Timed Mobile Ambients for Network Protocols

Bogdan Aman[2] and Gabriel Ciobanu[1,2]

[1] "A.I.Cuza" University, Faculty of Computer Science
Blvd. Carol I no.11, 700506 Iaşi, Romania
[2] Romanian, Academy, Institute of Computer Science
Blvd. Carol I no.8, 700505 Iaşi, Romania
baman@iit.tuiasi.ro, gabriel@info.uaic.ro

Abstract. Ambient calculus is a calculus for mobile computing able to express local communications inside hierarchical domains. So far the timing properties have not been considered in the framework of mobile ambients. We add timers to capabilities and ambients, and provide an operational semantics of the new calculus. Certain results are related to the passage of time, and some new behavioural equivalences over timed mobile ambients are defined. Timeout for network communication (TTL) can be naturally modelled by the time constraints over capabilities and ambients. The new formalism can be used to describe network protocols; Simple Network Management Protocol (SNMP) may implement its own strategy for timeout and retransmission in TCP/IP.

1 Introduction

Ambient calculus is a formalism for describing distributed and mobile computation introduced in [6]. In contrast with other formalisms for mobile processes such as the π-calculus [19] whose computational model is based on the notion of *communication*, the ambient calculus is based on the notion of *movement*. An ambient represents a unit of movement. Ambient mobility is controlled by the capabilities *in, out*, and *open*. Capabilities are similar to prefixes in CCS [18] and π-calculus. Several variants of the ambient calculus have been proposed by adding and/or removing features of the original calculus [5, 15, 17].

The definition of mobile ambients is related in [6] to the network communication. Ambient calculus can model communication protocols. Timing properties are important in network communication. For instance, a *Time to Live* (TTL) value is used to indicate the timeout for a communication unit before it should be discarded. Servers do not apply a single fixed timeout for all communication units. Simple Network Management Protocol (SNMP) could implement its own strategy for timeout and retransmission in TCP/IP communication protocol. TTL and retransmission in TCP/IP protocol provide a good motivation to add timers to ambients. So far the timing properties have not been considered in the framework of mobile ambients. In this paper we associate timers not only to ambients, but also to capabilities. The resulting formalism is called timed mobile ambients (tMA), and represents a conservative extension of the ambient calculus.

We use a clock just for the sake of uniformity; all the clocks work in the same way. In fact, working with located processes and a migration primitive *go*, we use only local clocks, and the change from one local clock to another one is possible by the migration primitive *go*. This is sound because we use relative time given by timers, and not an absolute time.

The structure of the paper is as follows. Section 2 introduces the pure mobile ambients followed by the description of the timed mobile ambients (tMA). The passage of time is given by a discrete time progress function. We provide an operational semantics of the new calculus given by a reduction relation. In Section 3 we use tMA to describe the Transmission Control Protocol (TCP). In Section 4 we introduce and study some behavioural equivalences over timed mobile ambients. Other results are related to the passage of time. Conclusion and references end the paper.

2 Mobile Ambients with Time Constraints

We provide a short description of the pure mobile ambients, an algebraic formalism which studies the distributed concurrent systems; more information can be found in [6]. The following table describes the syntax of mobile ambients.

Table 1: *Mobile Ambients Syntax*

n, m, p		ambient names	P, Q ::=		processes
C ::=		capabilities		$\mathbf{0}$	inactivity
	$in\ n$	can enter n		$C.P$	movement
	$out\ n$	can exit n		$n[P]$	ambient
	$open\ n$	can open n		$P \mid Q$	composition
				$(\nu n)P$	restriction
				$*P$	replication

Process $\mathbf{0}$ is an inactive process (it does nothing). A movement $C.P$ is provided by the capability C, followed by the execution of P. An ambient $n[P]$ represents a bounded place labelled by n in which a process P is executed. $P \mid Q$ is a parallel composition of processes P and Q. $(\nu n)P$ creates a new unique name n within the scope of P. $*P$ denotes the unbounded replication of a process P, producing as many parallel replicas of P as needed.

The semantic of the ambient calculus is given by two relations: structural congruence and reduction. The *structural congruence* $P \equiv Q$ relates different syntactic representations of the same process; it is used to define the reduction relation. The *reduction relation* $P \to Q$ describes the system evolution. We denote by $\to^*$ the reflexive and transitive closure of $\to$.

The structural congruence is defined as the least relation over processes satisfying the axioms from the table below:

Table 2: *Structural congruence*

$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	$P \equiv Q$ implies $Q \equiv P$
$P \mid Q \equiv Q \mid P\ ,\quad *P \equiv P \mid *P$	$P \equiv Q,\ Q \equiv R$ implies $P \equiv R$
$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$ if $n \neq m$	$P \equiv Q$ implies $(\nu n)P \equiv (\nu n)Q$
$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q$ if $n \notin fnAmb(P)$	$P \equiv Q$ implies $P \mid R \equiv Q \mid R$
$(\nu n)m[P] \equiv m[(\nu n)P]$ if $n \neq m$	$P \equiv Q$ implies $*P \equiv *Q$
$P \equiv P\ ,\quad P \mid \mathbf{0} \equiv P$	$P \equiv Q$ implies $n[P] \equiv n[Q]$
$(\nu n)\mathbf{0} \equiv \mathbf{0}\ ;\quad *\mathbf{0} \equiv \mathbf{0}$	$P \equiv Q$ implies $C.P \equiv C.Q$

The rules from the left side of the table describe the commutativity/ associativity of composition, unfolding recursion, changing the restriction scope. The rules from the right side describe how structural congruence is propagated across processes. The set of free names for a process is defined as follows:

$$fnAmb(P) = \begin{cases} \emptyset & \text{if } P = \mathbf{0} \\ fnAmb(R) \cup \{n\} & \text{if } P = cap\ n.R, \text{ with } cap \in \{in, out, open\} \\ fnAmb(R) \cup \{n\} & \text{if } P = n[R] \\ fnAmb(R) \cup fnAmb(Q) & \text{if } P = R \mid Q \\ fnAmb(R) - \{n\} & \text{if } P = (\nu n)R \\ fnAmb(R) & \text{if } P = *R \end{cases}$$

The reduction relation is defined as the least relation over processes satisfying the following set of axioms and rules:

Table 3: *Reduction Rules*

(In)	$n[in\ m.\, P \mid Q] \mid m[R] \to m[n[P \mid Q] \mid R]$
(Out)	$m[n[out\ m.\, P \mid Q] \mid R] \to n[P \mid Q] \mid m[R]$
(Open)	$open\ n.\, P \mid n[Q] \to P \mid Q$
(Res)	$P \to Q$ implies $(\nu n)P \to (\nu n)Q$
(Amb)	$P \to Q$ implies $n[P] \to n[Q]$
(Par)	$P \to Q$ implies $P \mid R \to Q \mid R$
(Struct)	$\dfrac{P' \equiv P,\ P \to Q,\ Q \equiv Q'}{P' \to Q'}$

The first three rules are the reductions for *in, out, open.* The next three rules propagate reductions across scopes, ambient nesting and parallel composition. The final rule allows the use of structural congruence during reduction.

