

LABELED RECONFIGURABLE NETS FOR MODELING CODE MOBILITY

Laïd Kahloul* and Allaoua Chaoui**

*Computer science department, Biskra University, Algeria
kahloul2006@yahoo.fr

**Lire laboratory, Constantine University, Algeria
a_chaoui2001@yahoo.com

ABSTRACT

Code mobility technologies attract more and more developers and consumers. Numerous domains are concerned, many platforms are developed and interest applications are realized. However, developing good software products requires modeling, analyzing and proving steps. The choice of models and modeling languages is so critical on these steps. Formal tools are powerful in analyzing and proving steps. However, poorness of classical modeling language to model mobility requires proposition of new models. The objective of this paper is to provide a specific formalism “labeled reconfigurable nets” and to show how this one seems to be adequate to model different kinds of code mobility.

Keywords: *code mobility, modeling mobility, labeled reconfigurable nets, mobile agent.*

1. INTRODUCTION

Nowadays, code mobility is one of the attracting fields for computer science researchers. Code mobility technology seems an interest solution for distributed applications facing bandwidth problems, users' mobility, and fault tolerance requirement. Numerous platforms were been developed [17]. Such platforms allow the broadcasting of this technology in many domains (information retrieving [9], e-commerce [11], network management [22], ...). Software engineering researches have provided some interest design paradigms influencing the development of the field. The most recognized paradigms [7] are: code on demand, remote evaluation, and mobile agent. To avoid ad-hoc development for code mobility software, many works attempt to propose methodologies and approaches ([16], [21], [14], ...). Indeed, these approaches are mostly informal. They lack in analyzing and proving system proprieties. Enhancing development process with formal tools was an attractive field in code mobility researches.

Traditional formal tools witch were massively used to model and analyze classical systems seem

to be poor to deal with inherent proprieties in code mobility systems. Works on formal tools attempt to extended classical tools to deal with code mobility proprieties. The most important proposition can be found in process algebra based model and state transition model. For the first one, π -calculus [13] is the famous one, and for the second, high-level Petri net (with many kinds) can be considered the good representative. π -calculus is an extension for CCS (communicating concurrent systems) [12]. CCS allows modeling a system composed of a set of communicating process. This communication uses names (gates) to insure synchronization between processes. In π -calculus information can be exchanged through gates. The key idea is that this information can be also a gate. With this idea, process can exchange gates. Once these gates received, they can be used by the receiver to communicate. In an extension of π -calculus, HO π -calculus [15], processes can exchange other processes through gates (the exchanged processes called agents).

To model mobility with Petri nets, high level PNets were proposed. The most famous are Mobile Nets (variant of coloured Petri nets) [1] and Dynamic Petri nets. In mobile Petri nets, names of places can appear as tokens inside other places. Dynamic Petri nets extend mobile Petri nets. In this last one, firing a transition can cause the creation of a new subnet. With high-level Petri nets, mobility in a system is modeled through the dynamic structure of the net. A process appearing in a new environment is modeled through a new subnet created in the former net by firing a transition. Many extensions have been proposed to adapt mobile Petri net to specific mobile systems: Elementary Object Nets [18], reconfigurable nets [3], Nested Petri Nets [10], HyperPetriNets [2], ... With respect to [20], all these formalisms lack in security aspect specification. To handle this aspect in code mobility, recently Mobile Synchronous Petri Net (based on labeled coloured Petri net) are proposed [19].

The objective of this work is to present a new formalism based on Petri nets. Our formalism “labeled reconfigurable nets” with a different

semantic from the one presented in [3] is dedicated to model code mobility systems. We attempt to propose to model mobility in an intuitive and an explicit way. Mobility of code (a process or an agent) will be directly modeled through *reconfiguration* of the net. We allow adding and deleting of places, arcs, and transitions at run time.

The rest of this paper is organized as follows. In section 2, we present some related works. Section 3 starts by presenting the definition of the model. In section 4 we show how labeled reconfigurable nets can be used to model the three mobile code paradigms: “remote evaluation”, “code on demand”, and “mobile agent”. We conclude this work and give some perspectives, in section 5.

