Formal Modeling of Temporal Interaction Aspects in Multi-Agent Systems

Djamila Boukredera
Laboratoire des mathématiques appliquées
Université Abderrahmane Mira
Béjaia, Algérie
boukredera@hotmail.com

Ramdane Maamri
LIRE Laboratory
Université Mentouri
Constantine, Algérie
rmaamri@yahoo.fr

Abstract—**Multi-agent interaction protocols play a crucial role in multi-agent systems (MAS) development. They are used to manage and to control interactions among several autonomous agents in a MAS. Their formal specification, as well as their verification, constitute an essential task for the design of MAS applications. Several approaches have been proposed to formally represent agent interaction protocols, but there still lacks a formalism for representing temporal interaction constraints. This time dimension is an essential parameter in the protocol modeling seeing that most real world applications they support are time-sensitive. This paper proposes to use Timed Colored Petri Nets (TCPN) to model correctly and formally this temporal issue often defined as interaction duration and message deadlines. We then take the well-known Contract Net protocol (CNP) as an example to show that interaction protocols with time constraints can be modeled naturally and efficiently with this formalism. Finally, thanks to simulation techniques and state space analysis we will prove that the most important keys namely model correctness, deadline respect, absence of deadlocks and livelocks, absence of dead code, agent terminal states consistency, concurrency and validity are met.**

Keywords-Interaction protocols, Contract net protocol, Multi-agent systems, Timed Colored Petri Nets.

I. Introduction

Agent Interaction protocols (AIPs) represent an essential component of the dynamical model of a MAS. It is now recognized that interaction is the most important characteristic of complex systems. Based on many interacting agent components, such systems are generally time-sensitive and are known to be more complex to specify, to verify and to validate. However, the main step in designing an AIP is certainly the formal specification phase, which is crucial since it conditions the protocol design success. This paper is an extension version of the conference paper [1] and aims at providing a greater insight into the formal approach proposed to model the temporal aspects of any agent interaction protocol. Several formal models were proposed in the literature [2]–[8], but few works tackled the modeling of temporal interaction aspects, that are specified by FIPA. However, in current real life applications, time is of great importance and must be taken into account in all the design steps.

This paper addresses this issue and proposes to extend the AIPs with time constraints. We propose to use Timed Colored Petri Nets (TCPN) to formally model the two temporal constraints:

- Deadlines: it is a time constraint for message exchange. They denote the time limit by which a message must be sent. Once the deadline expires, the manager starts the evaluation of the received proposals. All proposals, which arrive after the due time will be considered to be invalid and consequently ignored.
- Duration: it is the interaction activity time period. It represents the time elapsed between the sending of a request message and the reception of the response. Duration includes two periods: transmission time and response time (task duration).

We adopt TCPN models because, besides their simplicity, they are particularly suitable in the modeling, simulating and analyzing of timed concurrent systems and, moreover, they use appropriate and powerful tools to generate interactive simulations of the modeled systems and apply a wide range of formal analysis alternatives. Our work contributes to the formal specification as well as the verification of the temporal interaction aspects in MAS. We then demonstrate the efficiency of our approach on the well-known CNP example, and prove that the key properties are satisfied. This contribution can be enumerated as follows: firstly, we present and we implement the proposed model using CPN Tools. We analyze it by means of the simulation and the state space techniques for various values of the protocol parameters namely the deadline and the number of participants. Secondly, we prove that the above mentioned key properties of the protocol are satisfied.

The rest of this paper is structured as follows: Section II describes the temporal interaction constraints. Section III briefly introduces the modeling methodology and the support tools. Section IV presents the CNP. In Section V, we detail the structure and the operation of the extended CNP. Simulation and state space analysis of our model are given in Section VI. Lastly, Section VII concludes the paper and gives some perspectives.

II. Modeling temporal aspects of interaction

Temporal constraints are time related relationships that must be reflected in a MAS modeling. Such constraints can be within the specification of either the internal behavior (vertical constraints) or the external behavior (horizontal constraints) of interacting agents. Most real-life applications are time-sensitive and may require that the timing constraints must be satisfied for correct operation and acceptable outputs. This is why it is important to take into account this temporal dimension in a MAS design.

In this paper, we will consider the two temporal interaction aspects specified by FIPA [9]: duration constraint and deadline constraint. The first one is the interaction activity time period, which includes two periods: transmission time and response time. Figure 1 illustrates the AUML [10] representation of the duration constraint. In our model, we have assumed that the transmission time t1 is infinitesimal and can consequently be ignored. On the other hand, the response time represents an activity duration and hence random functions are proposed to estimate it.

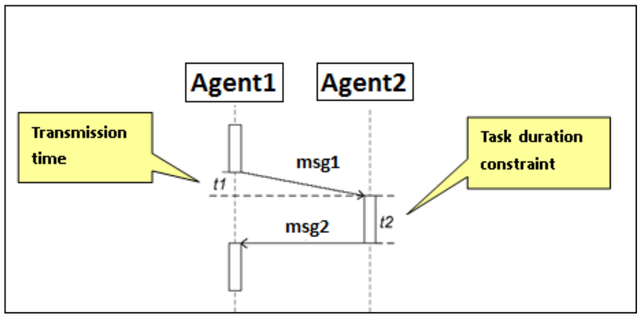


Figure 1: AUML representation of the interaction duration constraint

The second temporal aspect, deadlines, is a time limit for the message exchange. It may refer to a particular point in time by which a task must be accomplished or a time limit by which a message must be submitted. In most protocols, the moderator needs a deadline to decide whether a participant fails to reply or to meet a commitment and then to end an unachieved interaction. In this work, we model a deadline by means of timeout mechanism. In doing so, the moderator sets a wait time constraint (timeout) to receive replies from participants who must respond within this time limit, otherwise the response will be ignored. The response time value is declared as a random function depending on the specified deadline, which represent a key parameter of the timed model. Figure 2 illustrates the AUML modeling of the timeout mechanism. This time constraint indicates an alternative path when the deadline is reached. The alternative is therefore time-sensitive and this is graphically symbolized by the hourglass in the corner of the rectangle.

Figure 2 shows an agent *Seller* sending a proposal (*offer product*) to one or several receivers (*Buyers*) who have to answer by an offer before the expiration of deadline (equal to 100 units). Beyond this time limit, any answer from *Buyers* will be ignored and the *Seller* agent announces the identity of the winner buyer (*product sold (who)*). In this example, the *Seller* agent processes each bid received in the due time then determines and announces to the buyers the new top bid, if any. This process iterates until the deadline is reached.

Notice that AUML is one among the most used formalisms to represent agents' interactions [11]–[14]. However, AUML diagrams only offer a semi-formal specification of these interactions and their time constraints. This weakness can lead to several incoherencies in the description of MAS's behavior. That is why we prefer to adopt a more formal approach to specify agents' interaction, which obviously offers several advantages. Especially, it allows us to create more precise and rigorous specifications that can directly support verification and validation processes, and for which computer based support is available.

