Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 203 (2008) 5–18
www.elsevier.com/locate/entcs

Interaction in Time and Space

Gabriel Ciobanu[1]

Faculty of Computer Science
"A.I.Cuza" University
Iaşi, Romania

Abstract

Distributed π-calculus and ambient calculus are extended with timers which may trigger timeout recovery processes. Timers provide a useful notion of relative time with respect to the interaction in a distributed system. The rather flat notion of space in timed distributed π-calculus is improved by considering a hierarchical representation of space in timed mobile ambients. Some basic results are proven, making sound both formal approaches. An easily understood example is used for both extensions, showing how it is possible to describe a non-monotonic behaviour and use a decentralized control to coordinate the interacting components in time and space.

Keywords: timed distributed π-calculus, timed mobile ambients, interaction

1 Introduction

We refer to systems with a large number of interacting processes located at different sites which can migrate from one location to another. In order to manage the large nondeterminism of interaction, we add new quantitative ingredients restricting the interaction between processes.

Concurrent processes may overlap in time during their computation, and thus time represents an important aspect of concurrency. Distribution occurs when the system components are separated in space, and thus space represents an important aspect of distribution. We present two formalisms (called timed distributed π-calculus and timed mobile ambients, respectively) where explicit notions of location, timers, capacity, and other ingredients are used to control interaction and mobility in distributed systems. Explicit timeout of interactions and other quantitative elements can define a non-monotonic behaviour; for instance, when a timer expires, it may trigger a timeout recovery processes. Several challenges related to these systems include the specification of global behaviour and how local descriptions lead to such a global behaviour, mechanisms and techniques of coordination

[1] Email: gabriel@info.uaic.ro

doi:10.1016/j.entcs.2008.04.083

based on the restriction of the nondeterminism, a decentralized control taking into account the relative time of interaction and other quantitative aspects. When we work with a large number of processes, the global behaviour could represent more than the sum of parts behaviour, certain emergent behaviour being possible.

We work with local clocks, considering the relative time of interaction given by timers. The decreasing of a timer measures the relative time, and this procedure is uniform (it is similar for each local clock). The timers define the available resources; the resources are accessed by interaction according to the interval of time given by timers. Using such a time control, we can restrict the nondeterminism of the evolution by selecting only the processes ready to interact according to their timers and satisfying certain quantitative requirements.

Throughout this paper we consider the following example which can motivate and illustrate our approach. We assume the situation of a student finishing his lectures, moving out the university building and looking for an available vehicle in order to move to a given place. In front of the university there are a tram stop, a bus stop, and taxicabs. The student has the possibility to use any of the three types of vehicles: bus, tram and cab to reach the target location. Since the three variants (vehicles) have different costs, the student can establish a priority among them: the bus has the highest priority, followed by the tram, and finally the cab. The bus and the tram are moving according to a predetermined schedule which is evolving according to their own clocks. Our scenario involves a bus, a tram and two cabs (a hired cab and an available cab).

Interaction in time is associated with the ability to specify a starting and an ending point providing the availability of a resource. Interaction in space relates to the ability to identify a specific location in which a required and available resource exists. We use extensions with timers of both timed distributed π-calculus and timed mobile ambients.

2 Timed Distributed π-calculus

Distributed π-calculus ($D\pi$) as an extension with types and locations of the π-calculus [7]. $D\pi$ is presented in [6], and provides a theoretical framework for describing communications between distributed processes with restricted resource access. In [4] we extend the distributed π-calculus by introducing timers over channel names in order to define timeouts for communications. The resulting formalism is called timed distributed π-calculus ($tD\pi$). Over this formalism it is possible to define a coordination of the whole system by assigning specific values to timers and defining a set of time constraints [5].

In $tD\pi$, waiting for a communication on a channel is no longer indefinite (like in $D\pi$); if no communication happens in a predefined interval of time, the waiting process goes to another state. This approach leads to a method of sharing the channels in time. The timer Δt of each channel makes the channel available for communication only for the period of time determined by the discrete value t. To simplify our presentation we choose a simpler π-calculus and omit the syntax for

matching or *summation*. A communication channel is considered a fixed resource at a certain location.

The syntax of *Input* and *Output* communication uses a pair of processes. For instance, an *Input* expression $a^{\Delta t}?(X:T).(P,Q)$ evolves to P whenever a communication is established during the interval of time given by Δt; otherwise it evolves to Q. The variable X is considered bound only in P and we should provide its type T; the type system is presented in [4].

Table 1: *Syntax of timed distributed π-calculus*

$u ::= x$	Variable Name	$P, Q ::= stop$	Termination
$\mid a^{\Delta t}$	Timed Channel	$\mid P \mid Q$	Composition
$l ::= x$	Variable Name	$\mid (\nu\, u\colon A)P$	Channel Restriction
$\mid k$	Location Name	$\mid go\ l.(P,Q)$	Movement
$v ::= bv$	Base Value	$\mid u!\langle v\rangle.(P,Q)$	Output
$\mid u \mid l$	Name	$\mid u?(X:T).(P,Q)$	Input
$\mid u@l$	Located Name	$\mid {}^*P$	Replication
$\mid (v_1,..,v_n)$	Tuple of Values	$M, N ::= M \mid N$	Composition
$X ::= x$	Variable	$\mid (\nu\, u@l : T)N$	Located Restriction
$\mid X@l$	Located Variable	$\mid l[[P]]_\Gamma$	Located Process
$\mid (X_1,..,X_n)$	Tuple of Variables		

Two channels are equal $a_1^{\Delta t_1} = a_2^{\Delta t_2}$ if and only if $a_1 = a_2$ and $t_1 = t_2$. Waiting indefinitely on a channel a is allowed by considering Δt as ∞. For example, an output process defined by the expression $a^{\infty}!\langle v\rangle.(P,Q)$ awaits forever to send the value v, simulating the behaviour of an output process in untimed π-calculus. In the expression below, two processes are running in parallel and can interact along the common channel a:

$$a^{\Delta \mathbf{t}}!\langle v\rangle.(P,Q) \mid a^{\Delta \mathbf{t}'}?(X:T).(P',Q') \longrightarrow P \mid P'\{{}^{v}/_{X}\}$$

We define a type environment Γ as a set of *location types*. The purpose of the type environment associated with a specific process is to restrict the range of resources the process can access. Formally, $\Gamma \subseteq \mathcal{L} \times K$ is a relation associating to a location name a location type. A location type is a set of *location capabilities* which may contain *channel types*, *move* capability (i.e., permission to migrate to that location), or *channel creation* capability (i.e., permission to create channels).