We ignore the communications inside ambients and use pure mobile ambients to express the time and space constraints. We can also easily introduce channels and study the aspects related to them, with no difference in expressing the network protocols.

In order to identify an entity, TCP/IP protocols use the IP address, which uniquely identifies the connection of a host to the Internet. However, people prefer to use names instead of numeric addresses. A system that can map a name to an address or an address to a name is the domain name system, which is represented hierarchically in what follows:

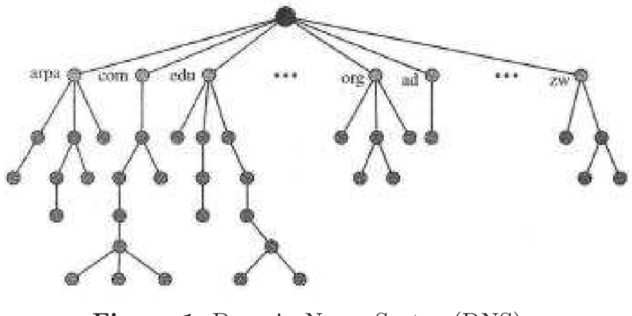

Figure 1. Domain Name System(DNS)

The information contained in DNS must be stored. One way to do this is to divide the whole space into many domains based on the first level.

Inspired from the domain name system, we also consider a distribution of parallel locations between which the ambients can migrate, each location being the place where the nested ambients interact. In our model the root node, represented in the above picture, disappears and its function is supplied by the execution of the migration primitive *go*. Thus we get a more realistic description of the distributed computation and mobility. A natural example motivating an extension from timed distributed π-calculus to timed mobile ambients is presented in [7].

The syntax of the timed mobile ambients is defined in Table 4.

Table 4: *Syntax of tMA*

$a, b, \ldots$		names	$P, Q ::=$	processes
C $::=$		capabilities	$\mathbf{0}$	inactivity
	$in\ n$	can enter an ambient n	$C^{\Delta t}.\,(P, Q)$	movement
	$out\ n$	can exit an ambient n	$(n^{\Delta t}[P]^{\mu}, Q)$	ambient
	$open\ n$	can open an ambient n	$P \mid Q$	composition
	$go\ k$	migration	$(\nu n)P$	restriction
M, N $::=$		located processes	$*P$	replication
	$l[[P]]$	location		
	$(\nu k)M$	restriction		
	$M \mid N$	composition		

We use m, n for *ambient names*; k, l for *physical locations*; a, p for *ambient tags* - a stands for *active* ambients, while p stands for *passive* ambients - we use μ to stand for both ambient tags.

In timed Mobile Ambients (tMA) capabilities and ambients are used as temporal resources; if nothing happens in a predefined interval of time, the waiting process goes to another state. Since the expiration of a timer offers an alternative, we shall not use the choice operator as in other process calculi. The timer Δt of each temporal resource indicates that the resource is available only for a determined period of time t. We add timers to both ambients and capabilities. A process can be executed only if it is inside a location. When an ambient migrates between locations, all the processes running inside suspend their execution until the ambient reaches its destination.

We write $n^{\Delta t}[P]^{\mu}$ to denote an ambient having the timer Δt and the tag μ. The tag μ is a neutral tag that indicates if an ambient is active or passive. The novelty comes from the fact that an ambient can disappear. If $t > 0$ the ambient behaves exactly as in untimed mobile ambients. Since the timer Δt can expire ($t = 0$) we use a pair $(n^{\Delta t}[P]^{\mu}, Q)$ to denote a timed ambient, where Q is a *safety process*. If nothing happens in t units of time, the ambient n is dissolved, the process P running inside the ambient is reduced to $\mathbf{0}$, and the process Q is executed. If $Q = \mathbf{0}$ we can simply write $n^{\Delta t}[P]^{\mu}$ instead of $(n^{\Delta t}[P]^{\mu}, Q)$. Similarly, for movement, we use a pair of processes. The process $open^{\Delta t} n.(P, Q)$ evolves to P whenever, in the period of time Δt, the process becomes sibling to an ambient n; otherwise it evolves to Q.

When we describe initially the ambients, we consider that all ambients are active, and we associate the tag a to them. From Table 4 it can be seen that we consider only ambients to be placed at some locations.

2.1 Semantics

The main feature of tMA is given by the explicit use of time. The passage of time is described by two discrete time progress functions: $\varPhi_\Delta$ defined over the set $\mathcal{L}$ of located processes, and ϕ_Δ defined over the set $\mathcal{P}$ of timed processes. The possible actions are performed at every tick of a universal clock. The function ϕ_Δ, inspired from [3], affects the ambients, and the capabilities which are not consumed. The consumed capabilities and ambients disappear together with their timers. If a capability or ambient has the timer equal to ∞, thus simulating the behaviour of an untimed capability or ambient, we use the equality $\infty - 1 = \infty$ when applying the function ϕ_Δ. This function modifies a process accordingly with the passage of time. Another property of the time progress function ϕ_Δ is that the passive ambients can become active in the next unit of time in order to participate in other reductions.

For the process $C^{\Delta t}.(P, Q)$ the timer of P is activated only after the consumption of capability $C^{\Delta t}$ (in at most t units of time). Reduction rules (Table 6) show how the time function $\varPhi_\Delta$ is used.

Definition 1. *(Global time progress function) We define $\varPhi_\Delta : \mathcal{L} \to \mathcal{L}$, by:*

$$\varPhi_\Delta(M) = \begin{cases} \varPhi_\Delta(M_1) \mid \varPhi_\Delta(M_2) & \text{if } M = M_1 \mid M_2 \\ (\nu k)\varPhi_\Delta(N) & \text{if } M = (\nu k)N \\ l[[\phi_\Delta(P)]] & \text{if } M = l[[P]] \end{cases}$$

where the function $\phi_\Delta : \mathcal{P} \to \mathcal{P}$ has the following definition:

$$\phi_\Delta(P) = \begin{cases} C^{\Delta(t-1)}.\,(R, Q) & \text{if } P = C^{\Delta t}.\,(R, Q),\ t > 0 \\ Q & \text{if } P = C^{\Delta t}.\,(R, Q),\ t = 0 \\ \phi_\Delta(R) \mid \phi_\Delta(Q) & \text{if } P = R \,|\, Q \\ (\nu n)\phi_\Delta(R) & \text{if } P = (\nu n)R \\ (n^{\Delta(t-1)}[\phi_\Delta(R)]^a, Q) & \text{if } P = (n^{\Delta t}[R]^\mu, Q),\ t > 0 \\ Q & \text{if } P = (n^{\Delta t}[R]^\mu, Q),\ t = 0 \\ P & \text{if } P = *R \text{ or } P = \mathbf{0} \end{cases}$$

Processes are grouped into equivalence classes by the following equivalence relation, $\varXi$, called structural congruence. This relation provides a way of rearranging expressions so that interacting parts can be brought together.