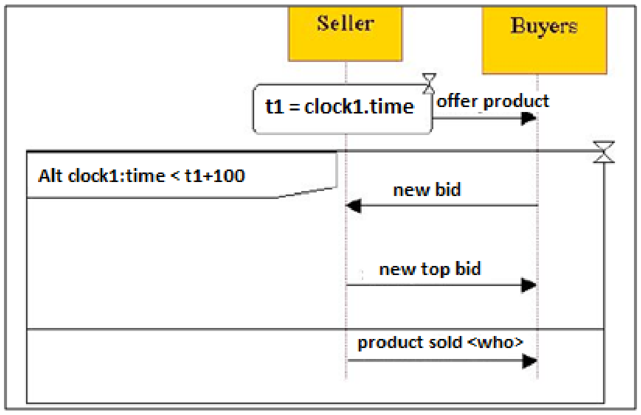


Figure 2: Representing the deadline by a timeout in AUML

III. Modeling methodologies and tools

When designing a software system we often have to deal with two central issues: (1) correctness and (2) performance. Verifying the correctness of the system means proving that it performs correctly all its specified functions and meets all the required key properties. The performance is a quantitative measure of how well a system works, it determines the usefulness of the system. The performance is often characterized by performance measures like: response times, waiting times, maximum capacity, etc.

To evaluate the correctness and performance of a complex system, we need powerful analysis methods and tools. Several formal specification techniques based on different theories exist in the literature, each of them has a preferred domain. In particular, in the field of interaction protocol systems, Petri nets have already proven to be extremely useful for description and analysis of such systems. They

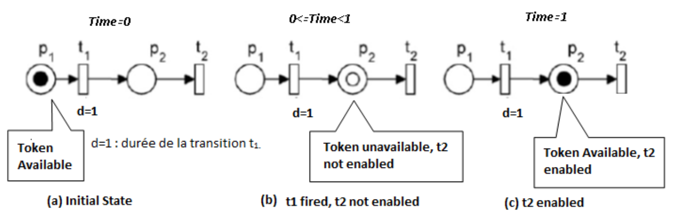

Time=0
0<=Time<1
Time=1
p1
t1
p2
t2
d=1
Token Available
d=1 : durée de la transition t1.
Token unavailable, t2 not enabled
Token Available, t2 enabled
(a) Initial State
(b) t1 fired, t2 not enabled
(c) t2 enabled

Figure 3: Timed Petri Nets with Holding Durations

follow an elaborated mathematical syntax and provide a clear, intuitive and demonstrative graphical representation of the model thus facilitating its simulation and analysis, which is a basic strength compared to other verification methods.

Since this work aims to assure the functional correctness of the proposed extended CNP, we adopt to use an approach based on Petri nets as suitable methodology and CPN Tools [15] as an adequate supporting tool suite, not only because of the maturity of both its theory and the related tool set CPN Tools [15] that this work rely on, but also because it allows to express and assess sophisticated temporal properties. In doing so, we assume a global clock.

A. Timed CPN

Petri nets are powerful tools for modeling, simulating, analyzing, supervising and debugging many complex distributed concurrent systems. TCPN, however are particularly suitable for real time systems because they allow the developer to produce a precise specification of the temporal behavior of such systems. The concept of time was not explicitly provided in the original definition of Petri nets. As described in [16], we distinguish three basic ways of representing time in CPN: Firing Durations (*FD*), Holding Duration (*HD*) and Enabling Duration (*ED*).

The principle of the *FD* is when a transition with a time delay becomes enabled, it removes immediately the tokens from the input places but does not create tokens in the output places until the firing duration has elapsed. However, if Holding Durations are included then the net semantics are changed. The principle of *HD* is based on two notions: the availability and the unavailability of the tokens. Available tokens can enable a transition whereas unavailable ones can not. In this case, when a transition, which is assigned a duration fires, removing and creating tokens are done instantaneously. However, the created tokens are unavailable and consequently can not enable any new transition until they have been in their output places for the time specified by the duration of the transition, which creates them. Figure 3 graphically illustrates the principle of HD.

In fact, *FD* and *HD* represent the same way of representing time. The only difference is that in *HD* the tokens are held by places whereas in *FD*, they are held by the transitions.

Besides, the *ED* leads to a different temporal behavior of the system. With *ED*, the firing of the transitions is done immediately; that is, removing and creating tokens are done instantaneously and the temporal duration is modeled by forcing the concerned transitions to be enabled for a specified period of time before they can fire. The main difference between *ED* and *HD* appears when there is conflicts in the petri nets, for more details the reader can refer to [16]. Choosing one of these three techniques depends strongly on the system to be modeled and its specifications. We should note, however, that it is natural to use *HD* technique in modeling most processes as transitions represent operation events which, once start, do not stop until they end. It is exactly the case of the system we are modeling. When a transition, which is assigned an *HD* duration, fires, removing and creating tokens are done instantaneously. However, the created tokens are not available to enable new transitions until they have been in their output place for the time specified by the transition, which created them. For more details concerning these three techniques of time modeling, the reader can refer to [16]. CPN versions, which use HD technique define implicitly the notion of tokens's unavailability by attaching to these tokens a timing attribute called a timestamp.

B. Formal definition of TCPN with Holding Durations

To represent tokens with timestamps we adopt the notation given by [17], [18]. Each token carries a timestamp preceded by the @ symbol. For instance, 2 tokens with timestamp equal to 10 are noted 2@10. The timestamp specifies the time at which the token is ready to be removed by an occurring transition. Timestamps are values belonging to a Time Set TS, which is equal to the set of non negative integers N+. The timed markings are represented as collection of timestamps and are multi-sets on TS: TS_{MS}. The formal definition of TCPN using holding durations is as follows: TCPN = (Σ, f, M_0) where Σ is a colored PN as described in [17]:

- Σ= (S, P, T, A, N, C, G, E) where:
  - S is a finite set of non-empty types, called color sets.
  - P is a finite set of places.
  - T is a finite set of transitions.
  - A is a finite set of arcs such that: $P \cap T = P \cap A = T \cap A = \emptyset$.
  - N is a node function. It is defined from A into P $\times$ T $\cup$ T $\times$ P.
  - C is a colour function. It is defined from P into S.
  - G is a guard function. It is defined from T into expressions such that: $\forall$ t$\in$ T: [Type(G(t)) = Bool $\wedge$ Type(Var(G(t))) $\subseteq$ S].
  - E is an arc expression function. It is defined from A into expressions. The arc expression associates

with every arc an expression, which will be used to verify or create new token-values. Every arc expression should evaluate to a set of tokens (a multi-set over the different types allowed by the place). E contains input expressions as well as outgoing actions.

- **f**: T $\rightarrow$TS represents the transition function, which assigns to each transition t $\in$ T a non negative determinist duration
- **M**: P $\rightarrow$ TS_{MS} is the timed marking, M_0 represents the initial marking of TCPN.

To determine whether tokens are available or unavailable, we define functions over the marking set M. So, For a marking M and the given model time (global clock), we have:
m: $P \times M \times TS \rightarrow$ N, which defines the number of available tokens and n: $P \times M \times TS \rightarrow$ N, which defines the number of unavailable tokens for each place of the TCPN model at a given instant k, where k and the model time belong to TS. There are several computer tools, which perform automatic validation and verification of Petri net models. Nevertheless, only CPN Tools permits, besides time representation, the modeling of high level petri nets particularly colored and hierarchical ones.

C. *CPN Tools*

CPN Tools [15] developed at the University of Aarhus is a strength tool for constructing, editing, simulating and analyzing CPN models. Using CPN Tools, it is possible to perform investigation of modeled system design and behavior using simulation, to verify properties by means of state space methods and model checking, and to conduct simulation-based performance analysis. CPN Tools proposes very powerful class of Petri nets for models' description namely hierarchical timed colored Petri nets, which we have chosen to use for our modeling. The language description is a combination of Petri net graph and programming language CPN ML (Markup Language). Notice that the functionality of the tool can be extended with user-defined Standard ML functions.

In the following, we will consider the CNP as an example to illustrate our proposition.