We extend the channel types of $D\pi$ with timers of the form Δt. Communication is now permitted on channels only in the interval of time given by the timer value t (i.e. until the timer of the channel type expires). These timers define the existence of the channel types inside the type environment. Timers decrease with each "tick" of their local clock, and this decreasing process is the same for each clock. Upon expiration, the channel types are discarded. Timers are created at once with the channel types, and are activated when the types are added to the type environment. When the processes receive new channel names, types for the new channels become available. It means that the processes can communicate on the new channels according to the new types. For example, if a process receives through an input channel a located name $a@k$, then it gains the capability to move to location k, and to communicate on channel a.

We define a function ψ which affects only the set of capabilities. It decreases

the timers of the channel types and removes the types with an expired timer. By removing channel types, it is possible to get location types with only *go* capability (we call them *empty locations*). A process can move to an empty location, but there it does not have the capability to perform any action, and consequently produces a *runtime error*. Thus ψ removes also the empty locations.

The passage of time is formalized by a *time-stepping function* ϕ_Δ defined over the set $\mathcal{P}_\Delta$ of tagged located processes. The possible communications are performed at each tick of the local clock, and the active channels are those that could be involved in these communications. ϕ_Δ affects the active channels which do not communicate (the channels involved in communication disappear together with their timers). Due to timers, the capabilities can be lost, which leads to "errors". We define ϕ_Δ to check the existence of the needed types and change the process accordingly.

Definition 2.1 *(Tagged time-stepping function)*
We define $\phi_\Delta : \mathcal{P}_\Delta \to \mathcal{P}_\Delta$, where Γ' is obtained by application of the cleanup function ψ. Note that we use a concise notation $a^{\Delta t}.(R,Q)$ to stand for both $a^{\Delta t}!\langle v\rangle.(R,Q)$ and $a^{\Delta t}?(X:T).(R,Q)$.

$$\phi_\Delta(l[[P]]_\Gamma) = \begin{cases} k[[R]]_{\Gamma'} & \text{if } P = go\,k.(R,Q) \text{ and } \Gamma(k) <: loc\{go\} \\ l[[Q]]_{\Gamma'} & \text{if } P = go\,k.(R,Q) \text{ and } k \notin dom(\Gamma) \\ l[[a^{\Delta(t-1)}.(R,Q)]]_{\Gamma'} & \text{if } P = a^{\Delta t}.(R,Q),\ t>1 \text{ and } t \neq \infty \\ l[[Q]]_{\Gamma'} & \text{if } P = a^{\Delta t}.(R,Q),\ t \leq 1 \\ l[[Q]]_{\Gamma'} & \text{if } P = a^{\Delta t}.(R,Q),\ t>1 \text{ and } \Gamma \not<: \Gamma(l,a) \\ \phi_\Delta(l[[R]]_\Gamma) \mid \phi_\Delta(l[[Q]]_\Gamma) & \text{if } P = R \mid Q \\ (\nu a@l:A)\phi_\Delta(l[[R]]_{\Gamma\{a@l:A\}}) & \text{if } P = (\nu a:A)R \\ l[[P]]_{\Gamma'} & \text{otherwise} \end{cases}$$

As ϕ_Δ decreases the channel timers, we extend it to take care of the type environments (by applying the cleanup function ψ). In the definition of ϕ_Δ we omit the channel type and the transmitted message in the input and output processes for brevity. For the *go k* syntax if the location type contains the capability *go*, then R is executed; if k is not defined in Γ, then Q is executed. If *go* is not present, the process is considered to do something against its permissions and an error is generated.

Well-typedness of processes is defined by a set of static rules (a detailed presentation of the static typing rules is given in [4]). These rules express the behaviour of a process with regard to its types. If a process tries to communicate on a channel for which it has no capability, it can still be well-typed if the alternative process Q is well-typed. Q is called the *safety process*.

We write $\Gamma \vdash P$ and say that *process P is well-typed with respect to type environment* Γ; we also write $\Gamma \vdash_k P$ and say that *P is well-typed to run at location k*. To say that $P = a^{\Delta t}!\langle v\rangle.(R,Q)$ is well-typed to run at location k, with respect to type environment Γ, the following statements should hold: *(i)* $\Gamma \vdash_k v : T$ which means that v is a well-formed value at location k of type T; *(ii)* $\Gamma \vdash_k a : res\{w\langle T\rangle\}\Delta t$ which means that channel a exists at location k and may communicate values of type T for another t units of time; *(iii)* $\Gamma \vdash_k R$; $\Gamma \vdash_k Q$ which means that R and Q are well-typed at location k. For a tagged located process $k[[P]]_\Delta$, the well-typedness relation is denoted by $\Vdash$, and it is defined by using the well-typedness relation $\vdash_k$

for a process P running at location k.

Since the function ψ changes the capability set Γ by removing channel and location types, we are interested if the process is still well-typed under the new Γ'. The following lemma relates the typing environment of the processes with the passage of time.

Lemma 2.2 *(Well-typedness is preserved by the cleanup function)*
If $\Gamma \Vdash l[[P]]_\Delta$ *then* $\Gamma \Vdash \psi(l[[P]]_\Delta)$.