Table 5: *Structural Congruence in tMA*

(S-Sym)	$P \varXi Q$ implies $Q \varXi P$	**(S-Refl)**	$P \varXi P$
(S-Trans)	$P \varXi R$, $R \varXi Q$ implies $P \varXi Q$	**(S-Par Assoc)**	$(P \,\vert\, Q) \,\vert\, R \varXi P \,\vert\, (Q \,\vert\, R)$
(S-Res)	$P \varXi Q$ implies $(\nu n)P \varXi (\nu n)Q$	**(S-Repl Par)**	$*P \varXi P \,\vert\, *P$
(S-Par)	$P \varXi Q$ implies $R \,\vert\, P \varXi R \,\vert\, Q$	**(S-Zero Par)**	$P \,\vert\, \mathbf{0} \varXi P$
(S-Par Com)	$P \,\vert\, Q \varXi Q \,\vert\, P$	**(S-Zero Res)**	$(\nu n)\mathbf{0} \varXi \mathbf{0}$
(S-Repl)	$P \varXi Q$ implies $*P \varXi *Q$	**(S-Zero Repl)**	$*\mathbf{0} \varXi \mathbf{0}$
(S-Amb)	$P \varXi Q$ and $R \varXi R'$ implies $(n^{\Delta t}[P]^\mu, R) \varXi (n^{\Delta t}[Q]^\mu, R')$		
(S-Loc)	$P \varXi Q$ implies $k[[P]] \varXi k[[Q]]$		
(S-Par Loc)	$k[[P \mid Q]] \varXi k[[P]] \mid k[[Q]]$		
(S-Cap)	$P \varXi Q$ and $R \varXi R'$ implies $C^{\Delta t}.\,(P, R) \varXi C^{\Delta t}.\,(Q, R')$		
(S-Res Res)	$(\nu n)(\nu m)P \varXi (\nu m)(\nu n)P$ if $n \neq m$		
(S-Res Par)	$(\nu n)(P \,\vert\, Q) \varXi P \,\vert\, (\nu n)Q$ if $n \notin fnAmb(P)$		
(S-Res Par Loc)	$(\nu k)(M \,\vert\, N) \varXi M \,\vert\, (\nu k)N$ if $k \notin fnLoc(M)$		
(S-Res Amb)	$(\nu n)(m^{\Delta t}[P]^\mu, Q) \varXi (m^{\Delta t}[(\nu n)P]^\mu, Q)$ if $m \neq n$ and $n \notin fnAmb(Q)$		

The set of free names for a located process is defined as follow:

$$fnLoc(M) = \begin{cases} fnLoc(P) \cup \{k\} & \text{if } M = k[[P]] \\ fnLoc(N_1) \cup fnLoc(N_2) & \text{if } M = N_1 \mid N_2 \\ fnLoc(N) - \{k\} & \text{if } M = (\nu k)N \end{cases} \text{, where}$$

$$fnLoc(P) = \begin{cases} \emptyset & \text{if } P = \mathbf{0} \\ fnLoc(R) \cup \{k\} & \text{if } P = go^{\Delta t}k.(R.R') \\ fnLoc(R) \cup fnLoc(Q) & \text{otherwise} \end{cases}$$

We denote by $\not\dashrightarrow$ the fact that none of the rules from the following Table, except the rule **(R-TimePass)** can be applied. The behaviour of processes is given by the following reduction rules:

Table 6: *Reduction rules*

$$\textbf{(R-Migrate)}\ \frac{t' = 1}{l[[(n^{\Delta t}[go^{\Delta t'}k.(P,P')]^{a}, Q)]] \dashrightarrow k[[(n^{\Delta t}[P]^{p}, Q)]]}$$

$$\textbf{(R-In)}\ \frac{-}{\begin{array}{c}(n^{\Delta t'}[in^{\Delta t}m.(P,P') \mid Q]^{a}, S') \mid (m^{\Delta t''}[R]^{\mu}, S'') \dashrightarrow \\ (m^{\Delta t''}[(n^{\Delta t'}[P \mid Q]^{p}, S') \mid R]^{\mu}, S'')\end{array}}$$

$$\textbf{(R-Out)}\ \frac{-}{\begin{array}{c}(m^{\Delta t'}[(n^{\Delta t''}[out^{\Delta t}m.(P,P') \mid Q]^{a}, S'') \mid R]^{\mu}, S') \dashrightarrow \\ (n^{\Delta t''}[P \mid Q]^{p}, S'') \mid (m^{\Delta t'}[R]^{\mu}, S')\end{array}}$$

$$\textbf{(R-Open)}\ \frac{-}{open^{\Delta t}m.\,(P,P') \mid (m^{\Delta t'}[Q]^{\mu}, S') \dashrightarrow P \mid Q}$$

$$\textbf{(R-Amb)}\ \frac{P \dashrightarrow Q}{(n^{\Delta t}[P]^{\mu}, R) \dashrightarrow (n^{\Delta t}[Q]^{\mu}, R)} \qquad \textbf{(R-Par1)}\ \frac{P \dashrightarrow Q}{P \mid R \dashrightarrow Q \mid R}$$

$$\textbf{(R-Par2)}\ \frac{P \dashrightarrow Q,\ P' \dashrightarrow Q'}{P \mid P' \dashrightarrow Q \mid Q'} \qquad \textbf{(R-Res)}\ \frac{P \to Q}{(\nu n)P \dashrightarrow (\nu n)Q}$$

$$\textbf{(R-Struct)}\ \frac{M' \Xi M,\ M \dashrightarrow N,\ N \Xi N'}{M' \dashrightarrow N'} \qquad \textbf{(R-Loc)}\ \frac{P \dashrightarrow Q}{l[[P]] \dashrightarrow l[[Q]]}$$

$$\textbf{(R-LocPar1)}\ \frac{M \dashrightarrow M'}{M \mid N \dashrightarrow M' \mid N} \qquad \textbf{(R-LocPar2)}\ \frac{M \dashrightarrow M',\ N \dashrightarrow N'}{M \mid N \dashrightarrow M' \mid N'}$$

$$\textbf{(R-LocRes)}\ \frac{M \dashrightarrow M'}{(\nu k)M \dashrightarrow (\nu k)M} \qquad \textbf{(R-TimePass)}\ \frac{M \not\dashrightarrow}{M \dashrightarrow \Phi_{\Delta}(M)}$$

In the rules **(R-In)**, **(R-Out)**, **(R-Open)** ambient m can be *passive* or *active*, while in the rules **(R-Migrate)**, **(R-In)**, **(R-Out)** ambient n is *active*. The difference between *passive* and *active* ambients is that the *passive* ambients can be used in several reductions in a unit of time, while the *active* ambients can be used in at most one reduction in a unit of time, by consuming their capabilities. In the rules **(R-In)**, **(R-Out)** the *active* ambient n becomes *passive*, forcing it to consume only one capability in one unit of time. The ambients which are tagged as *passive*, become *active* again by applying the global time-stepping function **(R-TimePass)**. We use the tag μ in these rules because it does not matter whether or not the ambient is passive or active.

