# FORMAL SPECIFICATION AND ANALYSIS OF A CONTROL SYSTEM BASED ON COMPUTER NETWORKS

Blum, I. and Juanole, G.

LAAS-CNRS, 7 avenue du Colonel Roche, 31077 Toulouse Cedex 4. FRANCE
T. (33) 5 61 33 62 58 F. (33) 5 61 33 64 11 E-mail : {blum, juanole}@laas.fr

**Abstract -** *The control architecture for a distributed application (elevator system) is at first specified in terms of functionnalities, networks and exchanges on the networks. This architecture is based on two types of local area networks (CANs and one network with a centralized control).*
*Second, this architecture is modelled with Stochastic Timed Petri Nets which allow to express the main mechanisms of real time distributed systems and to make qualitative and quantitative analysis. Properties and performances of the elevator system are verified and evaluated.*

## 1. INTRODUCTION

Taking into account the technological evolution of the last decade, the design of distributed applications based on communication networks becomes a very important area of research activity. These systems include a lot of complex mechanisms and then it is necessary during the design phase to use formal description techniques in order to control design errors, to verify properties and to forecast dependability and operational performances. This paper is precisely concerned by the formal design of a distributed application.

As formal model, we consider Stochastic Timed Petri Nets (STPN) [1, 2, 3, 4] which allow to express, in particular, parallelism, synchronisation and time constraints (essential mechanisms in the context of a real time distributed system) and to make qualitative and quantitative analysis of the modelled system on the basis of a randomized state graph [1, 2, 3, 4] (a randomized state graph represents the dynamic behaviour of the STPN model and its transitions are labeled with the name of actions represented by the transitions, probabilities and durations).

The purpose of this paper is double: (i) showing the different steps for the specification of the control architecture, (ii) underlining the interest of STPN model to formally specify and analyse communications and distributed applications.

The distributed application which is considered is an elevator system [5] which has been studied in the context of a french working group on the Real Time.

This paper is organized in four sections. The first section presents the elevator system. The second section presents the control system in terms of architecture, control strategies and communication requirements. The third section is concerned by the formal modeling using the STPN model. The fourth section considers one scenario and presents the qualitative and quantitative analysis which has been made.

## 2. ELEVATOR SYSTEM

### 2.1. General presentation

We suppose a building with several elevator shafts at each floor. For each elevator, there is a set of elevator buttons (a passenger presses a button to select a destination; we call internal request such a pressure), a set of elevator lamps to visualize the floor where a stop has been requested (these lamps are switched on as a consequence of pressures on the elevator buttons), a visual indicator which indicates the number of the floor in front of which the elevator passes, an elevator motor which is controlled by commands to move down, move up and stop, and an elevator door which is controlled by commands to open and close the door. There is also a floor arrival sensor at each floor in each elevator shaft to detect the arrival of an elevator. Furthermore, at each floor, we have:

- for the set of the elevators, two floor buttons (up and down buttons) which, a person wanting to take an elevator, presses to indicate the direction which is requested (we call an external request such a pressure) and two floor lamps which visualize the direction which is requested.

- for each elevator, a visual indication which indicates the elevator position (floor number).

For the top floor and bottom floor, there is only one floor button and one floor lamp.

In the context of the study which has been done, the only time property which must be checked is: a controller must give the stop order, for a requested floor, 50 ms, at the more, after the detection by the

0-7803-4182-1/97/$10.00 ©1997 IEEE

sensor of the arrival to the floor. Furthermore, we only consider the normal behaviour i.e. without failures and alarms.

## 2.2. Architecture

The elevator shaft is made of four modules (figure 1): the elevator itself; the floors, the sensors and the controller. The elevator includes four sub-modules (the buttons, the lamps, the visual indicators and the elevator behaviour).

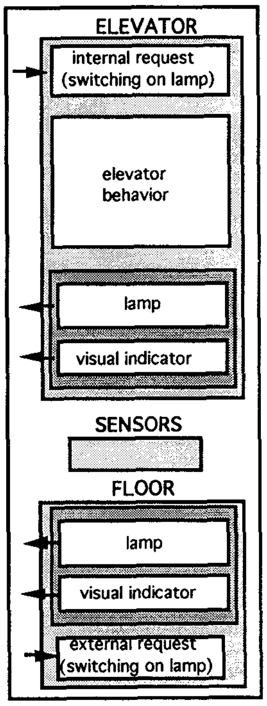


*figure 1: Elevator system*

The interactions between users and the shaft elevator are represented in figure 1. Users requests (internal or external) are represented by incoming arrows and visual informations for users are represented by outgoing arrows.