We consider the tagged located processes ranged over by N and M (e.g., N represents $l[[P]]_\Gamma$). We denote by $\not\rightarrow$ the fact that rules (R_Γ-COM1) and (R_Γ-COM2) cannot be applied. Using these notations, we give the following reduction rules providing a dynamic semantics for $tD\pi$.

$$(\mathrm{R}_\Gamma\text{-IDLE})\ \frac{l[[P]]_\Gamma \not\rightarrow}{l[[P]]_\Gamma \rightarrow \phi_\Delta(l[[P]]_\Gamma)}$$

$$(\mathrm{R}_\Gamma\text{-COM1})\ \frac{\Gamma(l,a) <: res\{r\langle T\rangle\}}{l[[a^{\Delta t}!\langle v\rangle.(P,Q)]]_\Delta \mid l[[a^{\Delta t'}?(X:T).(P',Q')]]_\Gamma \rightarrow \psi(l[[P]]_\Delta) \mid \psi(l[[P'\{^v/_X\}]]_{\Gamma \sqcap \{v@l:T\}})}$$

$$(\mathrm{R}_\Gamma\text{-COM2})\ \frac{\Gamma(l,a) <: res\{ro\langle T\rangle\}}{l[[a^{\Delta t}!\langle v\rangle.(P,Q)]]_\Delta \mid l[[a^{\Delta t'}?(X:T).(P',Q')]]_\Gamma \rightarrow \psi(l[[P]]_\Delta) \mid \psi(l[[P'\{^v/_X\}]]_\Gamma)}$$

$$(\mathrm{R}_\Gamma\text{-PAR})\ \frac{N \rightarrow N' \quad M \rightarrow M'}{N \mid M \rightarrow N' \mid M'} \qquad (\mathrm{R}_\Gamma\text{-RES})\ \frac{N \rightarrow N'}{(\nu a@l:T)N \rightarrow (\nu a@l:T)N'}$$

$$(\mathrm{R}_\Gamma\text{-CONG})\ \frac{N \equiv N' \quad N \rightarrow M \quad M \equiv M'}{N' \rightarrow M'}$$

Several results are presented in [4]; here we mention two of them claiming that the passage of time does not interfere with the typing system, and that once well-typed, a process remains well-typed (subject reduction).

Proposition 2.3 *If* $\Gamma \Vdash l[[P]]_\Delta$ *then* $\Gamma \Vdash \phi_\Delta(l[[P]]_\Delta)$.

Theorem 2.4 *(Subject Reduction)*
For all tagged located processes

(a) If $N \equiv N'$ *then* $\Gamma \Vdash N$ *if and only if* $\Gamma \Vdash N'$.

(b) If $N \rightarrow N'$ *then* $\Gamma \Vdash N$ *if and only if* $\Gamma \Vdash N'$.

We describe the evolution of a system in timed distributed π-calculus by considering the example in which a student intends to move from university to another location. This migration involves a bus, a tram and two cabs. The scenario could be described as follows:

$$in_{univ} = bus_{univ}^{\Delta t_1}!\langle stud_{univ}\rangle.bus_{univ}^{\Delta t_2}?(i).in_{univ}$$

$$in_{camp} = bus_{camp}^{\Delta t_3}?(i).bus_{camp}^{\Delta t_4}!\langle stud_{camp}\rangle.in_{camp}$$

$$bus = bus_{univ}^{\Delta t_5}?(i).go\ camp.bus_{camp}^{\Delta t_6}!\langle stud_1\rangle.bus_{camp}^{\Delta t_7}?(i).go\ univ.$$

$$bus_{univ}^{\Delta t_8}!\langle stud_2\rangle.bus$$

$$system = camp[[in_{camp}]] \mid univ[[in_{univ} \mid bus]]$$

where

$stud_1$ - the number of students that get out of bus in $camp$
$stud_2$ - the number of students that get out of bus in $univ$
$stud_{univ}$ - the number of students that get into bus in $univ$
$stud_{camp}$ - the number of students that get into bus in $camp$

Using bold letters to emphasize the interacting elements, a possible evolution of the system is given by

$$system = camp[[in_{camp}]] \mid univ[[in_{univ} \mid bus]] =$$

$$= camp[[in_{camp}]] \mid univ[[\mathbf{bus_{univ}^{\Delta t_1}}!\langle\mathbf{stud_{univ}}\rangle.bus_{univ}^{\Delta t_2}?(i).in_{univ} \mid \mathbf{bus_{univ}^{\Delta t_5}}?(\mathbf{i}).$$

$$go\ camp.bus_{camp}^{\Delta t_6}!\langle stud_1\rangle.bus_{camp}^{\Delta t_7}?(i).go\ univ.bus_{univ}^{\Delta t_8}!\langle stud_2\rangle.bus]]$$

$$\rightarrow camp[[bus_{camp}^{\Delta t_3}?(i).bus_{camp}^{\Delta t_4}!\langle stud_{camp}\rangle.in_{camp}]] \mid univ[[bus_{univ}^{\Delta t_2}?(i).in_{univ}|$$

$$\mathbf{go\ camp}.bus_{camp}^{\Delta t_6}!\langle stud_1\rangle.bus_{camp}^{\Delta t_7}?(i).go\ univ.bus_{univ}^{\Delta t_8}!\langle stud_2\rangle.bus]]$$

$$\rightarrow camp[[\mathbf{bus_{camp}^{\Delta t_3}}?(\mathbf{i}).bus_{camp}^{\Delta t_4}!\langle stud_{camp}\rangle.in_{camp} \mid \mathbf{bus_{camp}^{\Delta t_6}}!\langle\mathbf{stud_1}\rangle.$$

$$bus_{camp}^{\Delta t_7}?(i).go\ univ.bus_{univ}^{\Delta t_8}!\langle stud_2\rangle.bus]] \mid univ[[bus_{univ}^{\Delta t_2}?(i).in_{univ}]]$$

$$\rightarrow camp[[\mathbf{bus_{camp}^{\Delta t_4}}!\langle\mathbf{stud_{camp}}\rangle.in_{camp} \mid$$

$$\mathbf{bus_{camp}^{\Delta t_7}}?(\mathbf{i}).go\ univ.bus_{univ}^{\Delta t_8}!\langle stud_2\rangle.bus]] \mid univ[[bus_{univ}^{\Delta t_2}?(i).in_{univ}]]$$

$$\rightarrow camp[[in_{camp}|\ \mathbf{go\ univ}.bus_{univ}^{\Delta t_8}!\langle stud_2\rangle.bus]] \mid univ[[bus_{univ}^{\Delta t_2}?(i).in_{univ}]]$$

$$\rightarrow camp[[in_{camp}]] \mid univ[[\mathbf{bus_{univ}^{\Delta t_2}}?(\mathbf{i}).in_{univ} \mid \mathbf{bus_{univ}^{\Delta t_8}}!\langle\mathbf{stud_2}\rangle.bus]]$$

$$\rightarrow camp[[in_{camp}]] \mid univ[[in_{univ} \mid bus]] \ = \ system$$

A problem which appears in this description is the fact that the information exchanged in the interaction of the system components are lost. There is no way to remember the number of persons inside the bus, or the number of the persons who entered or exited the bus at each stop.