IV. The Contract Net Protocol

CNP, originally proposed by Smith [19], is one of the most popular interaction protocols used in diverse negotiation contexts. Developed to resolve decentralized task allocation, the CNP represents a distributed negotiation model based on the notion of call for bids. In this protocol, agents can dynamically take two roles: manager or contractor (initiator or participant according to FIPA terminology [9]).

In CNP as illustrated by the AUML diagram of Figure 4, a manager and participants interact with one another to find a solution for a problem through a four-stage negotiation process. The manager initiates the negotiation process by

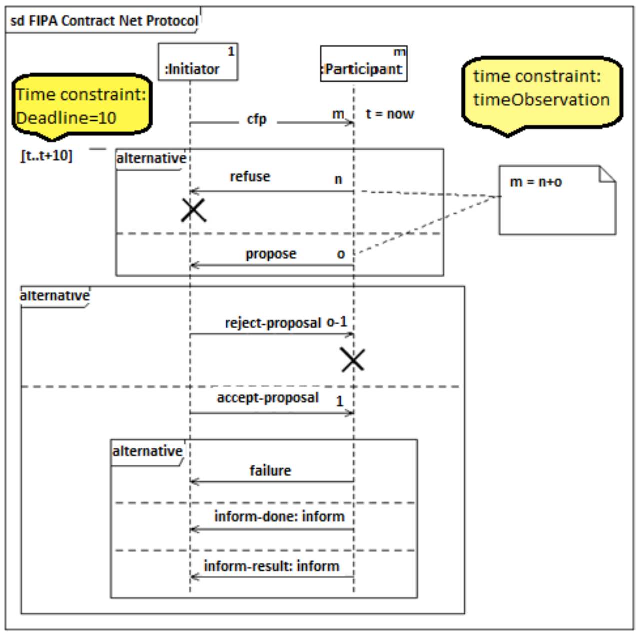


Figure 4: AUML diagram of the contract net protocol

issuing a Call For Proposals (CFP) announcing the task specification to a number of potential participants. The CFP includes a deadline by which the participants must respond with bids. Participants evaluate the CFP and decide whether to answer with a refusal message or with a proposal to perform the task. Once the deadline expires, the manager evaluates all the received proposals (in due time) and, in turn, awards the contract to the most appropriate participant, which becomes a contractor. The manager ignores any proposal that arrives beyond the deadline. The contractor performs the task and sends to the manager an informing message, which can be an error one in the case of a failure. Consequently, the negotiation process includes several scenarios depending on whether the bid process ends with or without a contract, and as the execution of the task ends with or without a success. Therefore, the manager and the participants can reach various states during this process. We suggest to represent the internal behavioral of both types of agents by means of AUML2 statesharts diagrams [10]. These diagrams define the different states that will be later used in the TCPN model of the protocol. Figure 5 (a) and Figure 5 (b) illustrate respectively the internal behavior of the manager and the participant agents. Table I summarizes the various states and their semantics.

V. TCPN model of the Contract Net Protocol

When modeling a protocol, there are several design requirements and key characteristics that this protocol should satisfy. Authors in [6] have summarized these issues in 5 factors: state set, role set, rule set, action set and message set. By analogy with our case study, Table I describes

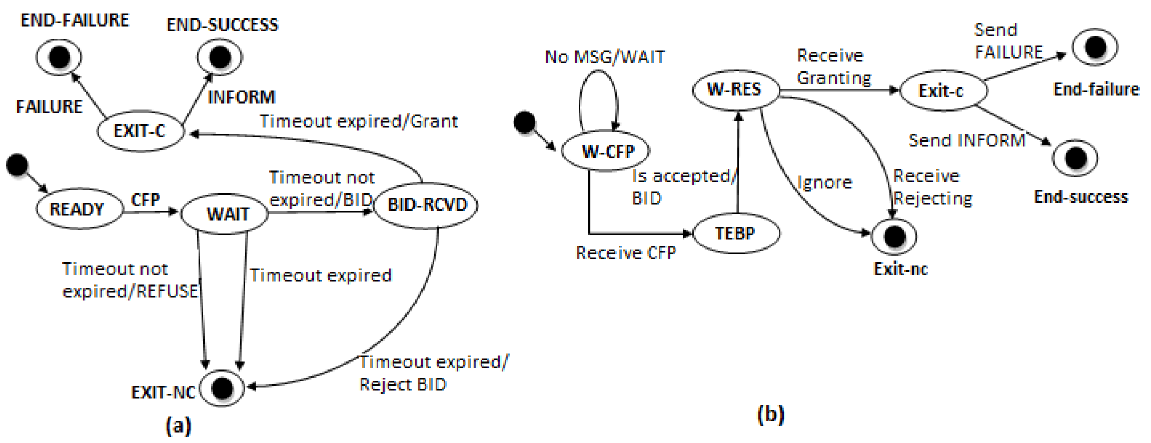


Figure 5: Internal behavior of the manager and the participant agents

Table I: Representation of states

Manager (Initiator)	Participant
READY (READY to send a CFP)	W-CFP (Waiting for CFP)
WAIT (Waiting for bids or for time-out)	TEBP (Task evaluation and bid preparation)
BID-RCVD (Bid received)	W-RES (Waiting for result)
EXIT-NC (EXIT with no contract)	Exit-nc (exit with no contract)
EXIT-C (EXIT with contract)	Exit-c (exit with contract)
END-SUCCES (END of negotiation with SUCCESS)	End-success (end of task execution with success)
END-FAILURE (END of negotiation with FAILURE)	End-failure (end of task execution with failure)

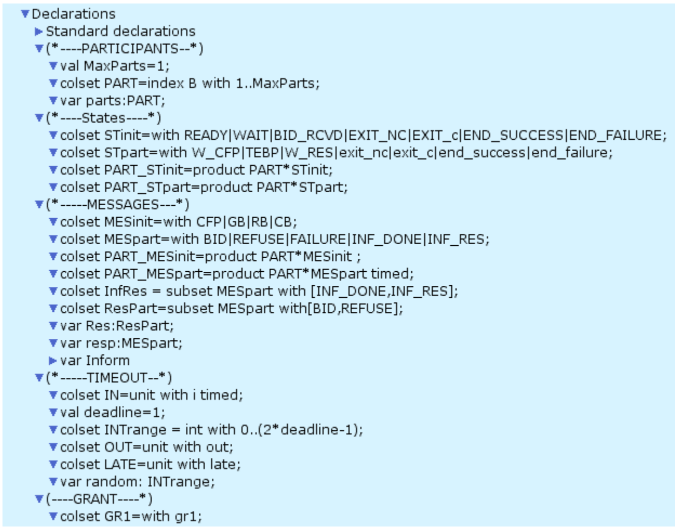
Table II: Representation of messages in the TCPN model

Messages issued by the manager	Messages issued by the participant
CFP (Call For Proposals)	BID (BID)
GB (Grant Bid)	REFUSE (REFUSE CFP)
RB (Reject Bid)	FAILURE (task Execution FAIL-URE)
CB (Cancel Bid)	INF-DONE (INForm-Done)
	INF-RES (INForm-RESults)

the various states that negotiation process should reach and Table II defines messages exchanged between the manager and the participants. This section highlights our contribution and presents how Contract Net Protocol extended with the temporal aspects described in Section II can be modeled as TCPN using CPN Tools. When creating the model, we have assumed some assumptions such as the reliability of the communication channel, and that participants have to reply to the CFP. Moreover, when modeling the interaction following the contracting phase, we should not take into consideration task duration, given that this work focuses on temporal interaction aspects. The manager starts evaluating bids after deadline expiration and lastly, the details of messages exchanged are excluded for a sake of abstraction.