## 2.3. State machine of the elevator behavior

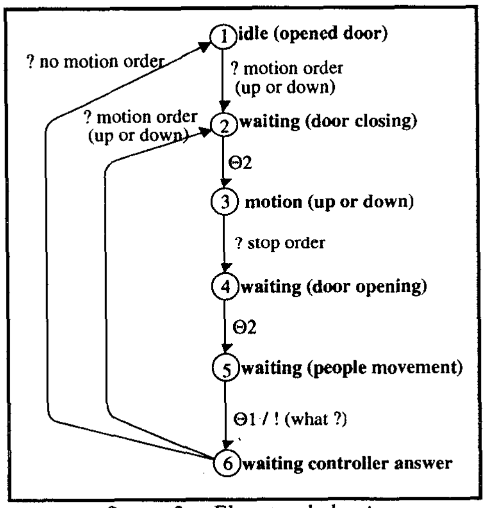


*figure 2 : Elevator behaviour*

The state machine of the elevator behaviour sub-module is represented on the figure 2 (Θ1 is the time to allow the people to go in and out; Θ2 is the time to close or to open the door).

## 3. CONTROL ARCHITECTURE

### 3.1. Control function architecture

We define (figure 3) the concepts of Manager (of the global system) and of elevator shaft controller (which controls the motion of the elevator).

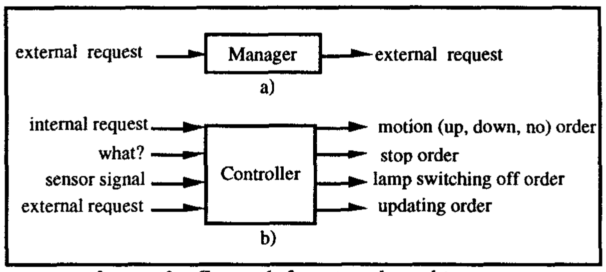


*figure 3: Control functional architecture*

The manager (figure 3.a) receives the external requests and delivers, using a strategy called global strategy (that we explain later), these external requests to the elevator shaft controllers which have been choosen. The global strategy is elaborated by questionning the elevator shaft controllers.

The elevator shaft controller (figure 3.b):

- receives the internal requests and also questions sent by the elevator about the future motion (what ?), external requests sent by the manager and the floor sensor signals.

- determines (by using a strategy called local strategy, that we explain later) the future motion of the elevator.

- and sends orders to the elevator (motion (up, down, no), stop, visual indicators updating, lamps switching off) and to the floors (visual indicators updating, lamp switching off).

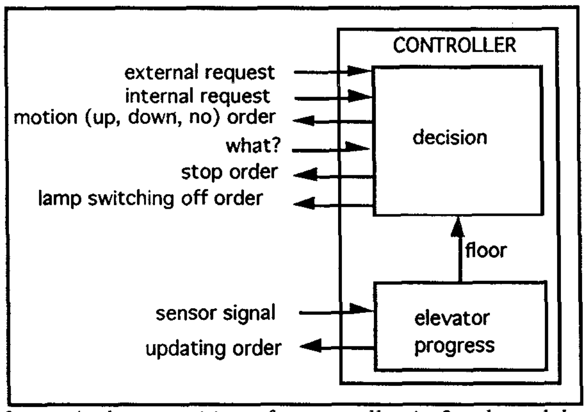


*figure 4: decomposition of a controller in 2 sub-modules*

The elevator shaft controller can be decomposed into two sub-modules (figure 4): the elevator progress which gives the information on the elevator position; the decision which elaborates the local strategy i.e. from the requests (internal, external), the elevator position and the questions (what ?) of the elevator behaviour sub-module, it gives orders to the behaviour sub-module.

The state machine of the decicion sub-module is represented on the figure 5.

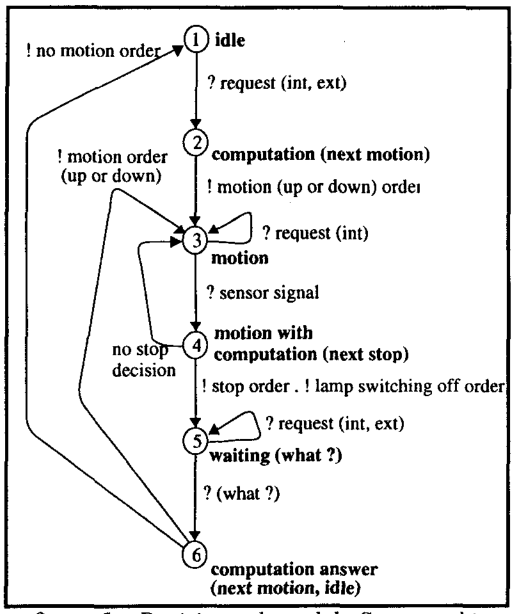

! no motion order
1 idle
? request (int, ext)
! motion order (up or down)
2 computation (next motion)
! motion (up or down) order
3 ? request (int)
motion
? sensor signal
no stop decision
4 motion with computation (next stop)
! stop order . ! lamp switching off order
5 ? request (int, ext)
waiting (what ?)
? (what ?)
6 computation answer (next motion, idle)

*figure 5 : Decision sub-module State machine*

## 3.2. Control strategies

### - Global strategy

In order to elaborate the global strategy, the manager asks the controllers for the elevator position and the last motion order they have given (up, down, no motion).

The global strategy is defined in the following way:

- an external request is sent to the controller of the elevator which goes in the direction which is wished by the request and which did not pass (during the motion in this direction) in front of the floor where this request is now,

- if several elevators satisfy this condition, the manager sends the external request, at random, to one of these elevators; if none elevators satisfy this condition, the manager sends again, at random, to one of the elevators.