We can remark that the notion of space (location) used in $tD\pi$ is flat. A more realistic account of physical distribution is obtained using a hierarchical representation of space (locality) as in the ambient calculus [3].

3 Mobile Ambients with Time Constraints

Ambient calculus [3] is a formalism for describing distributed and mobile computation in terms of ambients. An ambient is both a named location and the unit of movement. The mobility of an ambient is controlled by the capabilities *in, out*,

and *open*. In an ambient we have processes which may exchange messages. In [2] we introduce timed Mobile Ambients (tMA) where communication actions, capabilities and ambients are used as temporal resources. We add timers to ambients, capabilities, input and output actions. A timer Δt of each temporal resource makes the resource available only for a determined period of time t. Here we ignore the types, emphasizing how the interaction between ambients depends on timers and space proximity. A related approach is presented in [1].

We imagine large populations of interacting processes defining an interaction field. This means that we embed the viewpoint of interaction as a pair (action, reaction) into an interaction field where emergence properties and non-compositionality could play important roles. Since we do not know how to describe/specify this kind of interaction field, we use a notion of proximity expressing the fact that the interaction between pairs of processes depend on a dynamically changing topology given by the whole interaction field. A possible example could be the flocking behaviour given by the formation of flocks by birds and their evolution. In order to emphasize the spatial nature of interaction in the framework of an interaction fields, we denote by $p(n, r)$ the area of (inter)action of an ambient n; it depends on the radius of (inter)action r it has at a certain time.

We denote by $n^{\Delta t}_{(l,h,r)}[P]$ the fact that an ambient n has assigned a timer Δt, a *capacity* l representing the number of its free resources, a *weight* h representing the number of resources allocated in the parent ambient, and a radius r of its spatial proximity. If $t > 0$ the ambient behaves exactly as in untimed ambient calculus. Since the timer Δt can expire ($t = 0$), we use a pair $(n^{\Delta t}_{(l,h,r)}[P], k \lhd Q)$ to denote a timed ambient, where Q is a *safety process*. The prefix $k\lhd$ represents the number of resources needed by process Q to be executed. If a process $k \lhd Q$ does not have at least k free resources, Q cannot be executed and awaits until the required resources are available. If nothing happens in t units of time, the ambient n is dissolved, process P running inside the ambient is reduced to $\mathbf{0}$, and process $k \lhd Q \mid \Diamond_h$ is executed. h represents the total weight of the dissolved ambient; by $\Diamond_h$ we denote that h resources become available in the parent ambient of the dissolved ambient n. If $Q = \mathbf{0}$ we can simply write $n^{\Delta t}_{(l,h,r)}[P]$ instead of $(n^{\Delta t}_{(l,h,r)}[P], k \lhd Q)$. The behaviour of an untimed mobile ambient is given by using ∞ instead of Δt, and 0 instead of h, k, l and r. We use a pair of processes for the input, output and movement processes. The process $open^{\Delta t} n.(P, k \lhd Q)$ evolves to P whenever in time Δt the process becomes sibling to an ambient n; otherwise evolves to $k \lhd Q$. The process $!\langle m\rangle^{\Delta t}.(P, k \lhd Q)$ evolves to P whenever in time Δt the process becomes sibling to a process which is willing to capture the name m; otherwise evolves to $k \lhd Q$. The syntax of the timed Mobile Ambients is defined in the following table.

Table 2: *Syntax of timed mobile ambients*

n, m		names	$P, Q ::=$		processes
x, y		variables		$\mathbf{0}$	inactivity
M	$::=$	capabilities		$M^{\Delta t}.(P, k \lhd Q)$	movement
	$in\ n$	can enter n		$(n^{\Delta t}_{(l,h,r)}[P], k \lhd Q)$	ambient
	$out\ n$	can exit n		$P \mid Q$	composition
	$open\ n$	can open n		$(\nu n)P$	restriction
				$!\langle m\rangle^{\Delta t}.(P, k \lhd Q)$	output action
				$?(x)^{\Delta t}.(P, k \lhd Q)$	input action
				$*(k \lhd P)$	replication
				$P + Q$	choice

We define various technical ingredients, e.g, a function *weight* which counts the resources needed by a process to be executed. Timed Mobile Ambients use a discrete time. The passage of time is described by a (discrete) time-stepping function ϕ_Δ defined over the set $\mathcal{P}$ of tMA-processes in a similar way as in $tD\pi$. The local communication (given by the interaction between $!\langle m\rangle^{\Delta t}.(P, Q)$ and $?(x)^{\Delta t}.(R, k \lhd Q)$ inside the same ambient) does not consume time, so it is not included in the definition of ϕ_Δ. This function is used in the definition of the reduction rules.

The semantics of the timed mobile ambients is given by two relations: structural proximity relation and reduction relation. *Structural proximity relation* $P \equiv_p Q$ relates different syntactic representations of the same process; *reduction relation* $P \to Q$ describes the evolution of processes.