In the rules **(R-Migrate)** if the physical location k does not exist then it is created. Rule **(R-Migrate)** simulates the movement of an *active* ambient n from location l to location k in order to interact with some ambient located at

k; notice that the ambient tag changes to p, meaning that the ambient becomes *passive*.

In timed mobile ambients, if a process evolves by one of the rules **(R-In)**, **(R-Out)**, **(R-Open)**, **(R-Migrate)**, while another one does not perform any reduction, then rule **(R-Par1)** should be applied. If more than one process evolve in parallel by applying one of the rules **(R-In)**, **(R-Out)**, **(R-Open)**, **(R-Migrate)**, then the rule **(R-Par2)** should be applied. We use the rule **(R-Par2)** to compose processes that are active, and the rule **(R-Par1)** to compose processes that are active and passive. An example for the usage of the rule **(R-Par1)** is given by:

$$\frac{m^{\Delta t_1}[Q]^{\mu} \mid open^{\Delta t_2} m \rightarrow Q}{m^{\Delta t_1}[Q]^{\mu} \mid open^{\Delta t_2} m \mid in^{\Delta t_3} t \rightarrow Q \mid in^{\Delta t_3} t}$$

A similar argument can be used for arguing in case of the rules **(R-LocPar1)** and **(R-LocPar2)**. The rule **(R-LocRes)** propagate reductions across location scopes. In Section 3 illustrate how some of the rules from Table 6 are working.

We can say that a system described with tMA satisfies the properties [13]:

- **Time Determinism**: at each time only one reduction rule can be applied. A possible problem could appear only if we apply **(R-TimePass)** when we can apply another rule. However this is not possible because **(R-TimePass)** is applied only if the process does not evolve ($\not\rightarrow$).
- **Maximal Progress**: a process cannot delay if it can evolve.
- **Time Continuity**: to go from a process P at time t, to a process P_0 at time $t + \Delta t$, we must go through all the intermediate time steps of the interval $[t, t + \Delta t]$.

3 Transmission Control Protocol

Transmission Control Protocol (TCP) is a connection-oriented protocol. Using TCP applications on networked hosts can establish connections to one another, over which they can exchange data. The protocol is reliable and delivers the data from sender to receiver in the order it has been sent. TCP distinguishes data from multiple connections made by concurrent applications running on the same host.

TCP needs to establish a connection before sending data. To establish a connection, TCP uses a *three-way handshake*. In order for a client to connect to a server, the server must first open a port for the connection: this is called a *passive open*. A client can initiate an *active open*, only after the passive open is established. TCP connections have three phases:

1. the active open is performed by sending a synchronization packet (SYN flag set) to the server;
2. the server replies with a packet (SYN and ACK flag set);
3. the client sends a packet (ACK flag set) back to the server.

After all these steps are performed, both the client and the server have received an acknowledgement of the connection and the data transfer can begin.

The connection termination phase uses, at most, a *four-way handshake*. This is caused by TCP's *half closed*. Since a TCP connection is full-duplex (data can flow in each direction independently of the other direction), each direction must be shut down independently. When an endpoint wishes to stop its half of the connection, it transmits a FIN packet, which the other end acknowledges with an ACK. The receipt of a FIN only means that there can be no more data flowing in that direction. A TCP can still send data after receiving a FIN. Therefore, a connection termination requires a pair of FIN and ACK segments from each TCP endpoint.

It is also possible to terminate the connection by a 3-way handshake, when a process sends a FIN and the other host replies with a FIN & ACK (merely combines 2 steps into one) and first host replies with an ACK. This is perhaps the most common method.

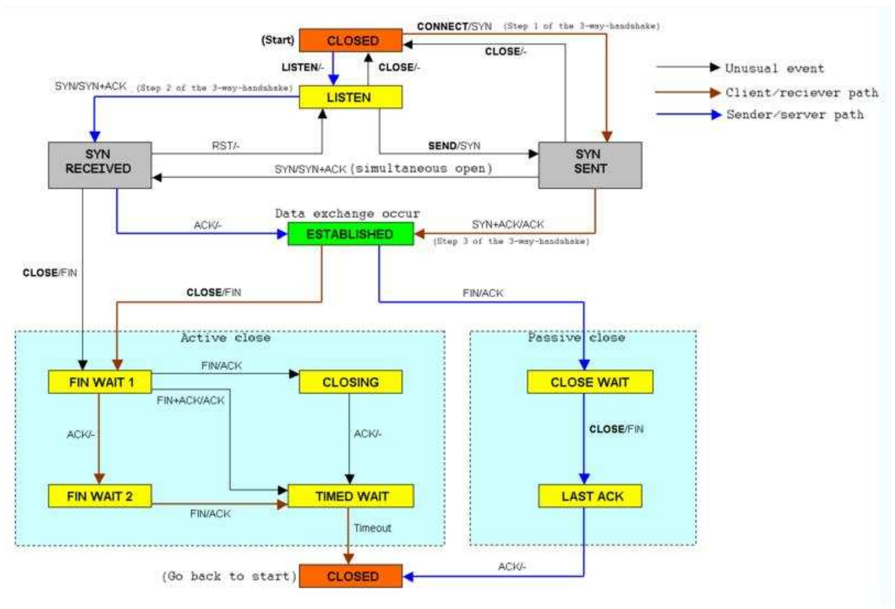

Figure 2. TCP State Diagram

Every implementation must choose a value for its *maximum segment lifetime*. It is the maximum amount of time any segment can exist in the network before being discarded; this justifies why we have added timers to ambients. We know this time limit is bounded, since TCP segments are transmitted as IP datagrams, and the IP datagram has the TTL field that limits its lifetime. (RFC 793 specifies the MSL as 2 minutes. Common implementation values, however, are 30 seconds, 1 minute, or 2 minutes. [21])

In what follows, we represent TCP in tMA, when only a client and a server are involved. For simplicity, we do not add the safety process to the capabilities which we know for sure that they are going to be consumed, and for the ambients which have the timer ∞.

