, 240:147–176, 2000.

[21] R.De Nicola, G.Ferrari, and R.Pugliese. KLAIM : A Kernel Lanaguage for Agents Interaction and Mobility. *IEEE Transactions on Software Engineering*, 24(5):315–330, 1998.

[22] D. E. Saïdouni. *Sémantique de Maximalité: Application au Raffinement d'Actions en LOTOS*. PhD thesis, LAAS-CNRS, 7 av. du Colonel Roche, 31077 Toulouse Cedex France, 1996.

[23] D. E. Saïdouni and J. P. Courtiat. Prise en compte des durées d'action dans les algèbres de processus par l'utilisation de la sémantique de maximalité. In *Ingénierie Des Protocoles (CFIP'2003)*. Hermes, France, 2003.

[24] A. Schmitt and J.-B. Stefani. The M-Calculus: A Higher Order Distributed Process Calculus. In *Proceeding 30th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2003)*, New Orleans, LA, USA, Jan. 2003.

[25] A. Unyapoth. *Nomadic Pi Calculi : Expressing and Verifying Infrastructure for Mobile Computation*. PhD thesis, University of Cambridge, Computer Laboratory, 2002.