Note furthemore that, if an elevator is in the state idle, an external request at this floor is not considered by the manager (as the door is opened this request is automatically served).

### - Local strategy

The main idea is to minimize the direction changes. Taking into account this main idea, we have the following cases:

1. an elevator which is going in a direction (for exemple up) maintains this direction as long as there are requests (internal and external) which want to go in this direction.

2. when an elevator stops at a floor with none request as expressed in 1, we can however have requests for going in the opposite direction; these requests can be:

a - external requests situated above the stop floor

b - external requests situated below the stop floor

c - internal requests for the floors below the stop floor.

If we have at least the case a, the elevator will serve the extreme external request (the higher). The detection of this situation is represented by a condition called "limit" which is associated to the concerned floor. When the elevator arrives at the floor where the limit is true, the elevator will cancel the condition limit and will change the direction.

If we have not the case a, but we have the cases b and / or c, the elevator will serve the closest such request.

## 3.3. Communication architecture for the global system

### - Architecture

The global system is completely distributed: the physical process (which consists in several elevators), the manager and the controllers (one for each elevator shaft). So we purpose the global system architecture which is represented on the figure 6.

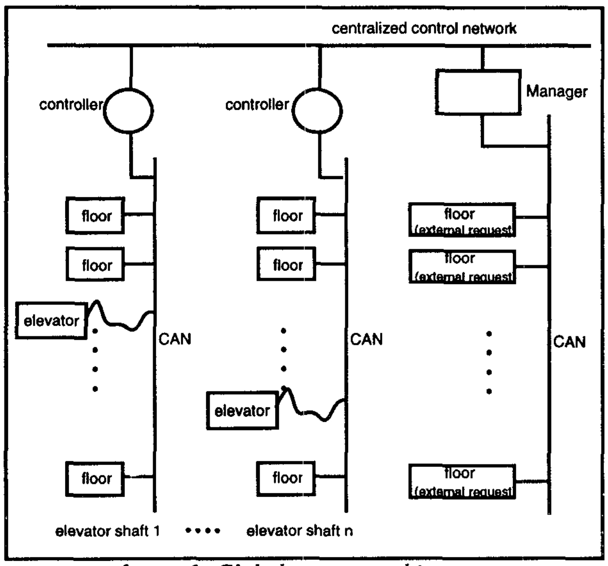


*figure 6: Global system architecture*

Taking into account these concepts and the different exchanges which have been identified, we specify the global system architecture which is represented on the figure 6:

- there is a communication network CAN associated to each elevator shaft (the exchanges of internal requests, questions (what ?), floor sensor signals and orders to the elevator and the floors); this network CAN is not concerned by the external requests;

- there is a communication network CAN which connect the floors and the manager in order to the manager collects the external requests,

- there is a computer network which connects the manager and the elevator shaft controllers and which is based on a centralized control for the medium access (the manager is the master and the controllers are the slaves); the manager (after the reception of the external requests through the communication network CAN) questions the controllers and then, after their replies, dispatches the external requests to the controllers.

CAN is choosen because it provides a medium sharing distributed and based on a priority scheme associated to the messages [6, 7, 8].

The centralized control network is choosen because the manager has the leading rôle in the activity bound to the external requests [9, 10, 11].

**- Messages exchanges specification**

We have to specify, on the one hand, the messages-priorities on the two types of CAN networks (the CAN network for each elevator shaft (we call it elevator CAN) and the CAN network to collect the external requests (we call it ext-req CAN)) and, on the other hand, the polling of the controllers by the manager on the centralized control network.

• Elevator CAN: we consider, at first, that the messages can be classified in three classes:

- the class of the messages needed by the decision module for computing decisions (decision computing message class); in this class, we have the sensors message, the (what ?) messages and internal requests.

- the class of the message representing the decision (decision message class); in this class, we have the stop order message and the motion (up, down, no) order message.

- the class of the messages for visualization aspects; we have the lamp switching off order message and the visual indicator updating order message.

We have defined the following decreasing priority scheme: priority 0 (sensors messages), priority 1 (what ?), priority 2 (internal requests), priority 3 (orders of stop or motion (up, down, no)), priority 4 (visual indication updating order), priority 5 (lamp switching off order).

• Ext-req CAN: we consider priorities which are inversely proportionnal to the floor number.

• Centralized control network: the manager polls cyclically the controllers.

## 3.4. Summary

We summarize, by only representing explicitly an elevator, the global view on the figure 7.