Table 7: *Transmission Control Protocol represented in tMA*

$system := l_1[[client^{\infty}[send \mid send\_ack]^{\mu_1}]] \mid l_2[[server^{\infty}[receive]^{\mu_2}]]$
$send =$
$\quad SYN^{\Delta t_1}[out^{\Delta t_2} client.go^{\Delta t} l_2.in^{\Delta t_3} server]^{\mu_3}$
$send\_ack =$
$\quad open^{\Delta t_4} SYNACK.\ (ACK^{\Delta t_5}[out^{\Delta t_6} client.go^{\Delta t} l_2.in^{\Delta t_7} server]^{\mu_4}, send \mid send\_ack)$
$receive =$
$\quad open^{\Delta t_8} SYN.\ (SYNACK^{\Delta t_9}[out^{\Delta t_{10}} server.go^{\Delta t} l_1.in^{\Delta t_{11}} client]^{\mu_5} \mid receive)$

We write *send*, *send_ack* and *receive* processes to simulate the *three-way handshake* for establishing the connection. The transmission of data and the end of the connection could be represented in a similar way.

The *client* tries to connect to the *server* by sending an ambient SYN. If $\mu_3 = a$, then the capability $out^{\Delta t_2} client$ can be executed immediately such that we do not use a safety process. This is realized by applying a rule **(R-Out)**

$$client^{\infty}[(SYN^{\Delta t_1}[out^{\Delta t_2} client.go^{\Delta t} l_2.\ldots]^{a}, \ldots) \mid \ldots]^{\mu_1}$$
$$\dashrightarrow client^{\infty}[\ldots]^{\mu_1} \mid (SYN^{\Delta t_1}[go^{\Delta t} l_2.\ldots]^{p}, \ldots)$$

If the timer Δt_4 representing the units of time the client is willing to wait for the $SYNACK$ ambient expires, then the client sends another SYN ambient. If $t_4 = 0$ then the rule **(R-TimePass)** is applied and the safety process is launched:

$$open^{\Delta 0} SYNACK.(ACK^{\Delta t_5}[out^{\Delta t_6} client.in^{\Delta t_7} server]^{\mu_4}, send \mid send\_ack)$$
$$\dashrightarrow send \mid send\_ack$$

At this moment the process of establishing the connection could begin again. Suppose that before the timer Δt expires the ambient SYN with $\mu_3 = a$ migrates to location l_2 by applying a **(R-Migrate)** rule:

$$l_1[[SYN^{\Delta t_1}[go^{\Delta t} l_2.in^{\Delta t_3} server]^{a}]] \dashrightarrow l_2[[SYN^{\Delta t_1}[in^{\Delta t_3} server]^{p}]]$$

Then by applying the rule **(R-In)** for $\mu_3 = a$ we obtain:

$$SYN^{\Delta t_1}[in^{\Delta t_3} server : l_4]^{a}]] \mid l_2[[server^{\infty}[\ldots]^{\mu_2} \dashrightarrow server^{\infty}[SYN^{\Delta t_1}[\ ]^{p}\ ]^{\mu_2}$$

Here the ambient SYN is dissolved and a new $SYNACK$ ambient is created. This is realized by applying a rule **(R-Open)**:

$$SYN^{\Delta t_1}[\ ]^{\mu_3} \mid open^{\Delta t_8} SYN.(SYNACK^{\Delta t_9}[\ldots]^{\mu_5}, \ldots) \dashrightarrow SYNACK^{\Delta t_9}[\ldots]^{\mu_5}$$

If the timer Δt_1 expires the client still waits for the timer Δt_4 to expire in order to send another ambient SYN. If the SYN ambient reaches the *server* ambient and an ambient $SYNACK$ is received from the server, then the *client* sends an ambient ACK. Once the server receives with success the ambient SYN it tries to sends an ambient $SYNACK$ to confirm that is agrees with the connection.

4 Timed Mobile Ambients Behaviour

In this section we provide some bisimulation relations with respect to the passage of time and locations inspired from domain name system. In process algebra two terms are said to be equivalent if they have the same behaviour in all possible contexts.

One of the most important scenario in which the services of ARP can be used is the following one: the sender is a router that has received a datagram destined for a host on another network. It checks its routing table and finds the IP address of the next router. The IP address of the next router becomes the logical address that must be mapped to a physical address.

The routing table consists of all the names of the routers from the first level, which in our cases are represented by top ambients. Also, DNS requires that each server keep a TTL counter for each mapping it caches. This two cases are treated in the following two subsections.

4.1 Location Bisimulation

Instead of comparing the behaviour of two ambients in all possible contexts, we compare the two ambients with respect to an observer placed at a given location k. That is, two ambients are equivalent with respect to an observer placed at a location k if they have the same observable behaviour at location k. We consider that an observer placed at the physical location k can only observe the top ambients from the physical location k.

Definition 2. ***i)*** *A k-barb predicate $\downarrow_{n@k}$ over ambients is defined inductively by the following system of rules:*

$$\frac{-}{k[[(n^{\Delta t}[P]^{\mu}, R)]] \downarrow_{n@k}} \quad \frac{k[[P]] \downarrow_{n@k}}{k[[P \mid Q]] \downarrow_{n@k}} \quad \frac{M \downarrow_{n@k}}{M \mid N \downarrow_{n@k}} \quad \frac{M \downarrow_{n@k} \text{ and } l \neq k}{(\nu l)M \downarrow_{n@k}}$$

ii) *A k-barbed bisimulation $\mathcal{R}$ over ambients is a symmetric binary relation over processes which for all $(M, N) \in \mathcal{R}$ implies*

1. *if $M \downarrow_{n@k}$, then $N \downarrow_{n@k}$ for any barb $\downarrow_{n@k}$;*
2. *if $M \dashrightarrow M'$, then $N \dashrightarrow N'$ and $(M', N') \in \mathcal{R}$.*

Two processes are k-barbed bisimilar over ambients with respect to a location k, denoted $M \dot{\sim}_k N$, if and only if $(M, N) \in \mathcal{R}$ for some k-barbed bisimulation over ambients $\mathcal{R}$.

Instead of considering observers placed at given physical locations, we say that two ambients are similar if they contain the same top ambients. A global observer has a global view of the system, while a local observer has a local view of the system.

Definition 3. ***i)*** *A global barb predicate $\downarrow_n$ over ambients is defined inductively by the following system of rules:*

$$\frac{-}{k[[(n^{\Delta t}[P]^{\mu}, R)]] \downarrow_{n}} \quad \frac{k[[P]] \downarrow_{n}}{k[[P \mid Q]] \downarrow_{n}} \quad \frac{M \downarrow_{n}}{M \mid N \downarrow_{n}} \quad \frac{M \downarrow_{n} \text{ and } l \neq k}{(\nu l)M \downarrow_{n}}$$

ii) *A global barbed bisimulation $\mathcal{R}$ over ambients is a symmetric binary relation over processes which for all $(M, N) \in \mathcal{R}$ implies*

1. *if $M \downarrow_n$, then $N \downarrow_n$ for any barb $\downarrow_n$;*
2. *if $M \dashrightarrow M'$, then $N \dashrightarrow N'$ and $(M', N') \in \mathcal{R}$.*

Two processes are global barbed bisimilar over ambients, denoted $M \dot{\sim} N$, if and only if $(M, N) \in \mathcal{R}$ for some global barbed bisimulation over ambients $\mathcal{R}$.