Processes are grouped into equivalence classes by $\equiv_p$. This relation provides a way of re-arranging expressions such that the interacting parts can be brought together. The structural relation $\equiv_p$ over the timed mobile processes is the least relation satisfying the following requirements:

Table 3: *Structural proximity congruence*

(S-Refl)	$P \equiv_p P$	**(S-Sym)**	$P \equiv_p Q$ implies $Q \equiv_p P$
(S-Trans)	$P \equiv_p R$, $R \equiv_p Q$ implies $P \equiv_p Q$		
(S-Res)	$P \equiv_p Q$ implies $(\nu n)P \equiv_p (\nu n)Q$		
(S-LPar)	$P \equiv_p Q$ implies $R \mid P \equiv_p R \mid Q$		
(S-RPar)	$P \equiv_p Q$ implies $P \mid R \equiv_p Q \mid R$		
(S-Repl)	$P \equiv_p Q$ implies $*(k \lhd P) \equiv_p *(k \lhd Q)$		
(S-Amb)	$P \equiv_p Q$ and $R \equiv_p R'$ implies $(n^{\Delta t}_{(l,h,r)}[P], k \lhd R) \equiv_p (n^{\Delta t}_{(l,h,r)}[Q], k \lhd R')$		
(S-Cap)	$P \equiv_p Q$ and $R \equiv_p R'$ implies $M^{\Delta t}.(P, k \lhd R) \equiv_p M^{\Delta t}.(Q, k \lhd R')$		
(S-Par Com)	if $weight(P) = 0$ then $P \mid Q \equiv_p Q \mid P$		
(S-Par Assoc)	$(P \mid Q) \mid R \equiv_p P \mid (Q \mid R)$		
(S-Repl Par)	$*(k \lhd P) \equiv_p k \lhd P \mid *(k \lhd P)$		
(S-Res Res)	$(\nu n)(\nu m)P \equiv_p (\nu m)(\nu n)P$ if $n \neq m$		
(S-Res LPar)	$(\nu n)(P \mid Q) \equiv_p P \mid (\nu n)Q$ if $(n) \notin fn(P)$		
(S-Res RPar)	$(\nu n)(P \mid Q) \equiv_p (\nu n)P \mid Q$ if $(n) \notin fn(Q)$		
(S-Res Amb)	$(\nu n)(m^{\Delta t}_{(l,h,r)}[P], k \lhd Q) \equiv_p (m^{\Delta t}_{(l,h,r)}[(\nu n)P], k \lhd Q)$ if $n \neq m$		
(S-Zero Par)	$P \mid \mathbf{0} \equiv_p P$	**(S-Zero Res)**	$(\nu n)\mathbf{0} \equiv_p \mathbf{0}$ **(S-Zero Repl)** $*\mathbf{0} \equiv_p \mathbf{0}$

Proposition 3.1 *If $P \equiv_p Q$ then $weight(P) = weight(Q)$.*

The timers of P in processes $M^{\Delta t}.(P, Q)$, $!\langle m\rangle^{\Delta t}.(P, Q)$ and $?(x)^{\Delta t}.(P, Q)$ are activated only after the consumption of $M^{\Delta t}$, $!\langle m\rangle^{\Delta t}$, and $?(x)^{\Delta t}$, respectively. We denote by $\not\to$ the fact that rules (R-Free), (R-Alloc), (R-In), (R-Out), (R-Open) and

(R-Com) cannot be applied. The behaviour of processes is given by the reduction rules of Table 4.

Table 4: *Reduction Rules*

(R-Idle) $$\frac{P \not\rightarrow}{P \rightarrow \phi_\Delta(P)}$$

(R-Free) $$\frac{-}{(n^{\Delta t}_{(l,h,r)}[R \mid \Diamond_k], k' \lhd Q) \rightarrow (n^{\Delta t}_{(l+k,h,r)}[\,R\,], k' \lhd Q)}$$

(R-Alloc) $$\frac{k \leq l}{(n^{\Delta t}_{(l,h,r)}[k \lhd Q], R) \rightarrow (n^{\Delta t}_{(l-k,h,r)}[Q], R)}$$

(R-In) $$\frac{h' \leq l'',\ r' \leq r'',\ m \in p(n, r')}{(n^{\Delta t'}_{(l',h',r')}[in^{\Delta t} m.(P, P') \mid Q], k' \lhd S') \mid (m^{\Delta t''}_{(l'',h'',r'')}[R], k'' \lhd S'') \rightarrow (m^{\Delta t''}_{(l''-h',h'',r'')}[(n^{\Delta t'}_{(l',h',r')}[P \mid Q], k' \lhd S') \mid R], k'' \lhd S'')}$$

(R-Out) $$\frac{h'' \leq l,\ p(n, r'') \cap p(m, r') \neq \emptyset}{(k^{\Delta t}_{(l,h,r)}[(m^{\Delta t'}_{(l',h',r')}[(n^{\Delta t''}_{(l'',h'',r'')}[out^{\Delta t} m.(P, P') \mid Q], k'' \lhd S'') \mid R], k' \lhd S')], k \lhd S) \rightarrow (k^{\Delta t}_{(l-h'',h,r)}[(n^{\Delta t''}_{(l'',h'',r'')}[P \mid Q], k'' \lhd S'') \mid (m^{\Delta t'}_{(l'+h'',h',r')}[R], k' \lhd S')], k \lhd S)}$$

(R-Open) $$\frac{-}{(m^{\Delta t'}_{(l',h',r')}[open^{\Delta t} n.\,(P, P') \mid (n^{\Delta t''}_{(l'',h'',r'')}[Q], k'' \lhd S'')], k' \lhd S') \rightarrow (m^{\Delta t'}_{(l'+l'',h',r')}[P \mid Q], k' \lhd S')}$$

(R-Com) $$\frac{-}{!\langle m\rangle^{\Delta t}.(P, Q) \mid ?(x)^{\Delta t'}.(P', Q') \rightarrow P \mid P'\{m/x\}}$$

(R-Amb) $$\frac{P \rightarrow Q}{(n^{\Delta t}_{(l,h,r)}[P], k \lhd R) \rightarrow (n^{\Delta t}_{(l,h,r)}[Q], k \lhd R)}$$

(R-LPar) $$\frac{P \rightarrow Q}{R \mid P \rightarrow R \mid Q}$$ (R-RPar) $$\frac{P \rightarrow Q}{P \mid R \rightarrow Q \mid R}$$

(R-Par) $$\frac{P \rightarrow Q,\ P' \rightarrow Q'}{P \mid P' \rightarrow Q \mid Q'}$$ (R-Res) $$\frac{P \rightarrow Q}{(\nu n)P \rightarrow (\nu n)Q}$$

(R-Sum) $$\frac{P \rightarrow P'}{P + Q \rightarrow P' + Q}$$ (R-Struct) $$\frac{P' \equiv_p P,\ P \rightarrow Q,\ Q \equiv_p Q'}{P' \rightarrow Q'}$$

The function ϕ_Δ decreases the timers, and for expired timers it discards the actions, capabilities and ambients. If one process evolves by one of the rules (R-In), (R-Out), (R-Open) and (R-Amb), while another one does not perform any reduction, then one of the rules (R-LPar), (R-RPar) should be applied. If more than one process evolve in parallel by applying one of the rules (R-In), (R-Out), (R-Open) and (R-Amb), then rule (R-Par) should be applied. When all the ambients become passive, the rule (R-Idle) is applied to simulate the passage of time, and so permitting the ambients to participate in other reductions in the next unit of time.

