Journal of Computing and Information Technology - CIT 15, 2007, 3, 195–213
doi:10.2498/cit.1000791

Toward Specifying Multimedia Requirements Using a New Time Petri Net Model

Abdelkrim Abdelli and Nadjib Badache

Computer Science Department, University *USTHB* of Algiers, Algeria

In this paper, we define a model dedicated to the specification of multimedia applications called Preemptive Time Petri Nets with synchronizing transitions (*STPTPN*) as an extension of T-time Petri nets where time is associated with transitions. The model is proposed in the general purpose to model a large scale of multimedia requirements. Thus, resource requirement issues are discussed in this paper, and addressed in the model. To deal with, resources are modelled as special places using a new mechanism called *preemptor hyperarc* which lets a transition be *resource strongly − enabled*, *resource − violated* or *resource − violating*. Moreover, two additional mechanisms are considered: A time suspension mechanism that uses inhibitor arcs associated with stopwatches and synchronization mechanisms that allow the simultaneous firing of a set of transitions (called *Rendezvous*), according to different schemes. Compared to other existing models, our model is provided with an adapted semantics, designed to represent clearly and accurately time requirements, as well as the complex resource-preempting mechanisms that are observed in multimedia systems.

Keywords: time Petri net, preemptor hyperarc, synchronizing scheme, stopwatch, inhibitor arc, multimedia requirement

1. Introduction

In recent years, several papers were focused on the development of the interactive multimedia documents based on different standards: *SMIL* [1], *Hitime* [14], *MHEG* [13], etc.

In fact, multimedia documents become a powerful medium of communication integrating different media types such as video, audio, animation, text and image, characterized by real time requirements, as well as specific synchronisation schemes. These constraints are generally difficult to describe and to handle, particularly when building large-scale distributed multimedia architectures. Hence, an important issue of the design of a multimedia system is the definition of its temporal and logical structure which will be used later for enforcing the required synchronizations when processing its different components. The capacity to define this structure entails the availability of an adequate model capable to specify accurately and clearly all the system requirements.

Given the complexity of multimedia/hyper - media authoring in terms of spatial and temporal synchronizations, formal methods have been widely used for specification, validation and testing multimedia requirements, such as media synchronizations, user interaction, resource allocation, etc. For instance, Petri nets [12] and algebraic specifications have proved to be of very high interest. Thus, extending these formalisms to handle multimedia and hypermedia requirements has been investigated through different works [3, 4, 6, 11, 16, 17].

Furthermore, interpreting these models with spatial parameters and resource descriptors, make it possible to derive executable applications for specific platforms [15]. More particularly, timed extensions of Petri nets [5, 6, 8, 9, 11, 16, 18] are powerful models for real time systems, mainly because they can model both the concurrency and the real time constraints in a natural way. They offer a graphical syntax for user-friendly authoring and formal verification techniques for checking documents against temporal inconsistencies and synchronization errors. However, to be applicable to the design of large

and complex multimedia hypermedia systems, the model should be able to satisfy some important requirements [2, 3, 4]:

- The capacity to describe hierarchically the document structures as well as the synchronizations levels.
- The capacity to express incomplete timing specifications.
- The capacity to express the user interaction-based synchronizations.
- The capacity to express a media object as logical unit, abstracting its content, without hindering however the capacity of expressing temporal relations that refer to parts of this media object.
- The capacity to express a wide range of synchronization patterns.
- The availability of a formal semantics together with verification techniques for detecting potential inconsistencies in large and complex multimedia documents.
- And finally, the simplicity and intuitive nature of the modelling concepts provided to the users.

Taking into account the previous requirements, we propose in this paper a new model that handles, in addition to logical, spatial and time requirements of multimedia systems, two specific issues that are not addressed in other existing models:

a) The model should be able to design the media resource demands providing thereto an adapted resource preempting mechanism, which resolves resource conflicts by means of a given priority order. In such systems, some resources (e.g. audio device, layout) are not mandatory to perform a media presentation. However, the unavailability of a resource handicaps the presentation since the latter will be devoid of information vector, affecting thus the information perceived by the user. On the other hand, a media may violate the resource required for its presentation, if held by a media having less priority, even if the latter has not finished its own presentation.

b) The model should consider the semantics introduced, for instance, in the *SMIL* language [1]. We think especially of time suspension mechanism encountered when performing anchor objects, where a media presentation can be suspended for an indefinite time and resumed afterwards.