The following proposition states that if two ambients are equivalent with respect to observers placed at all locations, then they are equivalent with respect to a global observer. The reverse of the proposition it is not true because in $M \dot{\sim} N$ there is no mention of any locations, so two ambients placed at different locations can contain the same top ambients, but they can contain different top ambients with respect to observers placed at all the possible locations.

Proposition 1. *If $M \dot{\sim}_k N$ for all the locations k, then $M \dot{\sim} N$.*

Proof (Sketch). From the definition of $\dot{\sim}_k$ it results that the processes perform the same reductions and contain the same top ambients related to an observer placed at physical location k. By considering observers placed at all the possible locations k, the processes execute the same reductions and contain the same top ambients after every reduction. Because they perform the same reductions related to every location k, it means that they have the same movement through space and time. From the definition of $\dot{\sim}$ it results that $M \dot{\sim} N$.

In both local and global bisimulations, the observer is restricted to observe only top ambients. In a similar way we can replace the power to observe ambients with the power to observe capabilities. Having locations, ambients and capabilities, it is rather natural to strengthen the observing power of the observer by combining these observation possibilities.

4.2 Timed Location Bisimulation

Since we also deal with timed features, we may consider the observer able to check the value of different timers. We consider that an observer placed at the physical location k can only observe the top ambients together with their timers placed at the physical location k.

Definition 4. ***i)*** *A timed k-barb predicate $\downarrow^t_{n@k}$ over ambients is defined inductively by the following system of rules:*

$$\frac{-}{k[[(n^{\Delta t}[P]^{\mu}, R)]] \downarrow^t_{n@k}} \quad \frac{k[[P]] \downarrow^t_{n@k}}{k[[P \mid Q]] \downarrow^t_{n@k}} \quad \frac{M \downarrow^t_{n@k}}{M \mid N \downarrow^t_{n@k}} \quad \frac{M \downarrow^t_{n@k} \text{ and } l \neq k}{(\nu l)M \downarrow^t_{n@k}}$$

ii) *A timed k-barbed bisimulation $\mathcal{R}$ over ambients is a symmetric binary relation over processes which for all $(M, N) \in \mathcal{R}$ implies*

1. *if $M \downarrow^t_{n@k}$, then $N \downarrow^t_{n@k}$ for any barb $\downarrow^t_{n@k}$;*
2. *if $M \dashrightarrow M'$, then $N \dashrightarrow N'$ and $(M', N') \in \mathcal{R}$.*

Two processes are timed k-barbed bisimilar over ambients related to location k, denoted $M \overset{t}{\dot{\sim}_k} N$, if and only if $(M, N) \in \mathcal{R}$ for some timed k-barbed bisimulation over ambients $\mathcal{R}$.

Instead of considering observers placed at given physical locations, we may say that two ambients are similar if they contain the same top ambients with the same timers.

Definition 5. ***i)*** *A timed global barb predicate $\downarrow_n^t$ over ambients is defined inductively by the following system of rules:*

$$\frac{-}{k[[(n^{\Delta t}[P]^{\mu}, R)]] \downarrow_n^t} \quad \frac{k[[P]] \downarrow_n^t}{k[[P \mid Q]] \downarrow_n^t} \quad \frac{M \downarrow_n^t}{M \mid N \downarrow_n^t} \quad \frac{M \downarrow_n^t \text{ and } l \neq k}{(\nu l)M \downarrow_n^t}$$

ii) *A timed global barbed bisimulation $\mathcal{R}$ over ambients is a symmetric binary relation over processes which for all $(M, N) \in \mathcal{R}$ implies*

1. *if $M \downarrow_n^t$ then $N \downarrow_n^t$ for any barb $\downarrow_n^t$;*
2. *if $M \dashrightarrow M'$, then $N \dashrightarrow N'$ and $(M', N') \in \mathcal{R}$.*

Two processes are timed global barbed bisimilar over ambients, denoted $M \overset{t}{\dot{\sim}} N$, if and only if $(M, N) \in \mathcal{R}$ for some timed global barbed bisimulation over ambients $\mathcal{R}$.

The following proposition is similar to Proposition 1, the main difference being the fact that the observers work also with the timers of the ambients.

Proposition 2. *If $M \overset{t}{\dot{\sim}}_k N$ for all the locations k, then $M \overset{t}{\dot{\sim}} N$.*

Proof (Sketch). The same reasoning as at Proposition 1.

Proposition 3. *The timed barbed bisimulation over ambients is strictly finer than the barbed bisimulation over ambients:*

1. *$\forall M, N$, if $M \overset{t}{\dot{\sim}}_k N$ then $M \dot{\sim}_k N$*
2. *$\exists M, N$ such that $M \dot{\sim}_k N$ and $M \overset{t}{\dot{\not\sim}}_k N$.*

Proof. It is easy to see that $M \downarrow_{n@k}^t$ implies $M \downarrow_{n@k}$. If the observer can observe the same top ambients and ambient timers at location k, then the observer can observe just the top ambients at location k while ignoring the timers of the ambients. For the second part we give a counterexample.
Counterexample: let us consider the processes M, N defined as follows:

$$M = k[[n^{\Delta t_1}[P]^{\mu_1}]] \text{ and } N = k[[n^{\Delta t_2}[P]^{\mu_2}]] \text{ with } t_1 \neq t_2$$

It holds that $M \downarrow_{n@k}$ and $N \downarrow_{n@k}$, and thus $M \dot{\sim}_k Q$. Following the definition of timed barbed bisimulation over ambients, it holds that $M \downarrow_{n@k}^{t_1}$ and $N \downarrow_{n@k}^{t_2}$, and since $t_1 \neq t_2$ we have $M \overset{t}{\dot{\not\sim}}_k N$.

Similar results can be obtained between various bisimulations by considering observers with power of observing any combination of ambients, capabilities, timers over ambients and capabilities, located or not.

Example 1. Let us consider the following two mobile ambients: $M = k[[n^{\Delta 4}[\,]^{\mu} \mid out^{\Delta 1} n.(m^{\Delta 6}[P]^{\mu}, Q)]]$ and $N = k[[n^{\Delta 4}[(m^{\Delta 7}[out^{\Delta 1} n.P]^{\mu}, Q)]]^{\mu}]]$.