2. RELATED WORKS

In [4], the authors proposed PrN (Predicate/Transition nets) to model mobility. They use concepts: agent space which is composed of a mobility environment and a set of connector nets that bind mobile agents to mobility environment. Agents are modeled through tokens. So these agents are transferred by transition firing from a mobility environment to another. The structure of the net is not changed and mobility is modeled implicitly through the dynamic of the net. In [19], authors proposed MSPN (Mobile synchronous Petri net) as formalism to model mobile systems and security aspects. They introduced notions of nets (an entity) and disjoint locations to explicit mobility. A system is composed of set of localities that can contain nets. To explicit mobility, specific transitions (called autonomous) are introduced. Two kinds of autonomous transition were proposed: *new* and *go*. Firing a *go* transition move the net from its locality towards another locality. The destination locality is given through a token in an input place of the *go* transition. Mobile Petri nets (MPN) [1] extended colored Petri nets to model mobility. MPN is based on π -calculus and join calculus. Mobility is modeled implicitly, by considering names of places as tokens. A transition can consumes some names (places) and produce other names. The idea is inherited from π -calculus where names (gates) are exchanged between communicating process. MPN are extended to Dynamic Petri Net (DPN) [1]. In DPN, mobility is modeled explicitly, by adding subnets when transitions are fired. In their presentation [1], no explicit graphic representation has been exposed.




In nest nets [8], tokens can be Petri nets themselves. This model allows some transition when they are fired to create new nets in the output places. Nest nets can be viewed as hierarchic nets where we have different levels of details. Places can contain nets that their places can also contain other nets et cetera. So all nets created when a transition is fired are contained in a place. So the created nets are not in the same level with the first

net. This formalism is proposed to adaptive workflow systems.

In [3], authors studied equivalence between the join calculus [6] (a simple version of π -calculus) and different kinds of high level nets. They used “reconfigurable net” concept with a different semantic from the formalism presented in this work. In reconfigurable nets, the structure of the net is not explicitly changed. No places or transitions are added in runtime. The key difference with colored Petri nets is that firing transition can change names of output places. Names of places can figure as weight of output arcs. This formalism is proposed to model nets with fixed components but where connectivity can be changed over time.

In this work, we attempt to provide a formal and graphical model for code mobility. We have extended Petri net with reconfigure labeled transitions that when they are fired reconfigure the net. Mobility is modeled explicitly by the possibility of adding or deleting at runtime arcs, transitions and places. Modification in *reconfigure transition*'s label allows modeling different kinds of code mobility. Bindings to resources can be modeled by adding arcs between environments. It is clear that in this model created nets are in the same level of nets that create them. Creator and created nets can communicate. This model is more adequate for modeling mobile code systems.

3. DEFINITION OF LABELED RECONFIGURABLE NETS

Labeled reconfigurable nets are an extension of Petri nets. Informally, a labeled reconfigurable net is a set of *environments* (blocs of *units*). Connections between these environments and their contents can be modified during runtime. A *unit* is a specific Petri net. A unit can contain three kinds of transitions (a unique *start* transition: , a set of ordinary transitions: , and a set of reconfigure transitions: ).

Preconditions and post-conditions to fire a start or an ordinary transition are the same that in Petri nets. Reconfigure transitions are labeled with labels that influence their firing. When a *reconfigure transition* is fired, a net N will be (re)moved from an environment E towards another environment E'. The net N, the environment E and E' are defined in the label associated to the transition. After firing a *reconfigure transition*, the structure of the labeled reconfigurable net will be updated (i.e some places, arcs, and transitions will be deleted or added). Here after we give our formal definitions of the concepts: unit, environment and labeled reconfigurable net. After the definition, we present the dynamic aspect of this model.

Formal Definition:

Let N_1, N_2, \dots, N_k be a set of nets.

for each $i: 1, \dots, n : N_i = (P_i, T_i, A_i)$, such that :

1. $P_i = \{p_1^i, p_2^i, \dots, p_n^i\}$ a finite set of places,
2. $T_i = ST_i \cup RT_i$
 - $ST_i = \{st_1^i, st_2^i, \dots, st_m^i\}$ a finite set of standard (ordinary) transitions,
 - $RT_i = \{rt_1^i, rt_2^i, \dots, rt_r^i\}$ a finite set (eventually empty) of “reconfigure transitions”,
3. $A_i \subseteq P_i \times T_i \cup T_i \times P_i$.