A. Declarations

Being inspired by [2], our TCPN model is readable and has a compact structure. For each type of agents, we use a single place, which would store all its possible states.



```
▼Declarations
  ▶Standard declarations
  ▼(*----PARTICIPANTS--*)
    ▼val MaxParts=1;
    ▼colset PART=index B with 1..MaxParts;
    ▼var parts:PART;
  ▼(*----States----*)
    ▼colset STinit=with READY|WAIT|BID_RCVD|EXIT_NC|EXIT_c|END_SUCCESS|END_FAILURE;
    ▼colset STpart=with W_CFP|TEBP|W_RES|exit_nc|exit_c|end_success|end_failure;
    ▼colset PART_STinit=product PART*STinit;
    ▼colset PART_STpart=product PART*STpart;
  ▼(*-----MESSAGES---*)
    ▼colset MESinit=with CFP|GB|RB|CB;
    ▼colset MESpart=with BID|REFUSE|FAILURE|INF_DONE|INF_RES;
    ▼colset PART_MESinit=product PART*MESinit ;
    ▼colset PART_MESpart=product PART*MESpart timed;
    ▼colset InfRes = subset MESpart with [INF_DONE,INF_RES];
    ▼colset ResPart=subset MESpart with[BID,REFUSE];
    ▼var Res:ResPart;
    ▼var resp:MESpart;
    ▶var Inform
  ▼(*-----TIMEOUT--*)
    ▼colset IN=unit with i timed;
    ▼val deadline=1;
    ▼colset INTrange = int with 0..(2*deadline-1);
    ▼colset OUT=unit with out;
    ▼colset LATE=unit with late;
    ▼var random: INTrange;
  ▼(----GRANT----*)
    ▼colset GR1=with gr1;
```

Figure 6: Declarations for the TCPN model of the CNP

Similarly, we distinguish two places, which represent a reliable channel for both directions of the communication. Figure 6, taken directly from CPN Tools, shows all the declarations used in the model.

B. Model structure

Figure 7 shows the TCPN diagram of CNP. The manager with the timeout mechanism is modeled on the left, the participants on the right. They communicate via a reliable not ordered channel represented by the two places INIT2PART and PART2INIT. The place INIT2PART only contains messages issued by the manager to the participants. Respectively, PART2INIT only contains messages of the participants to the manager. In this model, the timed messages carry timestamps indicating when they should be available. Initially, the manager is in the state READY with respect to all the participants. Whereas, all the participants

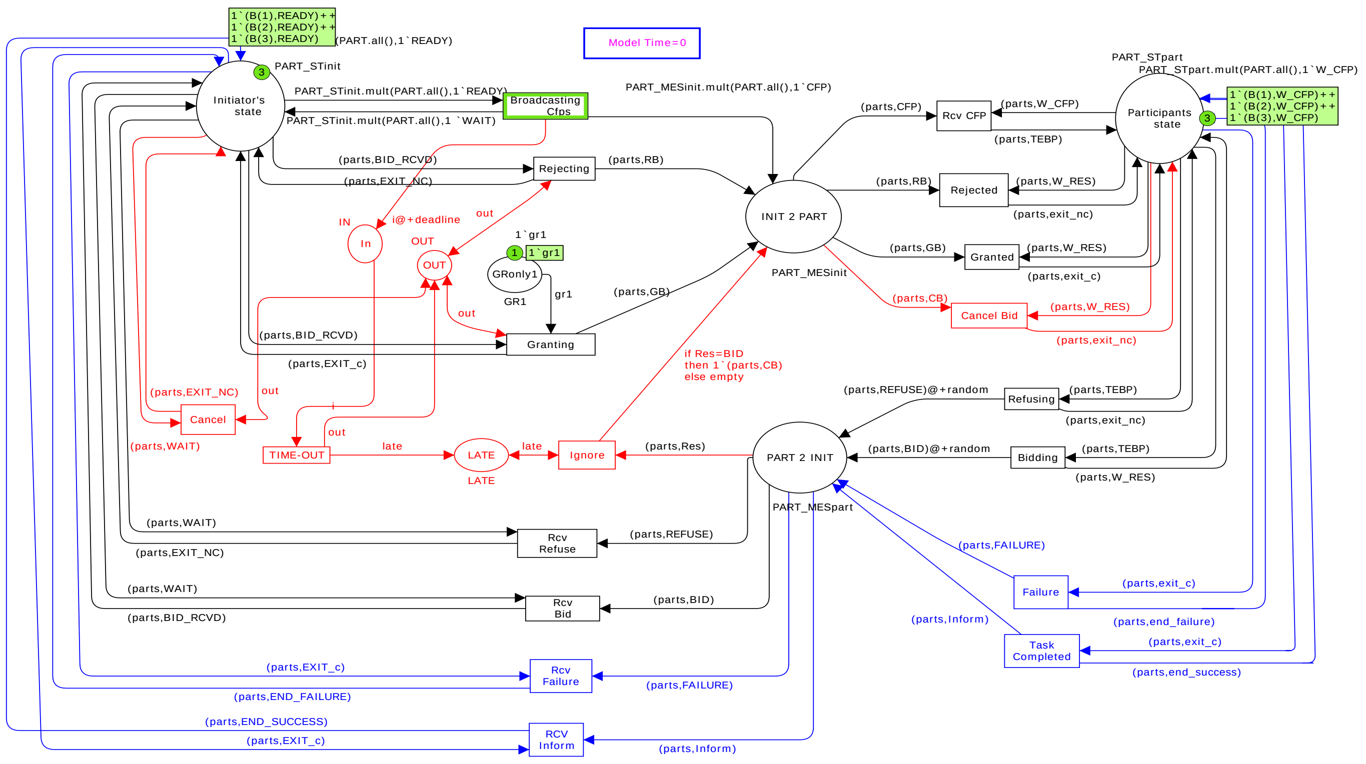


Figure 7: TCPN diagram of the contract net protocol

are in the state W_CFP. The place GRonly1 contains one token GR1 and all the other places are initially empty.

C. *Operation of the model*

Initially, the time model is equal to 0. The manager (in the state READY with respect to all the participants) is ready to send a CFP to each of the participants (which are in the state W_CFP). The manager initiates the negotiation process by issuing the CFP to all the participants. The transition *Broadcasting Cfps* is then fired and the manager changes state to WAIT with respect to all these participants. At the same moment, the timer is armed with the deadline via creating a timed token in the place In. This token carries a timestamp equal to deadline and, therefore, does not enable the transition *TIME-OUT* because it is unavailable for this duration. The transition *Rcv CFP* is consequently enabled and can be concurrently fired by all the participants. Once a participant removes its CFP from the place INIT2PART it changes state to TEBP and starts evaluating the given task, based on its capabilities and available resource. Then, it decides to make an offer or to refuse the request. In the first case, the participant have to prepare the bid that satisfies the criteria specified in the CFP. As mentioned above, and since they does not affect the operation of the model, all message details are omitted for an abstraction concern. At this point, both transitions *Refusing* and *Bidding* are enabled and the choice of the transition to fire is non deterministic. In the case where the transition *Bidding* fires then the participant changes state to W-RES (waiting for decision about its submitted bid). On the other hand, if the transition *Refusing* has rather occurred then the participant changes state to exit-nc (end of negotiation, with this participant, without a contract). The occurrence of either transition creates a timed token in the place PART2INIT, which represents the reply message. The timestamp of this token is randomly calculated by the predefined function *random* based on the interval [0..2*deadline-1]. In doing so, we assume that the response time of the participant cannot exceed 2 times the given deadline. It should be noted that any message gets through the place PART2INIT in due time updates the state of the manager. In this case, once the time allocated to the timed message expires the transition *Rcv Bid* or *Rcv Refuse* is enabled according to the message BID or REFUSE respectively. It is then that the manager changes state to BID-RCVD or EXIT-NC (with respect to this particular participant) depending on the occurrence of the *Rcv Bid* or *Rcv Refuse* respectively. This process of updating the manager state with respect to any participant reply arrived in the due time continuous until the deadline expires, i.e., the token in the place In becomes available and the transition *TIME-OUT* is, therefore, enabled and fired creating two tokens: one in the place OUT, which could enable both transitions *Rejecting* and *Granting*, and an other in the place LATE, which could enable the transition *Ignore* if any reply gets through the channel after the deadline. At this point, we distinguish 3 scenarios:

- ***Scenario 1***: *All the participants reply within the due time*. In this case, the transitions *Ignore* and *Cancel* would never be enabled. The manager starts evaluating the bids received (if any) and according to its negotiation strategy decides to accept or to reject a given offer. Note that the manager could grant any of the bids or reject all the bids. If any bid was received then both transitions *Rejecting* and *Granting* are concurrently enabled. In the case where the manager opts to accept a given bid, then the transition *Granting* is fired and a message GB (Grant Bid) is sent to the concerned participant. The token *gr1* is removed from the place GRonly1, signifying that only one bid could be granted. All the other received bids would be, therefore, rejected and a RB (Reject Bid) message must be sent to the corresponding participants via the transition *Rejecting*. In this situation, the manager would be in the state EXIT-C (end of negotiation with contract) with respect to the participant of the granted bid and in the state EXIT-NC with respect to the rest of the participants.
  Another possibility is that the manager could reject all the received bids leading to an end of negotiation without contract. In this case, the transition *Granting* would not occur and the manager would be in the state EXIT-NC with respect to all the participants. At this point, the transition *Rejected* or *Granted* is enabled depending on the message RB or GB respectively. The fire of the transition *Rejected* causes the participant to change state to exit-nc, while the occurrence of the transition *Granted* causes the participant to change state to exit-c. At this latter case, the negotiation ends with a contract and we propose to model the following bilateral interaction between the manager and the winning participant. That is, once the participant performs the task, it would complete it either with a success or a failure. We model this process of task completion non-deterministically. Thus, both the transitions *Failure* and *Task Completed* would be enabled and concurrently fired. On occurrence of the transition *Failure*, the participant change state to end-failure and sends a FAILURE message to the manager. However, if the transition *Task Completed* occurs then the participant change state to end-success and sends an inform message (which could be INF-DONE or INF-RES) to the manager. Once the message reaches the manager, the transition *Rcv Failure* or *Rcv Inform* would be enabled depending on the message FAILURE or Inform respectively. Firing the transition *Rcv Failure* causes the manager to change state to END-FAILURE (end of the negotiation with a failure), while the occurrence of the transition *Rcv Inform* causes the manager to change state to END-SUCCESS (end of the negotiation with success). It should be noted that the task duration has not been modeled, this is because this work focuses on representing temporal interaction aspects and not the real time task management. This would be the subject of a future work.
- ***Scenario 2***: *Some replies get through the channel after the deadline*. In this case and once the transition *TIME-OUT* occurs, two concurrent processes could be conducted by the manager: evaluation of the bids received (if any) and cancelation of any CFP that have not yet received a response. The first process operates in a similar way as mentioned in scenario1 where the negotiation could end either with none contract or with a contract awarded to one participant, which would complete the execution of the task with a success or a failure. In the second process, however, the transition *Cancel* is chosen and fired, implying that the manager would not wait any more the late replies and, consequently, it changes state to exit-nc with respect to those late participants. In the other hand, the occurrence of the transition *TIME-OUT* puts a token in the place LATE, which would enable the transition *Ignore* (ignore all late replies) every time a late message in the place PART2INIT becomes available. In the case the late message is a Bid, then the transition, whose its guard evaluates to true, fires and sends a CB (Cancel Bid) message to that participant. This is causes the enabling of the transition *Cancel Bid*, which once occurred, updates the state of the corresponding participant to exit-nc. In doing so, the bidders do not risk to wait indefinitely for a decision about their submitted bids. Moreover, we assure that at the end of the negotiation, the manager and the participants would be in consistent terminal states. Note that if the late message is REFUSE then the corresponding participant is already in the state exit-nc and the transition *Ignore* is, thus, a sink transition.
- ***Scenario 3:*** *All the replies get through the channel after the deadline*. In this particular case and once the transition *TIME-OUT* occurs, only the transition cancel is enabled. It operates in the same way as mentioned above and causes the manager to change state to EXIT-NC with respect to all the participants. This is the case where the negotiation process ends without a contract because of a deadline overrun (by all the participants). The late messages would be consumed by the the transition *Ignore* as soon as they become available, allowing, thus, the net cleaning.

VI. Verification of the model

Verification is a method to exhaustively examine a design and check to make sure certain predefined key properties are met. There are several software tools to automate this task, however, CPN Tools [15] is currently the most used tool for high level Petri nets particularly for the timed colored ones (TCPN). This tool helps us to assess the correctness of the model.

A. Simulation

Using CPN simulator, we have conducted several automatic and interactive simulations, which help us to identify and resolve several omissions and errors in the design. In simulation runs carried out the protocol terminated correctly and the agents were in the desired and coherent states. Interactive simulation also shows that the characteristics such as concurrency and validity are satisfied. This makes it likely that the protocol works correctly but it cannot guarantee that simulation covers all possible executions. That is why simulation cannot be used to verify other functional and performance properties such as the absence of deadlocks and others. However, state space analysis techniques allow us to verify if the system satisfies these behavioral properties.

B. State space analysis

With regard to untimed CPN models, calculating timed state space is a non trivial task and can be quite difficult and time consuming. This is because the reachability graph is too large and can be infinite even if the state space of the corresponding untimed CPN model is finite. This is due to the fact that several timed markings including global clock and timestamps can be different even if the corresponding untimed markings are identical. That is why we have to use some CPN ML (CPN Meta Language) queries to verify some properties.

Model Correctness. In this section, we verify the absence of deadlocks and the consistency in beliefs between the manager and the participants. Table III presents the state space analysis results. It shows the properties of the state space obtained by varying the parameter *MaxParts* (Maximum number of Participants) from 1 to 4 and the parameter deadline from 1 to 5. The analyzing of the property DeadMarking allows us to verify the model correctness. Each dead marking corresponds to a terminal state of the negotiation protocol. All dead markings are obtained after the deadline expiration, i.e., from t=d to t=2*d-1 (proposed estimation for the participants response time), for each discrete value of t belonging to this interval. For any value of *MaxParts*, one of the dead markings corresponds to an end of negotiation without a contract. In this marking, all the participants are in the state exit_nc and the manager in the state EXIT_NC with respect to all the participants. This is illustrated by the marking 14 in Figure 8. The description of this node shows that the place GRonly1 has still the token GR1 implying that none bid had been granted. The place In is empty, signifying that the deadline has expired and the timeout has fired. This particular dead marking is acceptable because the manager may reject all the bids or may not receive any bid in the due time. This worst case can be reached by 7 paths describing the pessimistic scenarios that can occur before and after the deadline. Figure 9 shows some paths examples, which lead to this particular case where the negotiation ends with no contract awarded. For example, the path (1,2,3,7,13,15,21,14) of Figure 9a corresponds to the scenario where the participant has issued an offer at t = 0, but it was rejected by the manager; the path (1,2,3,5,10,14) of Figure 9b corresponds to the situation where the participant has refused the CFP. We note that if a participant's response arrives at t = d then the choice between firing the timeout or receiving the response is non-deterministic: the transitions *TIME-OUT* and *Rcv_Bid* (or *Rcv_Refuse*) are concurrent, which leads to two different paths in the reachability graph. This is the case, for example, of the two paths (1,2,3,6,12,15, ...) and (1,2,3,6,11,18,22,14) in Figure 9c where the node 6, which corresponds to the reception of an offer at t=d is followed either by the node 12 (offer reception by the manager), or by the node 11 (timeout firing and hence cancelation of the offer and the end of the negotiation without contract). This case confirms that the concurrency property is satisfied in the model. Among the rest of the dead markings, we distinguish those calculated at t=d and those obtained at t>d:

At t = *d* and for any values of *MaxParts*: besides the particular dead marking mentioned above, the dead markings calculated at this time corresponds to the end of negotiations where a contract has been awarded to one participant (i=1..*MaxParts*) while the rest of negotiation with the rest of participants has ended without a contract. Therefore, P_i changes state to exit_c, performs the task, which can ends by a success or a failure. P_i can, then, be in the state end_success or end_failure respectively. At the same time, the manager, which was in the state EXIT_C with respect to P_i (and EXIT_NC with respect to the rest of the participants) changes to END_SUCCESS or END_FAILURE with regard to P_i. All the other participants P_j (j≠i) are in the state exit_nc. Thus, we can deduce that at t=d and for any value of *MaxParts* we have:

$$NumberDeadmarkings = (2*MaxParts + 1)$$

The rest of the dead markings is calculated at t>d, which correspond to scenarios after the fire of the timeout where at least one participant is not in the due time. Two cases can be distinguished: a particular case of a single participant (*MaxParts*=1) and a general case of several participants (*MaxParts* > 1):

t > *d* and *MaxParts* = 1: this is particular because the single participant may miss the deadline and, consequently, changes state to exit_nc because of the canceling of its late response. The manager is in the state EXIT_NC with respect

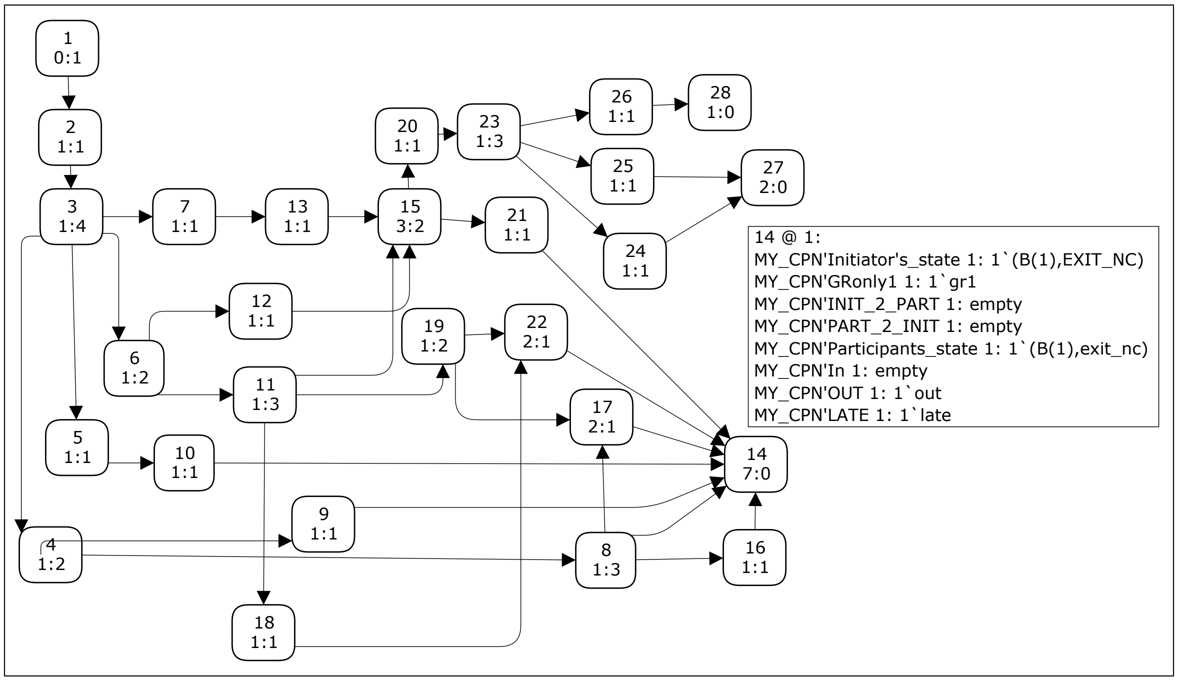


Figure 8: State space for (*MaxParts* = 1 et *d* = 1)

to this participant. This corresponds to the end of negotiation without a contract caused by the deadline overrun. This dead marking is reached for any discrete value of t where *d*<t>=2**d*-1, i.e., (*d*-1) times and thus we deduce:

$$NumberDeadmarkings = 2 * MaxParts + d \quad (1)$$

which is equal in this case to (2+*d*).

t > *d* and *MaxParts* > 1: all the dead markings calculated after the timeout and for each discrete value in the interval (*d*..2**d*-1) are similar to those obtained at t=*d*. The only difference is that the global clock values and the timestamps of the tokens differ. Thus, these are equivalent timed markings. Consequently, we obtain (*d*-1) times the same number of dead markings, i.e., (*d*-1)* (2**MaxParts* +1) and, therefore, we deduce:

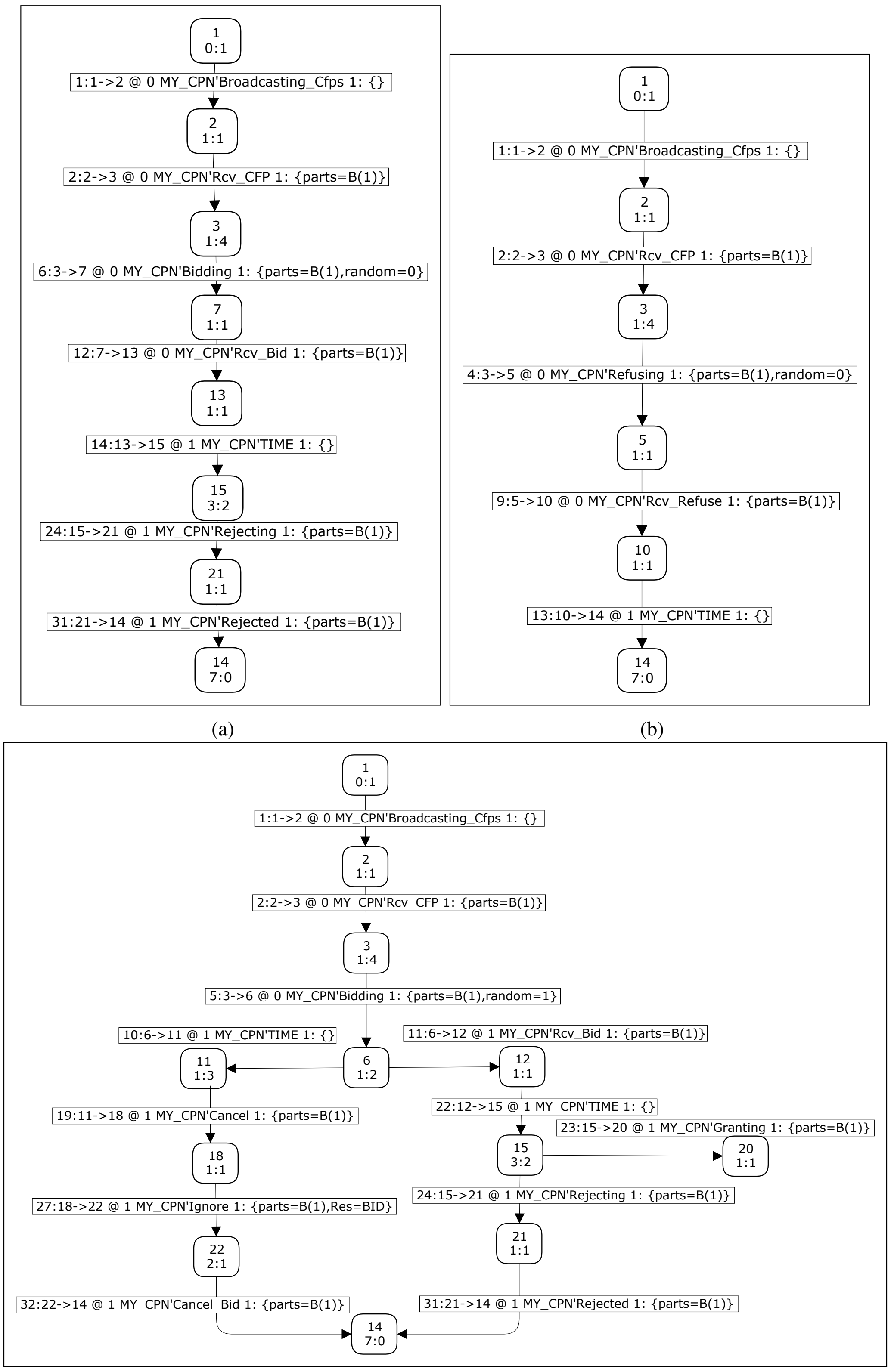
$$NumberDeadmarkings = (2 * MaxParts + 1) * d \quad (2)$$

All these dead markings are desired terminal states of the protocol. This discussion justifies that the protocol works correctly and the beliefs between the manager and the participants are consistent. Also, it should be noted that if for a given marking two or more transitions are enabled, then the choice of the transition to fire is non-determinist. This means that our system satisfies concurrency and non-determinism, which are key characteristics. About the communication channel, we note that at the end of negotiations, the places PART2INIT and INIT2PART are empty, signifying that there is no unprocessed messages in the network, proving, hence, that the property of cleaning the network from late messages is satisfied.

Absence of livelocks and correct termination. Table III shows that the size of the state space increases exponentially with the number of participants and the value of the deadline. This is illustrated by the graph of the Figure 10. The large number of nodes and particularly of dead markings is essentially caused by the increasing value of the deadline. The reason for this is that the timing information makes more markings distinguishable and contributes to the presence of more nodes in the state space leading to several equivalent timed markings. To verify that all the dead markings for all the values of *MaxParts* specified in Table III form a home space, we have used the CPN ML function HomeSpace (ListDeadMarkings()), which evaluates to true. This confirms that there is no livelocks and the system will always terminate correctly. Table III also shows that, for all values of *MaxParts* examined, the number of nodes and arcs in the SCC graph always remains the same as that of the state space, this implies that there is no cyclic behavior in the system, which is expected. From Table III, we conclude that there is no live transitions because of the presence of dead markings.

Absence of dead code. A dead code corresponds to a dead transition. According to Table III, there is no dead transitions in the system for all values of *MaxParts* examined, this implies that all the specified actions are executed.

Channel bound. Table III shows that the communication


1 0:1
1:1->2 @ 0 MY_CPN'Broadcasting_Cfps 1: {}
2 1:1
2:2->3 @ 0 MY_CPN'Rcv_CFP 1: {parts=B(1)}
3 1:4
6:3->7 @ 0 MY_CPN'Bidding 1: {parts=B(1),random=0}
7 1:1
12:7->13 @ 0 MY_CPN'Rcv_Bid 1: {parts=B(1)}
13 1:1
14:13->15 @ 1 MY_CPN'TIME 1: {}
15 3:2
24:15->21 @ 1 MY_CPN'Rejecting 1: {parts=B(1)}
21 1:1
31:21->14 @ 1 MY_CPN'Rejected 1: {parts=B(1)}
14 7:0
(a)
1 0:1
1:1->2 @ 0 MY_CPN'Broadcasting_Cfps 1: {}
2 1:1
2:2->3 @ 0 MY_CPN'Rcv_CFP 1: {parts=B(1)}
3 1:4
4:3->5 @ 0 MY_CPN'Refusing 1: {parts=B(1),random=0}
5 1:1
9:5->10 @ 0 MY_CPN'Rcv_Refuse 1: {parts=B(1)}
10 1:1
13:10->14 @ 1 MY_CPN'TIME 1: {}
14 7:0
(b)
1 0:1
1:1->2 @ 0 MY_CPN'Broadcasting_Cfps 1: {}
2 1:1
2:2->3 @ 0 MY_CPN'Rcv_CFP 1: {parts=B(1)}
3 1:4
5:3->6 @ 0 MY_CPN'Bidding 1: {parts=B(1),random=1}
10:6->11 @ 1 MY_CPN'TIME 1: {}
11:6->12 @ 1 MY_CPN'Rcv_Bid 1: {parts=B(1)}
11 1:3
6 1:2
12 1:1
19:11->18 @ 1 MY_CPN'Cancel 1: {parts=B(1)}
22:12->15 @ 1 MY_CPN'TIME 1: {}
23:15->20 @ 1 MY_CPN'Granting 1: {parts=B(1)}
18 1:1
15 3:2
20 1:1
27:18->22 @ 1 MY_CPN'Ignore 1: {parts=B(1),Res=BID}
24:15->21 @ 1 MY_CPN'Rejecting 1: {parts=B(1)}
22 2:1
21 1:1
32:22->14 @ 1 MY_CPN'Cancel_Bid 1: {parts=B(1)}
31:21->14 @ 1 MY_CPN'Rejected 1: {parts=B(1)}
14 7:0

(c)

Figure 9: Examples of paths leading to the worst case (*MaxParts* = 1 et *d* = 1)

Table III: State space analysis results as a function of the parameters *MaxParts* and deadline (*d*)

Properties	***MaxParts*=1**					***MaxParts*=2**					***MaxParts*=3**					***MaxParts*=4**	
	d=1	*d*=2	*d*=3	*d*=4	*d*=5	*d*=1	*d*=2	*d*=3	*d*=4	*d*=5	*d*=1	*d*=2	*d*=3	d=4	*d*=5	*d*=1	*d*=2
State Space Nodes	28	40	52	64	76	317	605	989	1469	2045	3669	9165	18645	33216	54164	42337	140513
State Space Arcs	38	53	68	83	98	801	1357	2081	2973	4033	14113	30143	55863	93817	146549	221393	619193
Time (seconde)	00	00	00	00	00	00	00	00	01	02	07	33	161	404	831	1298	16119
SCC nodes	28	40	52	64	76	317	605	989	1469	2045	3669	9165	18645	33216	54164	42337	140513
SCC Arcs	38	53	68	83	98	801	1357	2081	2973	4033	14113	30143	55863	93817	146549	221393	619193
Dead Markings	3	4	5	6	7	5	10	15	20	25	7	14	21	28	35	9	18
HomeSpace	true	true	true	true	true	true	true	true	true	true	true	true	true	true	true	true	true
Dead Transition Instances	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None
Live Transition Instances	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None
Channel bound	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	4	4