Consequently, in this paper we define an extension of time Petri nets [16] called *STPTPN* (*Preemptive time Petri nets with synchronizing transitions*) dedicated to modelling multimedia requirements, using *TPN* model extended with specific mechanisms and provided with adapted semantics:

— Taking into account the issue a), the resources are modelled as special places, while adapting the firing-rule semantics. Thus, since the resources availability does not condition the transitions firing (i.e., the transition may fire even if the resources are not available), we introduce three special events when firing a transition t:

- The *strong event* noted t, denoting that t is fired while getting all the resources required;
- A *violated event*, noted t^* denoting that t is fired while missing one of the resources required; and
- A *violator event* noted ${}^*t$, meaning that t is fired when violating the resources from other less-priority transitions.

Thus, a new mechanism, which we call *preemptor hyperarc*, is introduced to decide what event must be generated when firing a transition.

— In our model, a stopwatch [10] is associated with each transition, allowing to start, stop, and resume its time passage, by using classical arcs and inhibitor arcs [20]. This additional mechanism models the time suspension of a media presentation, as mentioned in b).

— Furthermore, to model the different synchronizing schemes as those defined in *TSPN* [5], we consider the simultaneous firing of a set of transitions (which we call *rendezvous*) according to different synchronizing rules.

The remainder of this paper is organized as follows. In Section 2, we survey some Petri net extensions. We focus particularly on works which have introduced mechanisms that are convenient for the modelling of multimedia requirements. In Section 3, we investigate a *TPN* extension model dedicated to the specification of a large-scale of multimedia applications. Consequently, taking into account time and synchronization requirements, as well as the resources

demands, we define progressively a general framework by using preemption, inhibition and synchronization mechanisms towards a correct design of the system requirements. Section 4 presents the syntax and the formal semantics of our proposed model, called *STPTPN* (Synchronized Transitions Preemptive Time Petri Net). Section 5 shows how the model can be used to specify *SMIL* requirements. At last, Section 6 compares our model with other existing models.

2. An Overview of some Petri Net Models

Petri nets [12] are powerful models, mainly because they can model both the concurrency and parallelism in a natural way. A Petri net is characterized by a set of places representing the system resources as well as the conditions that govern the occurrence of the system events. Each place is represented by a circle and characterized by a number of tokens modelling the number of available resources of a given state also called Marking. On the other hand, the system events are modelled by a set of transitions represented generally by rectangular boxes: the occurrence of an event, namely, firing a transition, requires the availability of a number of tokens in each entry place (i.e., the number of tokens is stamped on the oriented arc linking the place to the transition). We say then that the transition is enabled or firable for the current marking. After firing a transition, the tokens in the entry places are consumed while tokens on each exit place (the number of which is stamped on the arc linking the transition to the place), are produced (see Figure 1.a). The main timed extension of Petri net is Time Petri Nets (*TPN*) [16].

An interval is associated therein, with each transition, in order to allow expressing either delay duration, or event duration. An interval $[a, b]$ means that a transition t can fire after $a + \delta$ and before $b + \delta$ (where δ represents t enabling date), providing that t remains continuously enabled before, and no firing interval of an enabled transition has been overtaken (see Figure 1.e).

Furthermore, many Petri nets extensions have been investigated to resolve non determinism and conflict when different transitions are enabled and conflicting for the same marking. For example, the priority net [19] defines a binary priority relation (which is assumed to be non-reflexive and anti-symmetric) on conflicting transitions. In this model, an enabled transition can fire only if no other transition with higher priority is enabled for the same marking. Notice that in Figure 1.b, we represent the priority relation by an oriented arc going from the transition of lower priority to the transition of higher priority.