Definition 1 (Unit): a unit UN is a net N_i that has a specific transition st_j^i denoted *startⁱ*. So $T_i = \{start^i\} \cup ST_i \cup RT_i$.

Definition 2 (Environment): an environment E is a quadruplet $E = (GP, RP, U, A)$

- $GP = \{gp_1, gp_2, \dots, gp_s\}$ a finite set of specific places : “guest places”;
- $RP = \{rp_1, rp_2, \dots, rp_s\}$ a finite set of specific places : “resource places”;
- $U = \{N_1, N_2, \dots, N_k\}$ a set of nets.
- $A \subseteq GP \times StrT \cup RP \times T$. Such that :
 $StrT = \{start^1, start^2, \dots, start^k\}$ and $T = ST_1 \cup RT_1 \cup ST_2 \cup RT_2 \cup \dots \cup ST_k \cup RT_k$

Definition 3 (Labeled reconfigurable net):

A labeled reconfigurable net LRN is a set of environments. $LRN = \{E_1, E_2, \dots, E_p\}$ such that

- There exist at least one net N_i in LRN such that $RT_i \neq \emptyset$;
- For each $rt_j^i \in RT_i$, rt_j^i has a label $\langle N, E_e, E_g, \psi, \beta \rangle$, such that N is a unit, E_e and E_g are environments, ψ a set of places, β a set of arcs.

Dynamic of labeled reconfigurable nets:

Let $LRN = \{E_1, E_2, \dots, E_p\}$ be a labeled reconfigurable net,

Let $E_i = (GP^i, RP^i, U^i, A^i)$ be an environment in LRN,

- $GP^i = \{gp_1^i, gp_2^i, \dots, gp_s^i\}$;
- $RP^i = \{rp_1^i, rp_2^i, \dots, rp_p^i\}$;
- $U^i = \{N_1^i, N_2^i, \dots, N_k^i\}$;
- $A^i \subseteq GP^i \times starts^i \cup RP^i \times T^i \cup T^i \times RP^i$, where:
 $Sarts^i = \{start^1, start^2, \dots, start^k\}$ and
 $T^i = \{ST_1^i, ST_2^i, \dots, ST_k^i\} \cup \{RT_1^i, RT_2^i, \dots, RT_k^i\}$

Let RT_j^i be the non empty set of reconfigure transitions associated with the net N_j^i .

$$RT_j^i = \{rt_1^j, rt_2^j, \dots, rt_r^j\}.$$

Let $rt_m^i \langle N, E_e, E_g, \psi, \beta \rangle$ be a reconfigure transition in RT_j^i , such that :

- $E_e = (GP^e, RP^e, U^e, A^e)$;
- $N = (P, T, A)$ and $N \in U^e$;
- $E_g = (GP^g, RP^g, U^g, A^g)$;
- $\psi \subseteq RP^e$; $\psi = \psi_r \cup \psi_c$. (ψ_r denotes removed places and ψ_c denotes cloned places).
- β is a set of arcs. $\beta \subseteq RP^e \times T \cup RP^g \times T$.

Let *strt* be the start transition of N.

Conditions to fire $rt_m^i \langle N, E_e, E_g, \psi, \beta \rangle$:

In addition to the known conditions, we impose that there exists a *free* place p_g in GP^g ; witch means: for each $t \in starts^g$, $(p_g, t) \notin A^g$.