The resources are preserved through reduction only in a *closed* system, namely a system which is surrounded by an ambient which cannot be opened, and any ambient can pass through it. In an *open* system, the resources are not preserved through reduction because a process may acquire new resources from the environment, or transfer resources to the environment. Some resources may become restricted, and so unavailable for any other process, e.g. $(\nu n)(n^{\Delta t}_{(l,h,r)}[P], k \lhd Q)$.

We denote by $P \xrightarrow{\phi_\Delta}_t Q$ the fact that P evolves to Q after applying the rule (R-Idle) for $t \geq 0$ times, and with $t\phi_\Delta(R)$ the fact that function ϕ_Δ is applied t times to R. The passage of time cannot cause a nondeterministic behaviour.

Proposition 3.2 *If $P \equiv_p Q$, $P \xrightarrow{\phi_\Delta}_t P'$ and $Q \xrightarrow{\phi_\Delta}_t Q'$ then $P' \equiv_p Q'$.*

We can define several bisimulations in tMA, and some of them can require to match the timing aspects. Some bisimulations in tMA are presented in [2].

4 Interaction in Timed Mobile Ambients

We consider the example of an ambient *student* using any of the three transport ambients *bus*, *tram* and *cab* to reach a given location. Each of these entities is encoded in an ambient having the corresponding label. For an ambient A, $free\_resources(A)$ represents the (dynamically evolving) capacity l of A. If *bus* is in the proximity of radius r of the *student* ($bus \in p(student, r)$), and it has free resources ($free\_resource(bus) > 0$) then the *student* enters the *bus*. If the *bus* does not have free resources and the *tram* is in the proximity of radius r of the *student*, and it has free resources ($free\_resources(tram) > 0$) then the *student* enters the *tram*. If the *tram* does not have free resources and the *cab* is in the proximity of the *student*, and it has free resources ($free\_resources(cab) > 0$) then the *student* enters the *cab*. If the *tram* is not in the proximity of the *student*, the time needed for the *tram* to enter that proximity is greater that the time the *student* decided to wait for the *tram*, the *cab* is in the proximity of the *student* and it has free resources $free\_resources(cab) > 0$, then the *student* enters the *cab*.

If *bus* is not in the proximity of *student*, the time needed for the *bus* to enter that proximity is greater than the time the *student* decided to wait for the *bus*, the *tram* is in the proximity of radius r of the *student* and it has free resources ($free\_resource(tram) > 0$) then the *student* enters the *tram*. If the *tram* does not have free resources and the *cab* is in the proximity of radius r of the *student* and it has free resources ($free\_resources(cab) > 0$), then the *student* enter the *cab*. If the *tram* is not in the proximity of the *student*, the time needed for the *tram* to enter that proximity is greater that the time the *student* decided to wait for the *tram*, then the *student* searches for a *cab*; if the *cab* is in the proximity of the *student* and it has free resources $free\_resources(cab) > 0$, then the *student* enters the *cab*. If none of the above conditions holds, the time-stepping function ϕ_Δ is applied (simulating the passing of time), and then the above conditions are re-checked.

For this scenario the *bus* ambient can be described as:

$$\begin{array}{lcl} bus\_schedule & = & in^{\Delta t_1}\, univ.out^{\Delta t_2}\, univ.in^{\Delta t_3}\, camp.out^{\Delta t_2}\, camp.bus\_schedule \\ bus & = & bus^{\infty}_{(l\_bus, h\_bus, r\_bus)}[bus\_schedule] \end{array}$$

where

- Δt_1 - time the *bus* ambient needs to reach the *univ* starting from *camp*;
- Δt_2 - time the *bus* ambient awaits at a stop;
- Δt_3 - time the *bus* ambient needs to reach the ambient *camp*;
- *l_bus* - the free resources of the *bus* ambient;
- *h_bus* - the capacity of the *bus* ambient;
- *r_bus* - the proximity radius of the *bus* ambient.

Similarly, the *tram* ambient can be described as follows:

$$\begin{array}{lcl} tram\_schedule & = & in^{\Delta t_4} univ.out^{\Delta t_5} univ.in^{\Delta t_6} camp.out^{\Delta t_5} camp.tram\_schedule \\ tram & = & tram^{\infty}_{(l\_tram,h\_tram,r\_tram)}[tram\_schedule] \end{array}$$

The *cab* and the *client* can be described as in [9]; since our example contains only two locations, the description can be simplified. The *cab* can be described as follows:

$$\begin{array}{lcl} cab\_route & = & out^{\Delta t_7} univ.in^{\Delta t_8} camp.cab\_route + out^{\Delta t_7} camp.in^{\Delta t_9} univ.cab\_route \\ cab & = & cab^{\infty}_{(l\_cab,h\_cab,r\_cab)}[cab\_route] \end{array}$$