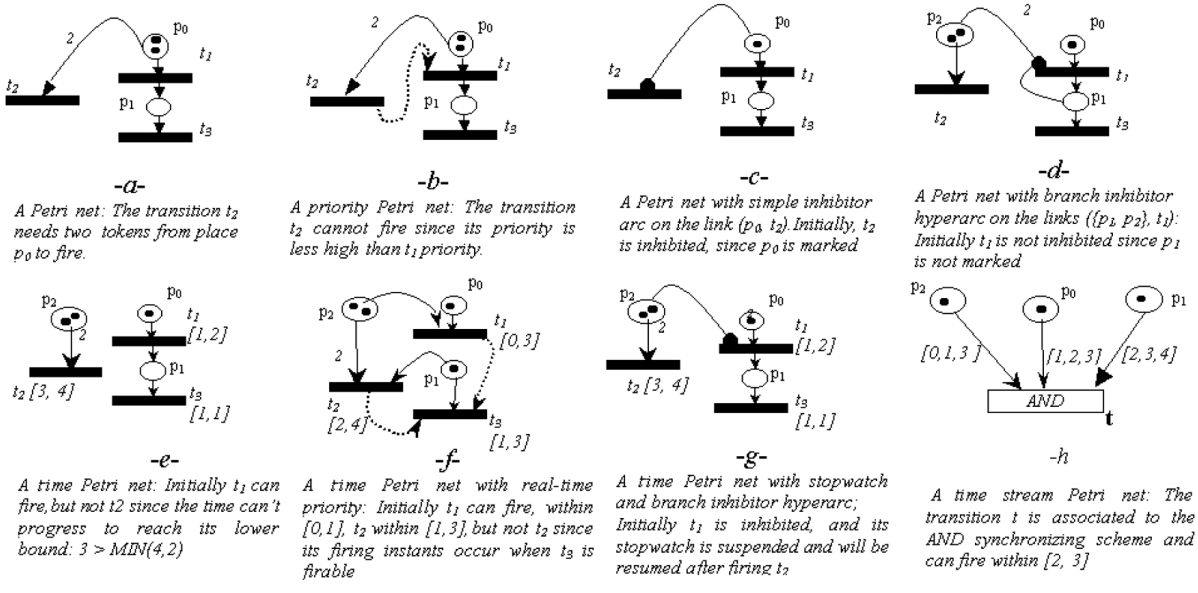

Figure 1. Petri nets extended to time and some mechanisms.

Besides, to avoid inter-blocking and famine situations due to the use of priority mechanism, *Abdelli at al.* proposed in [22] a mechanism so that the priority assigned to each transition is considered in real time, namely, an enabled transition can fire at instant δ if no other enabled transition with higher priority can fire at the same instant (see Figure 1.f).

For the same purpose, Petri nets with inhibitor arcs were introduced by *Agerwela* [20]. The standard execution rule for inhibitor arcs says that an inhibitor arc between condition place p and an event transition t means that t can only fire if t is unmarked. Recently, authors of [21] introduced inhibitor hyperarcs linking a set of places $p_1, ..., p_k$ to a transition t, the semantics of which means that t is unfirable if all the places are marked (see Figure 1.(c,d)). Inhibitor hyperarcs do not increase the expressivity of the model compared to inhibitor arcs, but, nonetheless, greatly improve the convenience of the modelling. To improve the expressivity of time Petri nets, additional mechanisms [9, 18] were introduced with the general purpose of modelling scheduling problems where clocks are associated with tasks and which can be suspended, resumed or reset. This clock mechanism named stopwatch has been applied on time Petri nets in [10, 18] so that each transition's clock can be reset, stopped and started by using classical arcs and branch inhibitor hyperarcs [10] (see Figure 1.g). Moreover, some authors have investigated extension of *TPN* semantics to different synchronizing schemes [5, 6, 11]. In this context, *TSPN* (*Time Stream Petri Net*) model [6] has been proposed for modelling complex parallel timed scenarios, and used particularly in modelling multimedia systems. This model extends timed link Petri net [8] to the seven synchronizing schemes defined in *OCPN* [11] so that processes are represented with places and their temporal characteristics are represented as arcs labeled with temporal intervals called temporal validity intervals *(TVI)*. These are defined as a 3-tuple $[x, n, y]$, where x, n and y are, respectively, the minimum, nominal and maximum allowed durations of the related process. There are three fundamental synchronizing strategies in *TSPN* (see Figure 5.b) which entail nine firing rules obtained from a consistent and complete combination of the absolute temporal validity intervals of arcs associated with a marked transition (see Figure 1.h).