After firing rt_m^i :

In addition to the known post-condition of a transition firing, we add the following post-condition:

LRN will be structurally changed such that:

If E_e and E_g denote the same environment then LRN will be not changed;

Else:

- 1) $U^g \leftarrow U^g \cup \{N\}$; $U^e \leftarrow U^e / \{N\}$;
- 2) $A^g \leftarrow A^g \cup (p_g, strt)$;
- 3) Let $DA = \{(a, b) \in A^e / (a \notin \psi \text{ and } b \notin \psi) \text{ and } ((a \in N \text{ and } b \notin N) \text{ or } (a \notin N \text{ and } b \in N))\}$, $A^e = A^e - DA$. DA –deleted arcs- to be deleted after moving N.
- 4) $RP^g \leftarrow RP^g \cup \psi$; $RP^e \leftarrow RP^e / \psi_r$
- 5) if A_{LRN} is the set of arcs in LRN,
 $A_{LRN} \leftarrow A_{LRN} \cup \beta$.

4. MODELING MOBILITY PARADIGMS WITH LABELED RECONFIGURABLE NETS

A mobile code system is composed of execution units (EUs), resources, and computational environments (CEs). EUs will be modeled as units and computational environments as environments. Modeling resources requires using a set of places.

Reconfigure transitions model mobility actions. The key in modeling mobility is to identify the label associated with the *reconfigure transition*. We must identify the unit to be moved, the target computational environment and the types of binding to resources and their locations. This label depends on the kind of mobility.

In general, a reconfigure transition *rt* is always labeled $\langle EU, CE, CE', \psi, \beta \rangle$, such that:

- EU: the execution unit to be moved.
- CE, CE': respectively, resource and target computational environments.
- ψ : will be used to model transferable resources. So ψ is empty if the system has no transferable resource.
- β : models bindings after moving.

The execution unit that contains *rt* and the EU that represents the first argument in the label will be defined according to the three design paradigms: remote REV) evaluation, code on demand (COD), and mobile agent (MA).

4.1 REMOTE EVALUATION

In remote evaluation paradigm, an execution unit EU_1 sends another execution unit EU_2 from a computational environment CE_1 to another one CE_2 . The reconfigure transition *rt* is contained in

the unit modeling EU_1 , and EU_2 will be the first argument in rt 's label.

Example 4.1: Let us consider two computational environments E_1 and E_2 . Firstly, E_1 contains two execution units EU_1 and EU_2 ; E_2 contains an execution unit EU_3 . The three execution units execute infinite loops. EU_1 executes actions $\{a_{11}, a_{12}\}$, EU_2 executes actions $\{a_{21}, a_{22}, a_{23}\}$, and EU_3 executes actions $\{a_{31}, a_{32}\}$. a_{21} requires a transferable resource TR_1 and a non-transferable resource bound by type PNR_1 which is shared with a_{11} . a_{22} and a_{12} share a transferable resource bound by value VTR_1 , and a_{23} requires a non-transferable resource NR_1 . In E_2 , EU_1 requires a non-transferable resource bound by type PNR_2 to execute a_{31} . PNR_2 has the same type of PNR_1 .

The system will be modeled as a labeled reconfigurable net LRN. LRN contains two environments E_1, E_2 that model the two computational environments (CE_1 and CE_2). Units EU_1 and EU_2 will model execution units EU_1 and EU_2 , respectively. In this case, the unit EU_1 will contain a *reconfigure transition* $rt_{\langle EU_2, E_1, E_2, \psi, \beta \rangle}$ such that:

1. $E_1 = (RP_1, GP_1, U_1, A_1)$; $RP_1 = \{TR_1, PNR_1, VTR_1, NR_1\}$. $U_1 = \{EU_1, EU_2\}$;
2. $E_2 = (RP_2, GP_2, U_2, A_2)$; $RP_2 = \{PNR_2\}$. $GP_2 = \{PEU_1\}$.
3. $\psi = \{TR_1\}$, $\psi_c = \{VTR_1\}$;
4. $\beta = \{(PEU_1, str_2), (PNR_2, a_{21}), (NR_1, a_{23})\}$.

Figure 1 shows the model this system.