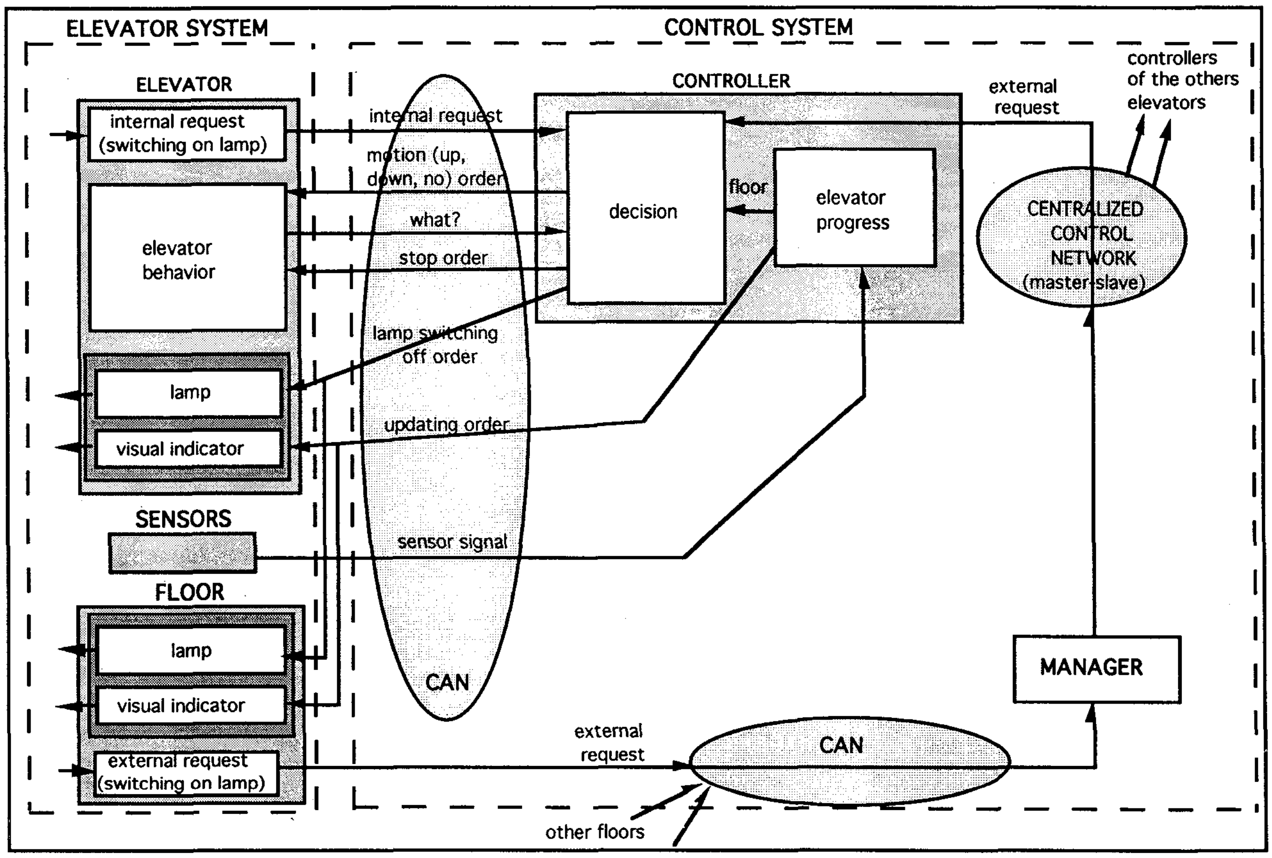


*figure 7: Global view (elevator system and control system)*

## 4. MODELLING

To analyse the elevator control system, we need the modelisation of the complete system. So we modelise all modules (presented on parts 2 and 3) by using the following methodology.

### 4.1. Methodology

The STPN model, as all the Petri net models, allows incremental modelisation:

• at first, we model the modules of an architecture (local modelling) with labeled and timed Petri nets:

- the labels associated to the transitions are of the form Predicate/Actions; a Predicate can be a local condition and/or a message reception (?message); an action can be a local action and/or a message sending (!message); a label can be Predicate/Action, Predicate/ or / Action;

- the time associated to the transitions can be either a zero time (which defines immediate transitions which are represented by a thin line), or a time different of zero (which defines timed transitions which are represented by a rectangle).

• second, we compose the models of the modules (global modelling) by considering the interactions between the modules. A module transition which has the label !X is coupled with the transition(s) of the module(s) which have the label ?X. This coupling can be made either by transition merging (that is a synchronous composition which requires time attributes identical for the concerned transitions) or by shared places (that is the asynchronous composition which must be necessarily used if the time attributes of the transitions are differents).

The STPN model includes also inhibitor arcs[12].

We now present the models of the module "Elevator Behaviour", the module "Controller Decision", the module "Elevator CAN" and the module "Sensors", which are the more interresting modules of the control elevator system.

### 4.2. Elevator behaviour

The model of the elevator behaviour is given on the figure 8. This model is directly obtained from the state machine of the figure 2 by representing each state with two places (one place for the condition of the elevator (idle, motion (up, down), waiting) and one place for the condition of the door (opened, closed)).

The places with hachures are places duplicated (for clarity reasons). The timed transitions ( which concern the closing and opening the door, and the waiting for the people movement) are represented with their deterministic distribution (δ(x-Θ2) and δ(x-Θ1)).