3. Modelling Multimedia Requirements Using TPN

We investigate thereafter how to use *TPN* model for the specification of multimedia requirements. First we show how to model basic media object and then, progressively, we consider complex presentations wherein *TPN* semantics is extended to preemptor hyperarcs, inhibitor arcs, stopwatches, and finally to synchronization like mechanisms.

3.1. Modelling a Basic Media Object

A basic media object denoted O is characterized usually by two events, its starting event $B(O)$ and its ending event $E(O)$. Besides, these events can be temporally constrained according to time requirements linked to the media. Thus, a basic characterization of a media object O can be specified by the *TPN* of Figure 2.a, where $[MinS,\ MaxS]$ represents the time delay before the start of the presentation of O, and $[Mind,\ Maxd]$ delimits the total duration of one occurrence of the media presentation. Notice that the time used here is relative.

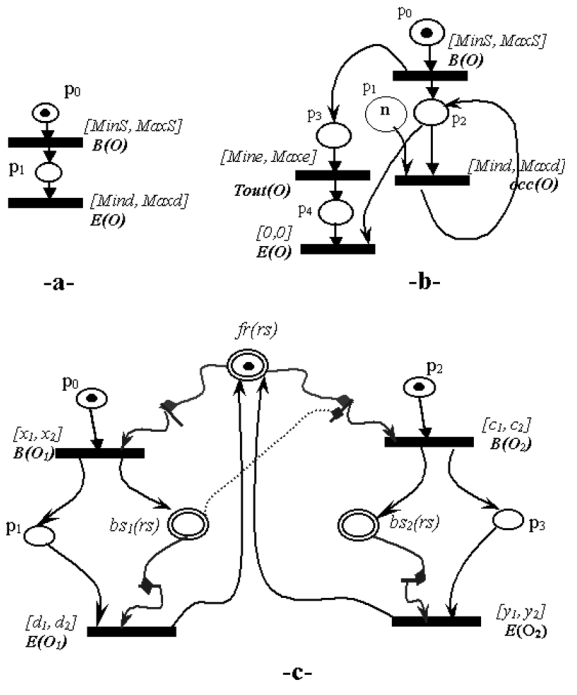

Figure 2. Modelling media presentation requirements.

However, a complete characterization of media object requirements entails, for instance, to consider additional occurrences of the same presentation. Within this intention, Figure 2.b shows how this specification can be modelled using a *TPN* model. We use the place p_1 to represent the number of repetitions to perform, according to the number of tokens (n) contained in place p_1.

Once fired, the transition labelled with the event "$occ(O)$" denotes one occurrence among the repetitions required for the presentation of O. The interval $[Mine, Maxe]$ denotes the time constraints within which all repetitions have to be performed. Thus, when this timeout is met (i.e., firing $tout(O)$), the presentation of O is ended (i.e., the token in place p_2 is definitely consumed when firing $E(O)$).

On the other hand, modelling multimedia applications implies taking into account, in addition to their time requirements, their resource demands which are quite different. For instance, audio presentations need the audio device to be user-perceived, whereas video presentations necessitate the availability of the layout to be displayed. However, the availability of such resources is not mandatory to perform the presentation. The latter can occur, even if some resources are not available, but without providing all the information vectors carried by the media. For instance, a video can be shown in its target layout, but its accompanying sound might not be delivered if the sound device is unavailable.

For example, in *SMIL* presentations [1], the layout is decomposed in different areas called "*regions*", a media presentation (*text*, *image* and *video*) has to be displayed in one of the defined regions. However, several media might be conflicting for the same region. In this situation, the priority is given, for instance by Real one player [7], to the last inserted media tag in the *SMIL* document. By the way, for such presentations, the layout seems to be the main resource liable to conflicting situations. Hence, specifying the layout time constraints in the model has to be performed accurately in order to achieve a complete specification of the system requirements.