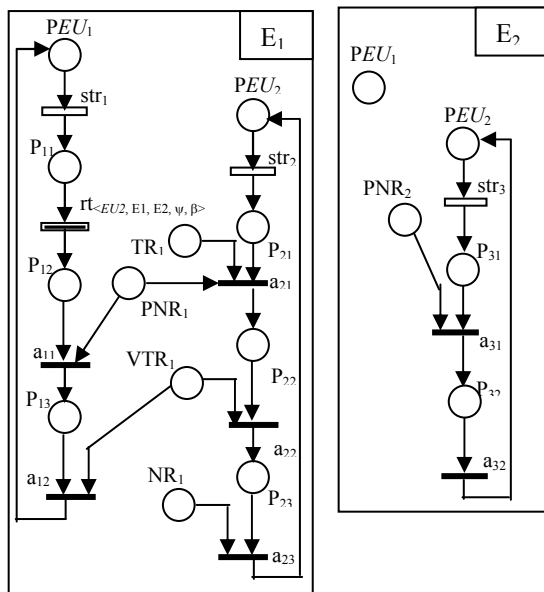


Figure 1: REV-model

The figure 2 shows the configuration after firing rt .

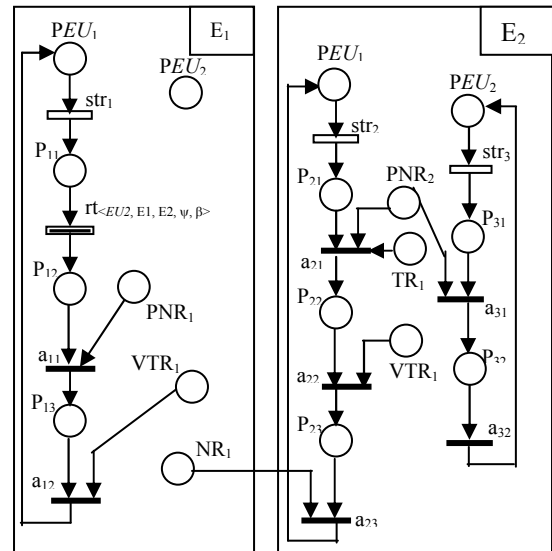


Figure 2: REV-Model after firing rt

4.2 CODE ON DEMAND

In code-on-demand paradigm, an execution unit EU_1 fetches another execution unit EU_2 . The reconfigure transition rt is contained in the unit modeling EU_1 , and EU_2 will be the first argument in rt 's label. If we reconsider the above example, the unit EU_1 will contain a reconfigure transition $rt_{\langle EU_2, E_2, E_1, \psi, \beta \rangle}$. Figure 3 shows the model proposed to model this system.

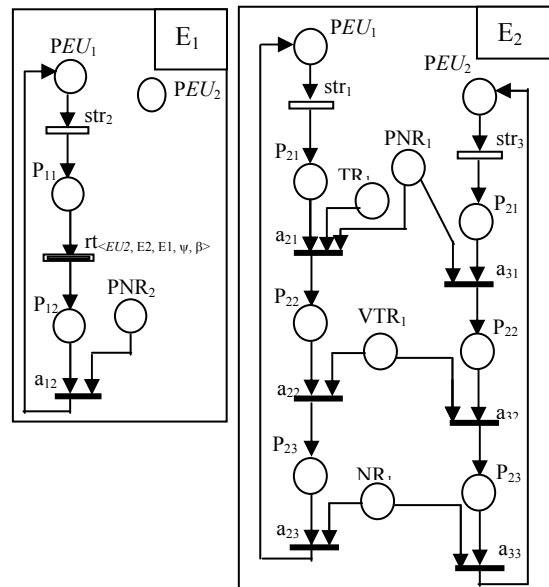


Figure 3: COD-model

The transition $rt_{\langle EU_2, E_2, E_1, \psi, \beta \rangle}$ means that EU_1 will demand EU_2 to be moved (from E_2 to E_1). In this case, $\psi = \{TR_1, VTR_1\}$, $\beta = \{(PEU_2, str_2), (PNR_2, a_{21}), (NR_1, a_{23})\}$. Figure 4 shows the configuration after firing rt .