The description of *student* is more elaborated:

$$\begin{array}{lcl} student & = & student^{\infty}_{(1,1,r\_student)}[travel] \\ travel & = & in^{\infty} univ.travel\_univ + in^{\infty} camp.travel\_camp \\ travel\_camp & = & bus\_camp + tram\_camp + cab\_camp \\ travel\_univ & = & bus\_univ + tram\_univ + cab\_univ \\ cab\_camp & = & in^{\Delta t_{15}} cab.(out^{\Delta t_9} cab.out^{\Delta t_{17}} univ.travel, cab\_camp) \\ cab\_univ & = & in^{\Delta t_{14}} cab.(out^{\Delta t_8} cab.out^{\Delta t_{16}} camp.travel, cab\_univ) \\ tram\_camp & = & in^{\Delta t_{13}} tram.(out^{\Delta t_4} tram.out^{\Delta t_{17}} univ.travel, tram\_camp) \\ tram\_univ & = & in^{\Delta t_{12}} tram.(out^{\Delta t_6} tram.out^{\Delta t_{16}} camp.travel, tram\_univ) \\ bus\_camp & = & in^{\Delta t_{11}} bus.(out^{\Delta t_1} bus.out^{\Delta t_{17}} univ.travel, bus\_camp) \\ bus\_univ & = & in^{\Delta t_{10}} bus.(out^{\Delta t_3} bus.out^{\Delta t_{16}} camp.travel, bus\_univ) \end{array}$$

According to the above definitions, the *bus* and the *tram* move based on a predefined schedule, even they contain or do not contain *student* ambients inside, while the *cab* ambient has no (time) restriction to move. Each *cab* has a cyclic movement between the two locations *univ* and *camp* even it has or does not have a *student* inside. The *student* can enter in the ambient found in his proximity, with no constraint imposed by other ambients. In the moment the *student* is at *univ* location, we can have one of the following scenarios:

- two of the ambients *bus*, *tram* and *cab*, or all are at *univ* and at least one of them has free resources; in this case *student* enters one of them which has free resources, by choosing nondeterministically with no regard to the cost of the trip;
- one of the ambients *bus*, *tram* and *cab* are at the *univ* location and has a free resource; in this case the *student* enters it with no regard to the cost of the trip, or the schedule of the others;
- one, two or all of the ambients *bus*, *tram* and *cab* are at the *univ* location, but no free resources are available; in this case the *student* awaits for an ambient with free resources, and moves to be in its proximity.

The cost of the trip is not considered in all these scenarios. We can add priorities and interaction requirements acting over an initial assignment of the timers, capacities and proximities.

Over tMA we can define a coordination pair $(\mathcal{T}, \mathcal{CP})$. The first component $\mathcal{T}$ of this coordinating pair is a function assigning initial values to the timers, capacities and proximities. The second component $\mathcal{CR}$ is given by a set of rules. In our example the timer t_{10} should have initially a smaller value than the timer t_1, because the *student* is not willing to wait for a *bus* which has to cover a distance bigger or equal than the one from the two locations *univ* and *camp*. Having similar motivations, the timers t_{11}, t_{12} and t_{13} can be smaller that the initial value for t_3, t_4, respectively t_6. The rules given by CR influence the evolution of the system, and decide a certain

behaviour according to the requirements expressed by the rules. For example, the *student* establishes a strategy for lowering its travel costs by imposing his priority order: $bus > tram > cab$. Now we can impose the following rules:

$$[l\_cab \neq h\_cab], [l\_cab = h\_cab], [t_{10} < t_1], [t_{11} < t_3] \ ,$$

$$[t_{10} < t_1 \wedge t_{12} < t_4], [t_{11} < t_3 \wedge t_{13} < t_6] \ .$$

These rules could be integrated in the syntax, and so the processes *cab* and *student* can be rewritten. Since the *cab* should make a trip only when it has a *student* inside, and it should accept a *student* only after it completes the previous trip, we rewrite the syntax of the *cab* ambient as

$$\begin{array}{lcl} cab & = & cab^{\infty}_{(l\_cab, h\_cab, r\_cab)}[\ [l\_cab = h\_cab] cab\_route] \\ cab\_route & = & [l\_cab \neq h\_cab] out^{\Delta t_7} univ.in^{\Delta t_8} camp.[l\_cab = h\_cab] cab\_route \\ & + & [l\_cab \neq h\_cab] out^{\Delta t_7} camp.in^{\Delta t_9} univ.[l\_cab = h\_cab] cab\_route, \text{ where} \end{array}$$

- $l\_cab \neq h\_cab$ - denotes the fact that the *cab* has an *student* inside it;
- $l\_cab = h\_cab$ - denotes the fact that the *cab* is available.

The initial *student* ambient has no strategy of lowering its travel cost by choosing a appropriate travel ambient. The coordinated *student* taking care of this aspect has the following syntax:

$$\begin{array}{lcl} student & = & student^{\infty}_{(1,1,r\_student)}[travel] \\ travel & = & in^{\infty} univ.travel\_univ + in^{\infty} camp.travel\_camp \\ travel\_camp & = & bus\_camp + tram\_camp + cab\_camp \\ travel\_univ & = & bus\_univ + tram\_univ + cab\_univ \\ cab\_camp & = & [t_{11} < t_3 \wedge t_{13} < t_6] in^{\Delta t_{15}} cab. \\ & & (out^{\Delta t_9} cab.out^{\Delta t_{17}} univ.travel, cab\_camp) \\ cab\_univ & = & [t_{10} < t_1 \wedge t_{12} < t_4] in^{\Delta t_{14}} cab. \\ & & (out^{\Delta t_8} cab.out^{\Delta t_{16}} camp.travel, cab\_univ) \\ tram\_camp & = & [t_{11} < t_3] in^{\Delta t_{13}} tram. \\ & & (out^{\Delta t_4} tram.out^{\Delta t_{17}} univ.travel, tram\_camp) \\ tram\_univ & = & [t_{10} < t_1] in^{\Delta t_{12}} tram. \\ & & (out^{\Delta t_6} tram.out^{\Delta t_{16}} camp.travel, tram\_univ) \\ bus\_camp & = & in^{\Delta t_{11}} bus.(out^{\Delta t_1} bus.out^{\Delta t_{17}} univ.travel, bus\_camp) \\ bus\_univ & = & in^{\Delta t_{10}} bus.(out^{\Delta t_3} bus.out^{\Delta t_{16}} camp.travel, bus\_univ), \quad \textbf{where} \end{array}$$

- $t_{10} < t_1$ - compares the time (t_{10}) needed for the *bus* to reach *univ*, with the time (t_1) the *student* is willing to wait for *bus*;
- $t_{11} < t_3$ - compares the time (t_{11}) needed for the *bus* to reach *camp*, with the time (t_3) the *student* is willing to wait for *bus*;
- $t_{12} < t_4$ - compares the time (t_{12}) needed for the *tram* to reach *univ*, with the time (t_4) the *student* is willing to wait for *tram*;
- $t_{13} < t_5$ - compares the time (t_{13}) needed for the *tram* to reach *camp*, with the time (t_5) the *student* is willing to wait for *tram*.