Therefore, regarding media presentation requirements and for the sake of simplicity, we assume that, in the model, a presentation can require only one resource. Notice that associating with each media more than one resource in the model yields a complex modelling which might lead to incoherent specifications.

We model hereafter a resource (a region, for instance), as a specific place represented with a double line circle. More precisely, the allocation scheme of a resource *rs* will be modelled as a pair $(fr(rs),\ bs_i(rs)\ i = 1..p)$, where $fr(rs)$ is a resource place denoting, when marked, that *rs* is free; and $bs_i(rs)\ i = 1..p$ is a set of resource places, so that each place $bs_i(rs)$ is associated with a media presentation O_i (requiring *rs*) and denotes, once marked, that *rs* is held by the presentation O_i.

For instance, Figure 2.c models two media, O_1 and O_2, which need the same resource *rs* for their presentations. Consequently, special arcs linking the place $fr(rs)$ to transitions $B(O_1)$ and $B(O_2)$ specify that the start of both presentations O_1 and O_2 needs the availability of the resource *rs*.

Therefore, we assume that the semantics of a resource place is different from one of the standard place in the sense that, to be enabled, a transition does not need the availability of the required token in its entry resource place (i.e., only tokens in the standard places are required). Note that this working scheme is inspired by players acting like *Real one player* and *Ambulant player* [7, 23], and encountered in particular when playing *SMIL* presentations [1].

Besides, based on our survey made on player functioning, we notice that the time constraints imposed on a media presentation are observed from the date when the media start-event occurs, even if the resources required are unavailable at the beginning.

Therefore, a media O_1, which is performing its presentation, might get its resource withdrawn from a media presentation with higher priority O_2. Thus, a media is not guaranteed to hold the resource during its entire presentation.

However, if a media presentation has been preempted or could not get its resource at the start, it can't recover the required resource afterwards, even if the latter is released before the media presentation ends. So, according to these observations, and in order to perform correct modelling and analysis of the system, we consider

different event types when firing a transition:

— A *strong event* noted t denoting that the transition t is fired while getting the required resource (i.e., the resource is free when firing t).

— Two special events called violation events: The first event noted t^* (which we call *"the violated event"*) defining the situation where the transition t is fired when the resource is not free (i.e., the media can't get its required resource held by a higher priority presentation). The second event noted ${}^*t$ (which we call "the *violator event*") is generated when a transition is fired when the resource is not free, while violating the resource from a lesser priority transition (i.e., the media preempts a presentation of lower priority, to get its required resource).

To model the previous behaviours, we define a new mechanism which we call *"preemptor hyperarc"*. The latter allows linking, in the input, a set of resource places to one transition. It uses two types of simple arcs: a *strong arc* represented as a continuous oriented arc; and a *violator arc* represented as a doted oriented arc. Hence, being given a preemptor hyperarc, the place linked through its *violator arc* is called a "*violator resource place*", whereas the place linked through its strong arc is called a "*strong resource place*". For instance, in the example given in Figure 2.c, the preemptor arc linked to the transition $B(O_2)$, is connected to the strong resource place $fr(rs)$ and to the violator resource place $bs_1(rs)$.

Consequently, a strong event t is generated when firing a transition t if its strong resource places are all marked. Otherwise, the violated event t^* is generated if both strong and violator resource places are unmarked. Finally, the violator event ${}^*t$ is generated if no strong resource place is marked, and there is at least one violator resource place marked. By the way, we can notice that the occurrences of strong events as the violated event deal with the input resource places of the linked transition, whereas the occurrences of violator events deal besides, with the output resource places of transitions of lesser priority. Hence, we assume that each media O has a fixed priority denoted by $Pr(O)$, and therefore all transitions involved in the modelling of the same object hold the same priority. We denote therefore by $Pr(t)$ the priority assigned to the transition t. Furthermore, the notation $t_1\ (rs) \rightarrow t_2$ denotes that the transition t_1 has got its resource rs violated by the transition t_2, and this necessarily implies that $Pr(t_1) \preceq Pr(t_2)$.