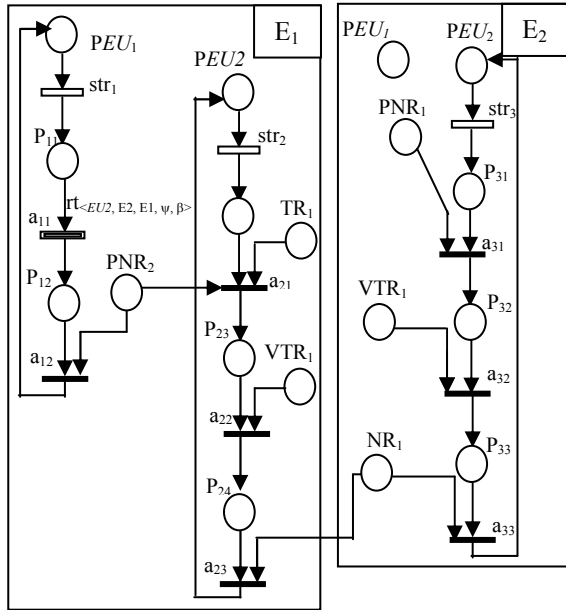


Figure 4: COD-Model after firing rt

4.3 MOBILE AGENT

In mobile agent paradigm, execution units are autonomous agents. The agent itself triggers mobility. In this case, rt –the *reconfigure transition*– is contained in the unit modeling the agent and EU (the first argument) is also this agent.

Example 4.2: let E_1 and E_2 two computational environments. E_1 contains two agents, a mobile agent MA and a static agent SA₁; E_2 contains a unique static agent SA₂. The three agents execute infinite loops. MA executes actions $\{a_{11}, a_{12}, a_{13}\}$, SA₁ executes actions $\{a_{21}, a_{22}, a_{23}\}$, and SA₂ executes actions $\{a_{33}, a_{32}\}$. To be executed, a_{11} require a transferable resource TR₁ and a non-transferable resource bound by type PNR₁ witch is shared with a_{21} . a_{12} and a_{22} share a transferable resource bound by value, and a_{13} and a_{23} share a non-transferable resource NR₁. In E_1 , SA₂ requires a non-transferable resource bound by type PNR₂ to execute a_{32} . PNR₂ has the same type of PNR₁.

The system will be modeled as a labeled reconfigurable net LRN. LRN contains two environments E_1, E_2 that model the two computational environments. In this case the unit A that models the mobile agent A will contain a reconfigure transition $rt \langle A, E_1, E_2, \psi, \beta \rangle$; such that:

1. $E_1 = (RP_1, GP_1, U_1, A_1)$; RP_1 contains at least four places that model the four resources. Let TR₁, NR₁, PNR₁ and VTR₁ be these places. GP_1 contains at least a free place PA_1 modeling that A can be received, and $U_1 = \{A\}$.
2. $E_2 = (RP_2, GP_2, U_2, A_2)$; $RP_2 = \{PNR_2\}$, $GP_2 = \{PA_2\}$.
3. $\psi_f = \{TR_1\}$, $\psi_c = \{VTR_1\}$;
4. $\beta = \{(PA_2, str_1), (PNR_2, a_{11}), (NR_1, a_{13})\}$.

Figure 5 shows the model of this system.

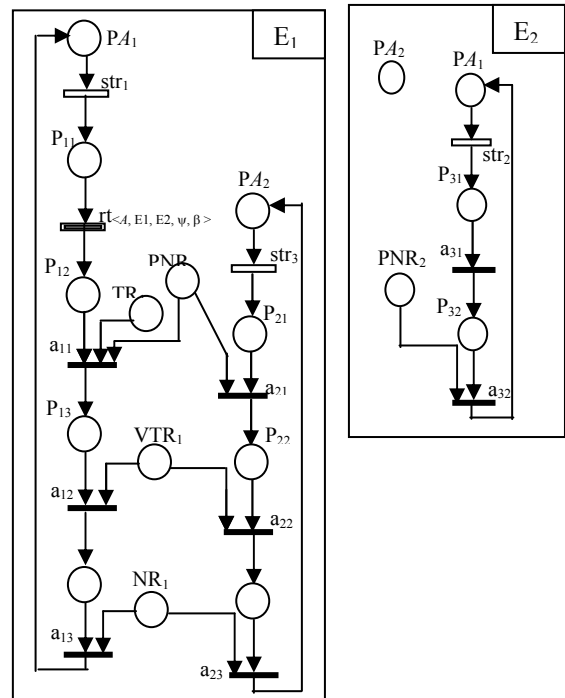


Figure 5: MA-model

The figure 6 shows the configuration after firing rt .