We have that $M \downarrow_{n@k}^4$ and also $N \downarrow_{n@k}^4$. After one reduction step we obtain: $M \dashrightarrow M'$, with $M' = k[[n^{\Delta 4}[\,]^{\mu} \mid (m^{\Delta 6}[P]^{\mu}, Q)]]$ and $N \dashrightarrow N'$, with $N' = k[[n^{\Delta 4}[\,]^{\mu} \mid (m^{\Delta 6}[P]^{\mu}, Q)]]$. We observe that $M' = N'$ so $M' \overset{t}{\dot{\sim}}_k N'$, from where it results that $M \overset{t}{\dot{\sim}}_k N$.

4.3 Properties Related to the Passage of Time

We denote by $M \overset{t}{\dashrightarrow} N$ the fact that process P evolves to process Q after applying the rule **(R-TimePass)** for $t \geq 0$ times. We denote by $\cong$ the relation which respects all the rules of Table 5 except the rule **(S-Repl Par)**.

We claim that the passage of time cannot cause a nondeterministic behaviour.

Proposition 4. *If $M \cong N$, $M \overset{t}{\dashrightarrow} M'$ and $N \overset{t}{\dashrightarrow} N'$ then $M' \cong N'$.*

Proof. The proof proceeds by structural induction, by studying all the cases from Table 5 except the rule **(S-Repl Par)**.

The following example motivates why we have removed the rule **(S-Repl Par)**. Let $P = in^{\Delta 5}n$. Then we have $k[[*P]] \Xi k[[P \mid *P]]$. By applying the time-progress function Φ_Δ, we obtain $\Phi_\Delta(k[[P \mid *P]]) = k[[in^{\Delta 4}n \mid *P]] \not\Xi k[[*P]] = \Phi_\Delta(k[[*P]])$.

We say that a process M simulates another process N if whenever N reduces, M may mimic this reduction and evolves into a new state which continues to be in the same simulation relation with the new state of N. Bisimilarity of two processes is defined by requiring that the simulation relation is symmetric, that is, each process can mimic any event of the other while remaining in the bisimulation relation with the new state of the former process. Since we have a clock, it is possible to define a bisimulation in tMA which requires processes to match their time passages.

Definition 6. *A binary relation $\mathcal{R}$ over processes is a strong simulation if whenever $(M, N) \in \mathcal{R}$, if $M \overset{t}{\dashrightarrow} M'$ then there exists N' such that $N \overset{t}{\dashrightarrow} N'$ and $(M', N') \in \mathcal{R}$. A binary relation $\mathcal{R}$ is said to be a strong bisimulation if both $\mathcal{R}$ and its converse are strong simulations. We say that M and N are strongly bisimilar, written $M \sim_t N$, if there exists a strong bisimulation $\mathcal{R}$ such that $M\mathcal{R}N$.*

Proposition 5. *If $M \sim_t N$, then $M \sim_{kt} N$ for any $k \in \mathbb{N}^*$.*

Proposition 6. *If $M \sim_t N$ and $M \sim_{t'} N$, then $M \sim_{[t,t']} N$ where by $[t, t']$ we have denoted the least common multiple.*

Proposition 7. *$\sim_t$ is an equivalence relation.*

Proof. To demonstrate that $\sim_t$ is an equivalence relation we must show that:
1. *$M \sim_t M$*
2. *if $M \sim_t N$ then $N \sim_t M$*
3. *if $M \sim_t N$ and $N \sim_t N_1$ then $M \sim_t N_1$*
1. *Obvious.*
2. *It results from the definition.*
3. *To demonstrate that $M \sim_t N_1$ we must show that if $M \overset{t}{\dashrightarrow} M'$ then there exist N_1' such that $N_1 \overset{t}{\dashrightarrow} N_1'$ with $M' \sim_t N_1'$. $M \sim_t N$ implies that if $M \overset{t}{\dashrightarrow} M'$ then there exists N' such that $N \overset{t}{\dashrightarrow} N'$ and $M' \sim_t N'$. Similarly, $N \sim_t N_1$ implies that if $N \overset{t}{\dashrightarrow} N'$ then there exists N_1' such that $N_1 \overset{t}{\dashrightarrow} N_1'$ and $N' \sim_t N_1'$. It results that if $M \overset{t}{\dashrightarrow} M'$ then there exists N_1' such that $N_1 \overset{t}{\dashrightarrow} N_1'$, and by induction and using the symmetry expressed by 2 we have that $M' \sim_t N_1'$. From Definition 6, it results that $M \sim_t N_1$.*

Definition 7. *A process context $\mathcal{C}$ is a process containing a hole, represented by $[\,]$. The elementary process contexts are given by the following syntax:*

$$\mathcal{C} ::= [\,] \mid (\nu n)\mathcal{C} \mid P \,|\, \mathcal{C} \mid \mathcal{C} \,|\, P \mid (n^{\Delta t}[\mathcal{C}]^{\mu}, Q), (n^{\Delta t}[\mathcal{P}]^{\mu}, C)$$

Let $\mathcal{C}(P)$ be the process obtained by filling the hole in $\mathcal{C}$ with a copy of P; we note that certain names free in P may become bound. We say that an equivalence relation is a *congruence* if it is preserved by all elementary contexts, namely the ones from the above definition.

Proposition 8. *$\sim_t$ is a congruence.*

Proof. *We know that $\sim_t$ is an equivalence relation (Proposition 7), and we should prove that if $k[[P]] \sim_t k[[Q]]$ then the following relations hold:*

1. $k[[(\nu n)P]] \sim_t k[[(\nu n)Q]]$
2. $k[[P \,|\, R]] \sim_t k[[Q \,|\, R]]$
3. $k[[R \,|\, P]] \sim_t k[[R \,|\, Q]]$
4. $k[[(n^{\Delta t}[P]^{\mu}, R)]] \sim_t k[[(n^{\Delta t}[Q]^{\mu}, R)]]$
5. $k[[(n^{\Delta t}[R]^{\mu'}, P)]] \sim_t k[[(n^{\Delta t}[R]^{\mu}, Q)]]$

We consider only the second relationship; the others are similar. We prove that

$$\mathcal{R} = \{(k[[P \,|\, R]], k[[Q \,|\, R]]) \mid k[[P]] \sim_t k[[Q]]\}$$

is a strong bisimulation. Let $k[[P \,|\, R]] \overset{t}{\dashrightarrow} U$. We must find V such that $k[[Q \,|\, R]] \overset{t}{\dashrightarrow} V$ and $(U, V) \in \mathcal{R}$. We have that $U = k[[P' \mid R']]$ and $V = k[[Q' \mid R']]$, where $k[[P]] \overset{t}{\dashrightarrow} k[[P']]$, $k[[Q]] \overset{t}{\dashrightarrow} k[[Q']]$ and $k[[R]] \overset{t}{\dashrightarrow} k[[R']]$. From $k[[P]] \sim_t k[[Q]]$ we have that $k[[P']] \sim_t k[[Q']]$, which means that $(U, V) \in \mathcal{R}$.