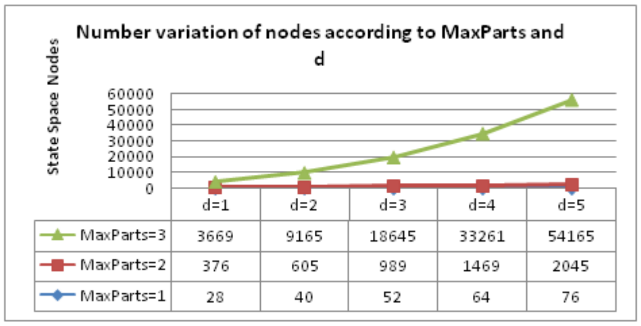


Figure 10: Number variation of the reachability graph nodes according to *Maxparts* and the deadline

channel is bounded by the MaxParts value examined, this confirms that the manager issues a single message to each of the participants and then *MaxParts* messages. Similarly, each participant issues, at a given moment, one message to the manager justifying the limit of MaxParts responses.

VII. Conclusion and perspectives

In this paper, we have proposed to extend the AIPs with temporal aspects. We have taken the CNP as an example to illustrate our approach. A TCPN model of the contract net protocol was presented and thanks to the simulation and the state space analysis techniques we have proved that the proposed model satisfies some key properties for different values of both parameters *MaxParts* and deadline. We have also proved the beliefs consistency between the manager and the participants and that the protocol works and ends correctly. The properties namely concurrency, absence of livelocks and absence of dead code were verified too. Furthermore, we have shown how the number of dead markings (terminal states) is related to both *MaxParts* and deadline parameters. The channel bound is, however, related to only the *MaxParts* parameter.

As perspectives, we would like to use advanced state space reduction methods [20], [21] like equivalence classes to alleviate the impact of the state explosion problem, which is most accentuated for timed models. In doing so, we would verify the model for wider values of the protocol parameters. We would also like to model real time contract net [22] where, besides interaction aspects, time constraints related to task execution would be considered. These extensions would concern more complex versions of CNP. On the other hand, we would like to take into account the commitment violation by modeling a fault tolerant AIP [23] so that the sender provides a fault tolerant behavior if ever the receiver crashes during task performing or fails to meet a commitment.

References

[1] D. Boukredera, R. Maamri, and S. Aknine, *A Timed Colored Petri-Net-based Modeling for Contract Net Protocol with Temporal Aspects*, in Proceedings of the Seventh International Multi-Conference on Computing in the Global Information Technology (ICCGI 2012), pp 40-45, Venice (Italy), June 24-29, 2012.

[2] J. Billington and A. Gupta, *Effectiveness of Coloured Petri Nets for Modelling and Analysing the Contract Net Protocol*, Proc. Eighth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, Aarhus, Denmark, 2007, pp. 49-65 (ISSN 0105 8517).

[3] S. Aknine, S. Pinson, and M. F. Shakun, *An Extended Multi-Agent Negotiation Protocol, Autonomous Agents and Multi-Agent Systems* 8(1), pp. 5-45 (2004).

[4] J. Shujuan, Q. Tian, and Y. Liang, *A Petri-Net-Based Modeling Framework for Automated Negotiation Protocols in Electronic Commerce*, Lecture Notes in Computer Science, 2009, Volume 4078/2009, pp. 324-336.

[5] J. Billington, A. K. Gupta, and G. E. Gallasch, *Modelling and Analysing the Contract Net Protocol - Extension Using Coloured Petri Nets*, Lecture notes in computer science, 2008, NUMB 5048, pp. 169-184, Springer-Verlag.

[6] F. S. Hsieh, *Automated Negotiation Based on Contract Net and Petri Net*, Lecture Notes in Computer Science, vol. 3590, pp. 148-157, 2005. (SCI).

[7] W. L. Yeung, *Behavioral modeling and verification of multi-agent systems for manufacturing control*, Expert Systems with Applications, 38(11):13555-13562, 2011.

[8] L. Changyou and W. Haiyan, *An Improved Contract Net Protocol Based on Concurrent Trading Mechanism*, iscid, vol. 2, pp. 318-321, 2011 Fourth International Symposium on Computational Intelligence and Design, 2011.

[9] *FIPA*, Foundation for intelligent physical agents 2003. *FIPA Modeling Area: Temporal Constraints*. Retrieved May 10, 2012, from *http://www.fipa.org*.

[10] *Agent Unified modeling language, AUML*. Retrieved May 15, 2012, from *http://www.AUML.org*.

[11] L. Cabac, *Modeling Agent Interaction with AUML Diagrams and Petri Nets*, Diplomarbeit, University of Hamburg, Department of Computer Science, Vogt-Kolln Str. 30, 22527 Hamburg, Germany, 2003.

[12] M .P. Huget and J. Odell, *Representing Agent Interaction Protocols with Agent UML*, In Proceedings of the Third International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004), New York, USA, July 2004.

[13] M. P. Huget, *Extending Agent UML Protocol Diagrams*, In Proceedings of Agent Oriented Software Engineering (AOSE-02), Fausto Giunchiglia and James Odell and Gerhard Weiss (eds.), Bologne, Italie, July 2002.

[14] L. Kahloul, K. Barkaoui et Z. Sahnoun, *Using AUML to derive formal modelling agents interactions* , In 3rd ACS/IEEE Int'l. Conf on Computer Systems and Applications, pp. 109-116, IEEE Computer Society Press, 2005. (ref. CEDRIC 1384).

[15] *CPN tools homepage*. Retrieved May 10, 2012, from *cpn-tools.org/*.

[16] F. D. J. Bowden, *A brief survey and synthesis of the roles of time in Petri nets*, Mathematical and Computer Modelling 31 (2000) pp. 55-68.

[17] K. Jensen and L. M. Kristensen, *Coloured Petri Nets - Modelling and Validation of Concurrent Systems*, Springer, July 2009.

[18] W. M. P van der Aalst, *Interval timed coloured petri nets and their analysis*, In Application and Theory of Petri Nets 1993, Proc. 14th International Conference, volume 691, pp. 453-472, Chicago, (USA), 1993. Springer-Verlag, Lecture Notes in Computer Science.

[19] R. G. Smith, *The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver*, IEEE Trans. Computers 29(12): 1104-1113 (1980).

[20] B. Berthomieu, *La méthode des Classes d' Etats pour l'Analyse des Réseaux Temporels - Mise en Oeuvre, Extension à la multi-sensibilisation, Modélisation des Systèmes Réactifs*, In Proc. of MSR'2001, pp. 254-263, Toulouse, France, 2001. Hermes Sciences.

[21] M. A. Piera and G. Music, *Coloured Petri net scheduling models: Timed state space exploration shortages*, Mathematics and computer in simulation 82, (2011), pp. 428-441, Elsevier.

[22] L. Qiaoyun, L. Jiandong, D. Dawei, and K. Lishan, *An extension of contract net protocol with real time constraints*. Wuhan University Journal of Natural Sciences. Wuhan University Journals Press. ISSN:1007-1202 (Print) 1993-4998, Volume 1, Number 2 / juin 1996, pp. 156-162.

[23] N. Dragoni, M. Gaspari, and D. Guidi, *An ACL for specifying Fault-Tolerant protocols*, AI*IA 2005: Advances in Artificial Intelligence 9th Congress of the Italian Association for Artificial Intelligence, Italy 2005 pp. 237-248.