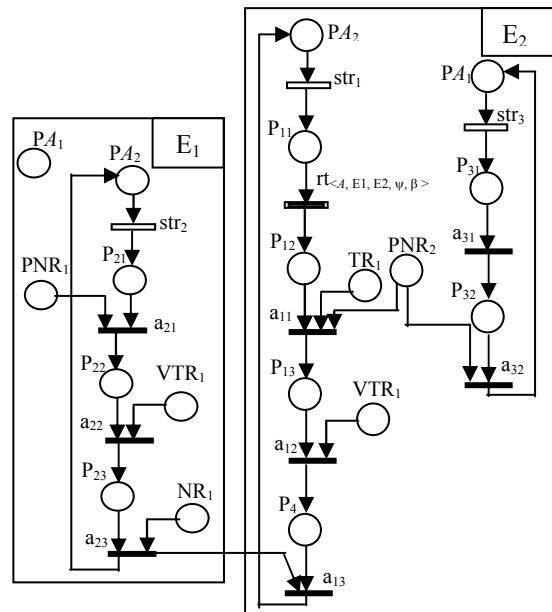


Figure 6: MA-Model after firing rt

5. CONCLUSION

Proposed initially to model concurrency and distributed systems, Petri nets attract searchers in mobility modeling domain. The ordinary formalism is so simple with a smart formal background, but it fails in modeling mobility aspects. Many extensions were been proposed to treat mobility aspects. The

key idea was to introduce mechanisms that allow reconfiguration of the model during runtime. The most works extends coloured Petri nets and borrow π -calculus or join calculus ideas to model mobility. The exchanging of names between processes in π -calculus is interpreted as exchanging of place's names when some transitions are fired. This can model dynamic communication channels. In much formalism, mobility of process is modeled by a net playing as token that moves when a transition is fired. All these mechanisms allow modeling mobility in an implicit way. We consider that the most adequate formalisms must model mobility explicitly. If a process is modeled as a subnet, mobility of this process must be modeled as a reconfiguration in the net that represents the environment of this process.

In this paper, we have presented a new formalism "*labeled reconfigurable nets*". This formalism allows explicit modeling of computational environments and processes mobility between them. We have presented how this formalism allows, in a simple and an intuitive approach, modeling mobile code paradigms. We have focused on bindings to resources and how they will be updated after mobility. We believe that the present formalism is an adequate model for all kinds of code mobility systems. In our future works we plan to focus on modeling and analyzing aspects. In modeling aspects, we are interested to handle problems such that modeling multi-hops mobility, process's states during travel, birth places and locations. On the analysis aspect, we are thinking about an encoding of our model in maude or mobile maude [5] in order an analysis automation of our models.