Definition 8. *A binary relation $\mathcal{R}$ over processes is a weak timed simulation if whenever $(M, N) \in \mathcal{R}$, if $M \overset{t}{\dashrightarrow} M'$ then there exists N' and $t' \geq t$ such that $N \overset{t'}{\dashrightarrow} N'$ and $(M', N') \in \mathcal{R}$. We say that M can simulate in time N, written $M \rhd_t N$, if there exists a weak timed simulation $\mathcal{R}$ such that $M\mathcal{R}N$.*

Proposition 9. *$M \sim_t N$ iff $M \rhd_t N$ and $M \rhd_t M$.*

Proof (Sketch).
If $M \sim_t N$ then it is obvious that exists $t' = t$ such that $M \rhd_t N$ and $N \rhd_t M$.
If $M \rhd_t N$ and $N \rhd_t M$ it results that there exists t' in both cases such that $t = t'$, which implies that $M \sim_t N$.

5 Conclusion

Process algebra is the general study of distributed concurrent systems in an algebraic framework. In the past few years, some successful models have been formulated within this framework: ACP [4], CCS [18], CSP [16], distributed π-calculus [14], MA [6]. None of these approaches is able to naturally describe properties of timing. Process algebra with timing features are presented in [1, 9–12, 20]. We have extended the pure mobile ambients by adding time constraints to capabilities and ambients. Two formalisms called timed π-calculus and timed distributed π-calculus are presented in [2], respectively [8]; it also uses a relative time given by timers, and a clock whose tick decreases the timers. Timers are used to restrict the interaction between components, and both types and timers are used to control the resource availability. In timed distributed π-calculus the notion of space is flat. A more realistic account of physical distribution is obtained using a hierarchical representation of space, and this is given in timed mobile ambients. Thus we get a more realistic description of the distributed computation and mobility. A natural example motivating an extension from timed distributed π-calculus to timed mobile ambients is presented in [7].

The formalism defined in this paper does not follow any of the other process algebra mentioned above. It is well motivated by the existence of timers in TCP/IP communication protocols; the timers fit very well to the description of messages as mobile ambients. Another motivation for our work is given by the Real-time Transport Protocol (or RTP) which defines a standardized packet format for delivering audio and video over the Internet. RTP can carry any data with real-time characteristics, such as interactive audio and video. It goes along with the RTCP and it is built on top of the User Datagram Protocol (UDP). Applications using RTP are less sensitive to packet loss, but typically very sensitive to delays, so UDP is a better choice than TCP for such applications. The protocols themselves do not provide mechanisms to ensure timely delivery. They also do not give any Quality of Service (QoS) guarantees. These things have to be provided by some other mechanism.

Starting from such motivations, we extend with time restrictions a formalism designed for mobility in order to study various aspects related to time. The formalism used is the basic ambient calculus, which means that we have not taken into account the primitives for communication. The novelty comes from the fact that the ambients can also expire, simulating in this way the maximum amount of time any package can exist in a network before being discarded. We have provided an operational semantics by a structural equivalence and a reduction relation. The structural relation introduced in this paper is different from the structural congruence for mobile ambients because it does not allow sibling ambients to commute their position. The reduction relation is intuitive, and we have shown in Section 4 how some of the reduction rules are used, namely **(R-In)**, **(R-Out)**, **(R-Open)**, **(R-Migrate)** and **(R-TimePass)**. To describe thes passage of time we have given a discrete time progress function. We have introduced and studied some behavioural equivalences over timed mobile ambients. After introducing the aspects of time and locations over mobile ambients, we have established some bisimulations between processes by defining different barbs.

References

1. L. Aceto, D. Murphy. Timing and Causality in Process Algebra. *Acta Informatica* vol.33(4), 317-350, 1996.
2. M. Berger. Basic Theory of Reduction Congruence for Two Timed Asynchronous pi-Calculi. *CONCUR*, Lecture Notes in Computer Science vol.3170, Springer, 115-130, 2004.

3. M. Berger. *Towards Abstractions for Distributed Systems* PhD thesis, Imperial College, Department of Computing, 2002.
4. J.A. Bergstra, J.W. Klop. Process Theory based on Bisimulation Semantics. *Linear Time, Branching Time and Partial Orders and Models For Concurrency*, Lecture Notes in Computer Science vol.354, Springer, 50-122, 1989.
5. M. Bugliesi, G. Castagna, S. Crafa. Boxed Ambients. *Theoretical Aspects of Computer Software*, Lecture Notes in Computer Science vol.2215, Springer, 38-63, 2001.
6. L. Cardelli, A. Gordon. Mobile Ambients. *Theoretical Computer Science* vol.240(1), 170-213, 2000.
7. G. Ciobanu. Interaction in time and space. Proceedings of Foundations of Interactive Computation, 45-61, to appear in *Electronic Notes in Theoretical Computer Science*, 2007.
8. G. Ciobanu, C. Prisacariu. Timers for Distributed Systems. *International Workshop on Quantitative Aspects of Programming Languages*, Electronic Notes in Theoretical Computer Science vol.164(3), 81-99, 2006.
9. R. Cleveland, A. Zwarico. A theory of testing for real-time. *Logic in Computer Science*, 110-119, 1991.
10. F. Corradini. On performance Congruences for Process Algebras. *Information and Computation* vol.145(2), 191-230, 1998.
11. F. Corradini. Absolute versus relative time in process algebras. *Information and Computation* vol.156(1), 122-172, 2000.
12. R. Gorrieri, M. Roccetti, E. Stancampiano. A Theory of Processes with Durational Actions. *Theoretical Computer Science* vol.140(1), 73-94, 1995.
13. M. Hennessy, T. Regan. A process algebra for timed systems. *Information and Computation*, vol.117, 221-239, 1995.
14. M. Hennessy, J. Riely. Resource access control in systems of mobile agents. *TInformation and Computation*, vol.173(1), 82-120, 2002.
15. D. Hirschkoff, D. Teller, P. Zimmer. Using ambients to control resources. *International Conference on Concurrency Theory*, Lecture Notes in Computer Science vol.2421, Springer, 288-303, 2002.
16. C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall International, 1985.
17. F. Levi, D. Sangiorgi. Controlling interference in ambients. *Principles of Programming Languages*, 352-364, 2000.
18. R. Milner. *Communication and Concurrency.* Prentice Hall International, 1989.
19. R. Milner. *Communicating and mobile systems: the π-calculus.* Cambridge University Press, 1999.
20. F. Moller, C. Tofts. A temporal Calculus of Communicating Systems. *International Conference on Concurrency Theory*, Lecture Notes in Computer Science vol.527, Springer, 401-415, 1991.
21. W.R. Stevens. *TCP/IP Illustrated, Volume 1 - The Protocols* Addison-Wesley, 1993.