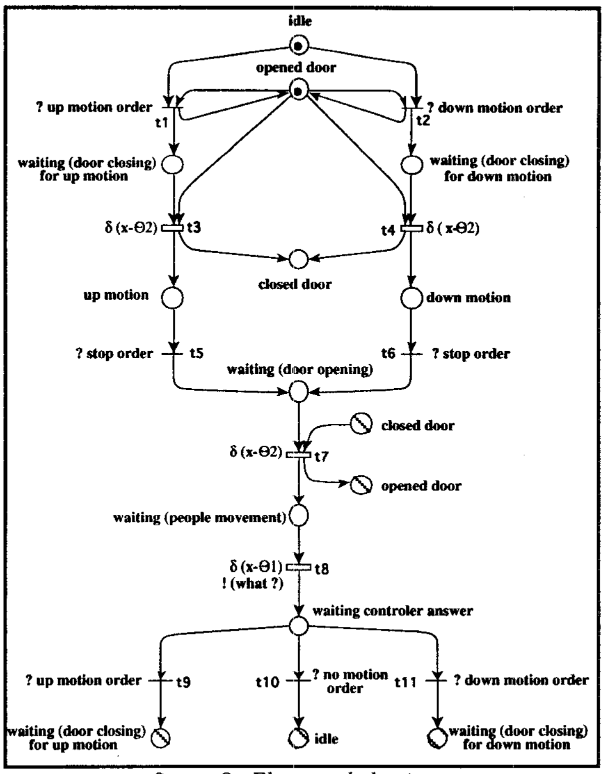


*figure 8: Elevator behaviour*

### 4.3. Controller decision

The controller decision makes computations for implementing the local strategy. We have computations made after the sensor signals reception during the motion state and during the states of no motion (when a request (int,ext) is received in the state "idle" and when the message (what ?) is received in the state "waiting"). Each computation case is represented by Petri net models.

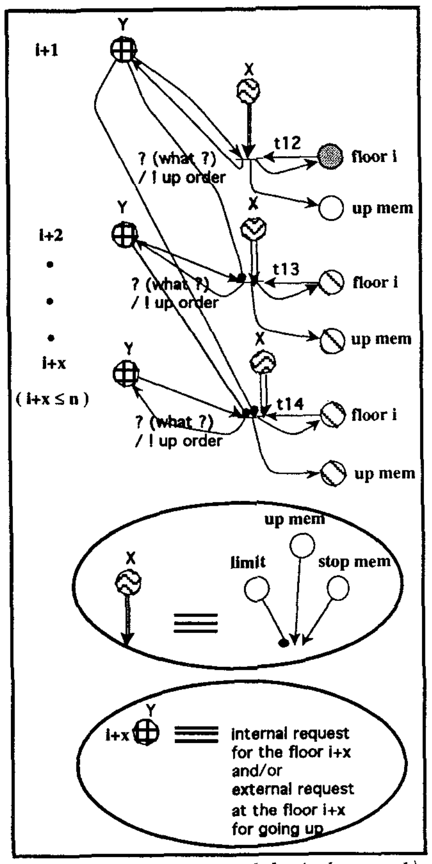
Here, we only want to give an idea of this modelling and we represent the modelling when the elevator is in the state waiting at a floor and the motion direction, in arriving at the floor, is up, and furthermore there are requests for above the floor.

Two cases must be considered whether, before receving the message (what ?), the condition limit is false (case a) or true (case b). We only model the first case. Two sub-cases must still be distinguished:

1. after receiving the message (what ?), the computation does not give the condition limit as true (that means that there are internal and / or external requests for providing services in the direction up; then these requests are served gradually).

2. after receiving the message (what ?), the computation gives the condition limit as true; then the elevator goes to serve the highest external request and then the direction will be changed (going down).

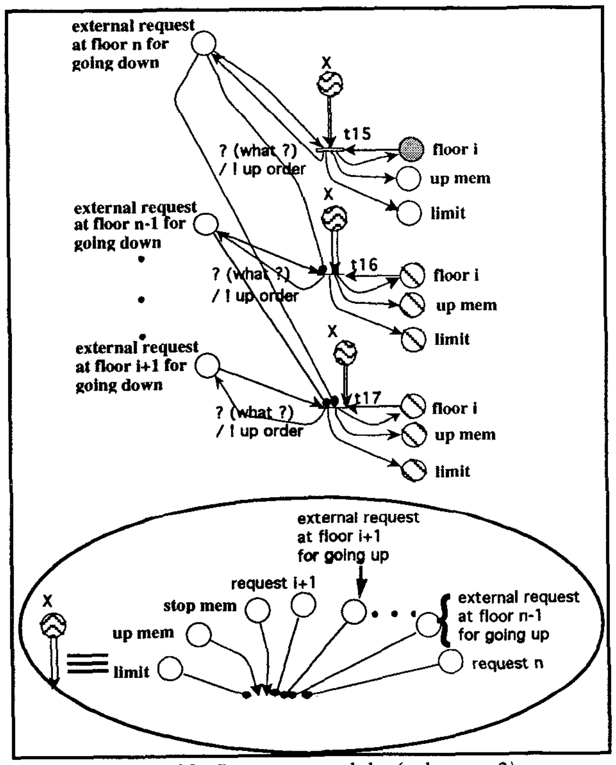
The Petri net models which represent the sub-cases 1 and 2 of the mechanisms of the decision module are given respectively on the figures 9 and 10.