REFERENCES

- [1] Andrea Asperti and Nadia Busi. "*Mobile Petri Nets*". Technical Report UBLCS-96-10, Department of Computer Science University of Bologna, May 1996.
- [2] M.A. Bednarczyk, L. Bernardinello, W. Pawlowski, and L. Pomello. "*Modelling Mobility with Petri Hypernets*". 17th Int. Conf. on Recent Trends in Algebraic Development Techniques, WADT'04. LNCS vol. 3423, Springer-Verlag, 2004.
- [3] M. Buscemi and V. Sassone. "High-Level *Petri Nets as Type Theories in the Join Calculus*". In Proc. of Foundations of Software Science and Computation Structure (FoSSaCS '01), LNCS 2030, Springer-Verlag.
- [4] Dianxiang Xu and Yi Deng, "*Modeling Mobile Agent Systems with High Level Petri Nets*". 0-7803-6583-6/00/ © 2000 IEEE.
- [5] Francisco Durán, Steven Eker, Patrick Lincoln and José Meseguer. "*principles of mobile maude*". In D.Kotz and F.Mattern, editors, Agent systems, mobile agents and applications, second international symposium on agent systems and applications and fourth international symposium on mobile agents, ASA/MA 2000 LNCS 1882, Springer Verlag, Sept 2000.
- [6] Cédric Fournet Georges Gonthier, "*The Join Calculus: a Language for Distributed Mobile Programming*". In *Applied Semantics. International Summer School, APPSEM 2000, Caminha, Portugal, September 2000*, LNCS 2395, pages 268--332, Springer-Verlag. August 2002.
- [7] Alfonso Fuggetta, Gian Pietro Picco and Giovanni Vigna, "*Understanding Code Mobility*". IEEE transactions on software engineering, vol. 24, no. 5, may 1998.
- [8] Kees M. van Hee, Irina A. Lomazova, Olivia Oanea, Alexander Serebrenik, Natalia Sidorova, Marc Voorhoeve: "*Nested Nets for Adaptive Systems*". 14 EE. ICATPN 2006: 241-260.
- [9] P. Knudsen, "*Comparing Two Distributed Computing Paradigms, A Performance Case Study*"; MS thesis, Univ. of Troms 1995.
- [10] I.A. Lomazova. "*Nested Petri Nets*"; Multi-level and Recursive Systems. Fundamenta Informaticae vol.47, pp.283-293. IOS Press, 2002.
- [11] M. Merz and W. Lamersdorf, "*Agents, Services, and Electronic Markets: How Do They Integrate?*"; Proc. Int'l Conf. Distributed Platforms, IFIP/IEEE, 1996.
- [12] R. Milner. "*A Calculus of Communicating Systems*". Number 92 in Lecture Notes in Computer Science. Springer Verlag, 1980.
- [13] R. Milner, J. Parrow, and D. Walker. "*A calculus of mobile processes*". Information and Computation, 100:1-77, 1992.
- [14] Reinhartz-Berger, I., Dori, D. and Katz, S. (2005) "*Modelling code mobility and migration: an OPM/Web approach*", Int. J. Web Engineering and Technology, Vol. 2, No. 1, pp.6-28.
- [15] D. Sangiorgi and D. Walker. "*The π -Calculus: A Theory of Mobile Processes*". Cambridge University Press, 2001.
- [16] Athie L. Self and Scott A. DeLoach. "*Designing and Specifying Mobility within the Multiagent Systems Engineering methodology*". Special Track on Agents, Interactions, Mobility, and Systems (AIMS) at the 18th ACM Symposium on Applied Computing (SAC 2003). Melbourne, Florida, USA, 2003.
- [17] Tommy Thorn, "*Programming languages for mobile code*". Rapport de recherche INRIA, N ° 3134, Mars, 1997.
- [18] R. Valk. "Petri Nets as Token Objects: An Introduction to Elementary Object Nets". Applications and Theory of Petri Nets 1998, LNCS vol.1420, pp.1-25, Springer-Verlag, 1998.
- [19] F. Rosa Velardo, O. Marroqn Alonso and D. Frutos Escrig. "*Mobile Synchronizing Petri Nets: a choreographic approach for coordination in Ubiquitous Systems*". In 1st Int. Workshop on Methods and Tools for Coordinating Concurrent,

Distributed and Mobile Systems, MTCoord'05. ENTCS, No 150.

[20] Fernando Rosa-Velardo. "*Coding Mobile Synchronizing Petri Nets into Rewriting Logic*", this paper is electronically published in Electronic Notes in Theoretical Computer science URL: www.elsevier.nl/locate/entcs.

[21] Sutandiyono, W., Chhetri, M, B., Loke, S,W., and Krishnaswamy, S. "*mGaia: Extending the Gaia Methodology to Model Mobile Agent Systems*",

Accepted for publication as a poster in the Sixth International Conference on Enterprise Information Systems (ICEIS 2004), Porto, Portugal, April 14-17.

[22] D.J. Wetherall, J. Guttag, and D.L. Tennenhouse, "*ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols*" Technical Report, MIT, 1997, in Proc. OPENARCH'98.