The rules restrict the evolution of a system, acting when we have more than one interaction choice in a reduction step. In what follows we describe the possible evolutions of the system when the *student* is placed in the proximity of the *univ*

ambient, after rewriting the ambients *cab* and *student*. When *student* is at *univ* location, we have the following reduction:

$$student^{\infty}_{(1,1,r\_student)}[travel] \mid univ^{\infty}_{(l\_univ,h\_univ,r\_univ)}[\,] \\ \rightarrow univ^{\infty}_{(l\_univ-1,h\_univ,r\_univ)}[student^{\infty}_{(1,1,r\_student)}[travel\_univ]\,]$$

If one the following scenarios holds:

- the ambient *bus* having free resources and is in the proximity of *student*;
- the *bus* is not at *univ* location, $t_{10} > t_1$ (meaning that *student* waits for the *bus*) and the *bus* is arriving after t_1 units of time and has free resources;

the second reduction in the system is:

$$student^{\infty}_{(1,1,r\_student)}[travel\_univ] \mid bus^{\infty}_{(l\_bus,h\_bus,r\_bus)}[\,] \\ \rightarrow bus^{\infty}_{(l\_bus-1,h\_bus,r\_bus)}[student^{\infty}_{(1,1,r\_student)}[out^{\Delta t_3} bus.out^{\Delta t_{16}} camp.travel]\,]$$

If one the following scenarios holds:

- in the *univ* ambient there is no *bus* ambient; ambient *tram*, having free resources, is inside *univ* in the proximity of the *student* ambient, and $t_{10} < t_1$ (which means that the *student* is not willing to wait for the *bus*);
- *tram* is not inside *univ*, $t_{12} > t_4$ (*student* is willing to wait for a *tram*), $t_{10} < t_1$ and the *tram* is arriving after t_4 units of time, and has free resources;

the second reduction in the system is:

$$student^{\infty}_{(1,1,r\_student)}[travel\_univ] \mid tram^{\infty}_{(l\_tram,h\_tram,r\_tram)}[\,] \\ \rightarrow tram^{\infty}_{(l\_tram-1,h\_tram,r\_tram)}[student^{\infty}_{(1,1,r\_student)}[out^{\Delta t_6} tram.out^{\Delta t_{16}} camp.travel]\,]$$

If one the following scenarios holds:

- in the *univ* ambient there is no *bus* or *tram* ambient; ambient *cab*, having free resources, is inside *univ* is in the proximity of the *student* ambient, $t_{10} < t_1$ (which means that the *student* is not willing to wait for the *bus*) and $t_{12} < t_4$ (which means that the *student* is not willing to wait for the *tram*);
- the *cab* is not inside *univ*, $t_{14} > t_9$ (*student* is willing to wait for a *cab*), $t_{10} < t_1$, $t_{12} < t_4$ and an available *cab* is arriving after t_9 units of time;

the second reduction in the system is:

$$student^{\infty}_{(1,1,r\_student)}[travel\_univ] \mid cab^{\infty}_{(l\_cab,h\_cab,r\_cab)}[\,] \\ \rightarrow cab^{\infty}_{(l\_cab-1,h\_cab,r\_cab)}[student^{\infty}_{(1,1,r\_student)}[out^{\Delta t_8} cab.out^{\Delta t_{16}} camp.travel]\,]$$

After the *student* enters one of the ambients *bus*, *tram* or *cab* by applying one of the three steps above, *student* is carried to the *camp* ambient where *student* is discarded. Then *student* exits the *camp* ambient, and we can have a similar scenario in order to move *student* from *camp* to *univ*.

5 Conclusion

Both timed distributed π-calculus and timed mobile ambients use discrete and relative time given by timers, based on local clocks whose timers decrease uniformly.

Timers are used to restrict the interaction between components, and they can be used to control the resource availability. In timed mobile ambients we have a hierarchical representation of space, and a more realistic description of the distributed computation and mobility. In timed mobile ambients the interaction between components depends also on a proximity function which suggest a dynamic topology related to an interaction field, and allow the interaction between two processes in a general context.

In both formalisms we can have a clear separation of the coordination aspects from the computation aspects. Interaction in time and space could be controlled by assigning specific values to timers, and by a set of rules restricting the evolution of the system, acting also when we have more than one interaction choice.

Acknowledgement

The author thanks the referees for their helpful comments. Many thanks to Cristian Prisacariu for his collaboration concerning timed distributed π-calculus, and to Bogdan Aman for his collaboration concerning timed mobile ambients.

References

[1] B. Aman, G. Ciobanu. Timers and Proximities for Mobile Ambients. *Proceedings CSR*, Lecture Notes in Computer Science vol.4649, Springer, 33–43, 2007.

[2] B. Aman, G. Ciobanu. Mobile Ambients with Timers and Types. *Proceedings ICTAC*, Lecture Notes in Computer Science vol.4711, Springer, 50–63, 2007.

[3] L. Cardelli, A. Gordon. Mobile Ambients. *Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science vol.1378, Springer, 140–155, 1998.

[4] G. Ciobanu, C. Prisacariu. Timers for Distributed Systems. *Electronic Notes in Theoretical Computer Science* vol.164, 81–99, 2006.

[5] G. Ciobanu, C. Prisacariu. Coordination by Timers for Channel-Based Anonymous Communications. *Electronic Notes in Theoretical Computer Science*, vol.175, 3-17, 2007.

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

[7] R. Milner. *Communicating and Mobile Systems: the π-calculus.* Cambridge University Press, 1999.

[8] G.A. Papadopoulos, F. Arbab. Coordination Models and Languages. *Advances in Computers* vol.46, Academic Press, 329–400, 1998.

[9] D. Teller, P. Zimmer, D. Hirschkoff. Using Ambients to Control Resources. *Concurrency Theory* , Lecture Notes in Computer Science vol.2421, Springer, 288–303, 2002.