i+1
i+2
i+x
( i+x ≤ n )
Y
X
t12
floor i
up mem
? (what ?) / ! up order
t13
t14
up mem
limit
stop mem
internal request for the floor i+x and/or external request at the floor i+x for going up

*figure 9: Decision module (sub-case 1)*

The figure 9 shows that, at the floor i, we go up if there is a request for the floor (i+1) or if there is no request for the floor (i+1) but there is a request for the floor (i+2) . . . and so on. The place X (the place is duplicated) and the arc coming from this place are a macro-place and a macro-arc as presented below the figure. The place Y (which is duplicated) is also a macro-place as presented below the figure (note furthermore, what is not represented here for figure clarity reasons, that the transitions t12, t13 and t14 must be duplicated by considering all the combinations of the places of the macro-place Y).



external request at floor n for going down
external request at floor n-1 for going down
external request at floor i+1 for going down
X
t15
t16
t17
floor i
up mem
limit
? (what ?) / ! up order
external request at floor i+1 for going up
request i+1
stop mem
up mem
limit
external request at floor n-1 for going up
request n

*figure 10: Decision module (sub-case 2)*

The figure 10 (the place X is a macro-place as indicated below the figure) shows that if there are no internal requests and / or external requests for going up, the elevator goes to serve the highest external request for going down.

Remark: if the motion direction on arriving at a floor is down, we have a symetrical behaviour.

## 4.4. Elevator CAN

The elevator CAN is the network which transmits the sensor signals (priority 0), the messages (what ?) (priority 1), the internal requests (priority 2), the orders of stop or motion (up, down, no) (priority 3), the visual indication updating orders (priority 4) and the lamp switching off orders (priority 5). We give its Petri net model on the figure 11 by emphasizing the internal requests exchanges (so we only represent the internal request and the messages with higher priorities).

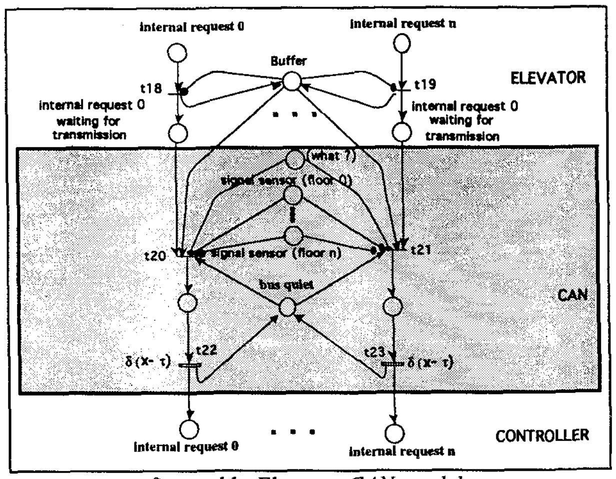


*figure 11: Elevator CAN model*

We have one buffer for receiving the requests (we have a request possible for any floor, hence internal request 0, . . ., internal request n) and then the request in this buffer is in competition with the sensor signals and the messages (what ?) for accessing the bus (transitions t20 . . . t21). The transitions t22 . . . t23 represent the transmission in the network (the deterministic distribution $\delta(x-\tau)$ is associated to these transitions, $\tau$ being the transmission time).

## 4.5. Sensors

We represent the Petri net model on the figure 12 by introducing two places which are shared with the Elevator behaviour model (figure 8): the "up motion" and "down motion" places. When the elevator is at a floor (we represent it by a place called "physical floor" in order to not make confusion with the places "floor" in the controller which represents the floor numbers), its direction is determined by the motion direction (up, down) which will induce, when it will arrive at the next floor, the sending of the sensor signal (the transitions 24, t25, t26, t27, t28 and t29 represent the transit between two floors; deterministic distributions δ(x-Θ3) are associated to these transitions, Θ3 being the transit time between two floors).

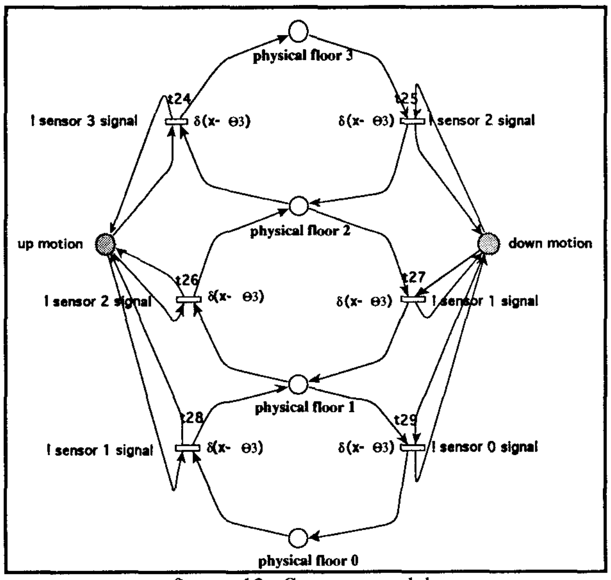


*figure 12: Sensors model*

## 5. ANALYSIS

We only consider one elevator shaft with a ground floor (floor 0) and three floors above (floors 1, 2 and 3). In this case, the manager obviously sends all the external requests to the controller of the only elevator shaft.

The purpose of the analysis is, at first, to check that the elevator shaft (taking into account for internal and external requests) provides the expected service (qualitative analysis) and, second, to evaluate the performances (quantitative analysis). So we consider a scenario (arrival and interleaving of requests) and we analyse it.

### 5.1. Scenario

In the proposed scenario, which is analysed, three important points are illustrated:

- the consideration of the priorities of the external requests (inversely proportionnal to the floor number (sub-section 3.2))
- the motion of the elevator
- the strategy (local) implemented by the controller (minimazing the direction changes (sub-section 3.2)).

The scenario is constitued by the following chronological scheme (figure 13):

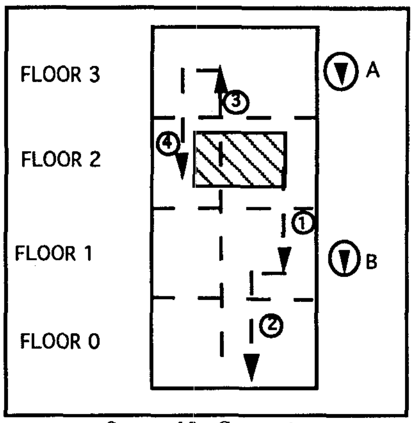


*figure 13: Scenario*

Now we detail the sequence of requests and what the system must do:

1. Initially the elevator is idle at the floor 2 and we simultaneously have two external requests, one at the floor 3 (external request A to go down) and one at the floor 1 (external request B also to go down).

2. As the ext-req CAN network gives priority to the lower floor (i. e. to the external request B), the elevator must go down and stop at floor 1 (after the reception of the stop order message sent by the controller as soon as it has received the floor 1 sensor signal); Note furthermore than after receiving the floor 1 sensor signal, the controller is aware of the external request A (which was pending) and will receive later but before the message (what?) the internal request of the custumer (we suppose that the customer wants to go to the ground floor (floor 0)).

3. The controller applies the local strategy i.e. it must go down (as there is the internal request for going to the floor 0).

4. When the elevator is at the floor 0, it (obviously) changes its direction, sets the condition "limit" to true (there only is the external request A wanting to go down) and must go up to the floor 3 to serve the request A.

5. At the floor 3, the custumer expresses the wish to go to the floor 2 (internal request for the floor 2).

6. The elevator must go down and stops at floor 2 (the custumer goes out). There is no more service which is requested and then the elevator must be idle at floor 2.

In order to analyse this scenario, we have considered the following numerical values: Θ1=2,5 s; Θ2=0 s (the time for opening and closing the door is neglected); Θ3=4,5 s; τ=560 μs. Note that in the centralized control network (we have not represented here the model), the transmission time is of 200 μs. Considering these values, the randomized state graph, which has been obtained, has 164 states and no deadlock.

## 5.2. Qualitative analysis

We make a projection on relevant events of the randomized state graph (relevant from the point of view of the service provided by the elevator): the requests (external and internal), the sensor signals and the controller orders. We get the abstract view represented on the figure 14. We have at first the interleaving of the two external requests (states 1, 2, 3, 21). But this interleaving occurs in zero time and then the two requests are in competition, at the same moment, on the elevator CAN network which gives priority to the request B: hence the decision to go down (transition 3 -> 4).

We can verify on the figure 14 that the elevator system provides the sequence of the services which are expected (and explained in part 5.1), taking into accound the priority scheme on CAN and the local strategy.

## 5.3. Quantitative analysis

• The time between the stop sending order and the sending of the sensor signal is 560 μs. So we verify that the constraint defined by the french working group on the Real Time (this time must be inferior to 50 ms) is satisfied.

• We have also evaluated the request service times:

- external request B : ≈ 4,5 s
- internal request (after B) : ≈ 5,5 s
- external request A : ≈ 27,5 s
- internal request (after A) : ≈ 5,5 s.

## 6. CONCLUSION

The study presented in this paper wants to show the different steps of the formal design of a control system for a distributed application. Such a design requires at first to specify the main modules of the architectures, the networks which are necessary and the exchanges on these networks. The choice of CANs (event driven networks) is arbitrary but we have made it because such networks process data in real time. But note however time triggered networks can also provide real time services if they have adequate cycles.

The second point that we want to underline is the interest of the Stochastic Timed Petri Nets. This model permits to modelise time constraints and dynamic aspects. It can be seen as a complementary technique with respect to SA-RT [13, 14] or HOOD/PNO [15, 16] which modelise static aspects during the specification process. More, this model allows to verify properties and to evaluate performances.

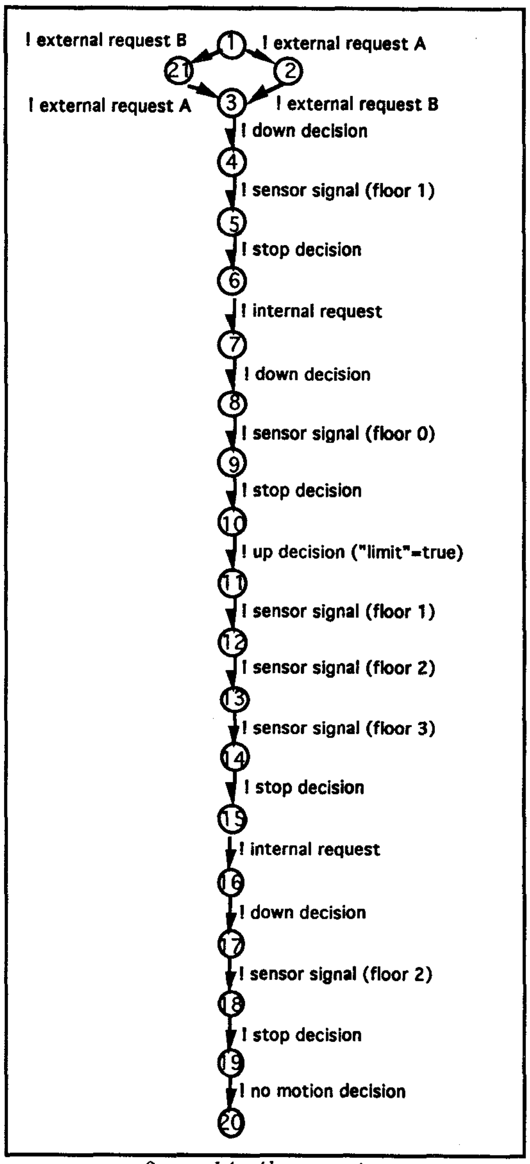


*figure 14: Abstract view*

## ACKNOWLEDGMENT

The autors would like to thank the reviewers for their helpful suggestions which allowed to improve this paper.

## REFERENCES

[1] Y. Atamna, Réseaux de Petri Temporisés Stochastiques Classiques et Bien Formés: Définition, Analyse et Application aux Systèmes Distribués Temps Réel. PhD thesis, Université Paul Sabatier de Toulouse (France), october 1994.

[2] G. Juanole and Y. Atamna, " Dealing with Arbitrary Time Distributions with the Stochastic Timed Petri Net Model. Application to Queueing Systems". In PNPM'91, the Fourth International Workshop on Petri Nets and Performance Models, pp. 32-43, Melbourne, Australia, December 1991.

[3] G. Juanole and L. Gallon, "Critical Time Distributed Systems: Qualitative and Quantitative Analysis based on Stochastic Timed Petri Nets". In FORTE'95, the 8th International IFIP Conference on Formal Description Techniques for Distributed Systems and Communications Protocols, Montreal, Canada, october 1995.

[4] G. Juanole and L. Gallon, "Formal Modeling and Analisys of a Critical Time Communication Protocol". In WFCS'95, IEEE International Workshop on Factory Communication Systems, Lausanne, Switzerland, october 1995.

[5] H. Gomaa, Software Design Methods for Concurent and Real-Time Systems. Addison-Wesley, 1993.

[6] Controller Area Network (CAN), an In-Vehicle Serial Communication Protocol-SAE J1583; (report of the Vehicule Network for Multiplexing and Data Communication Standards Comitee); SAE Handbook, vol. II, 1992.

[7] K.W. Tindell, H. Hansson, A.J. Wellings, Analysing Real-Time Communications: Controller Area Network (CAN). In Proceedings Real-Time Systems Symposium. San Juan, Puerto Rico, december 1994.

[8] Road Vehicules - Interchange of Digital Information- Controller Area Network (CAN) for High Speed Communication. ISO:DIS 11898. Febury 1992.

[9] A. Tanebaum. Computer Networks. Prentice Hall, 1983.

[10] W. Bux, K. Kuemmerle and H. Keemmerle and H. Lihth Truong, "Balanced HDLC procedures: A performance analysis". IEEE Trans. Commun., Vol COM-28, pp. 1889-1898, Nov. 1980

[11] HDLC - Proposed balanced class of procedures. Doc. ISO/TC 97/5C6-1444.

[12] G.W. Brams, Réseaux de Petri (tomes 1 et 2). Masson,1983.

[13] D.J. Hatley and I.A. Pirbhai, Strategies for Real-Time System Specification. Dorset House Publishing, 1987.

[14] P.T. Ward and S.J. Mellor, Structured Development for Real-Time Systems. Prentice Hall,1985.

[15] M. Paludetto and S. Raymond, "A methodology based on objects and Petri nets for development of real time software". IEEE Systems Man and Cybernetics, Le Touquet, France, october 1993.

[16] M. Paludetto, "Sur la commande de procédés industriels: une méthodologie basée objets et réseaux de Petri". . PhD thesis, Université Paul Sabatier de Toulouse (France), december 1991.