Note: According to the hypothesis made previously that a media presentation may require at most one resource, we assume that each preemptor hyperarc can contain at most one strong arc and each resource place contains at most one token. Moreover, to be coherent with the previous hypothesis, we admit that an occurrence of a *violated event* t^* can't produce a token in its output resource place since the resource is no longer held by the fired transition.

For example, Figure 2.c models two presentations O_1 and O_2 in conflict for the resource rs, whereon we assume that O_2 has priority (i.e., $Pr(O_1) \preceq Pr(O_2)$). To model the resource allocation scheme, we use four preemptor hyperarcs: Two of them are linked to transitions $B(O_1)$ and $B(O_2)$ meaning that both media O_1 and O_2 need the availability of the resource rs to start their presentations (i.e., a strong event is generated if the place $fr(rs)$ is marked). Otherwise, if the resource rs is unavailable for one of the two presentations, (the resource place $fr(rs)$ is unmarked and one of the resource places $bs_1(rs)$ or $bs_2(rs)$ is marked), then only O_2, which has priority (modelled using the violator arc linked to the resource place $bs_1(rs)$), might withdraw rs from O_1, violating thus the resource rs: $E(O_1)(rs) \rightarrow B(O_2)$ (i.e., the *violator event* ${}^*B(O_2)$ is generated since the violator resource place $bs_1(rs)$ is marked). The other two preemptor hyperarcs are linked to the transitions $E(O_1)$ and $E(O_2)$ denoting that the presentations of O_1 and O_2 need to hold the resource rs throughout the presentation. Hence, if the resource place $bs_1(rs)$ is unmarked when firing $E(O_1)$, a violated event is generated, denoting that the resource rs has been withheld from O_1 by O_2 during its presentation. Notice that in the example in Figure 2.c, only $E(O_1)$ can be violated since O_2 has priority. To model the case where no presentation has priority, (i.e., $Pr(O_1) = Pr(O_2)$) we need to add a violator arc going from the resource place $bs_2(rs)$ to the transition $B(O_2)$, namely both presentations can be preempted, and we can have initially either $E(O_1)(rs) \rightarrow B(O_2)$ or $E(O_2)(rs) \rightarrow B(O_1)$.

Note: For a given marking, only one place among $fr(rs), bs_1(rs), bs_2(rs)$ should be marked.

To highlight the semantics of the proposed mechanism, we give hereafter all possible scenarios in terms of events sequence obtained when performing the presentations O_1 and O_2 as modelled in Figure 2.c:

— $B(O_1) \rightarrow E(O_1) \rightarrow B(O_2) \rightarrow E(O_2)$ and $B(O_2) \rightarrow E(O_2) \rightarrow B(O_1) \rightarrow E(O_1)$: both scenarios model the case where the presentations O_1 and O_2 are not overlapping according to their time requirements (i.e., the presentations are performed in sequence). In this configuration, transitions are fired while generating strong events, since the strong resource places are marked (see Figure 3.a).

— $B(O_1) \rightarrow {}^*B(O_2) \rightarrow E(O_1)^* \rightarrow E(O_2)$: In this case, O_1 starts its presentation first (i.e. the strong event $B(O_1)$ is generated since $fr(rs)$ is marked) then, O_2 needs to violate the resource of O_1 to start its own presentation (i.e., the violator event ${}^*B(O_2)$ is generated since $fr(rs)$ is unmarked while $bs_1(rs)$ is). However, to express that O_1 has got its resource violated during its presentation, the violated event $E(O_1)^*$ is generated since $bs_1(rs)$ is no longer marked $E(O_1)(rs) \rightarrow B(O_2)$. Finally, firing $E(O_2)$ generates a strong event since $bs_2(rs)$ was marked after firing $B(O_2)$. Further, $fr(rs)$ will be marked again only after firing $B(O_2)$ because firing $E(O_1)$ as a violated event does not produce token in $fr(rs)$ (see Figure 3.b).

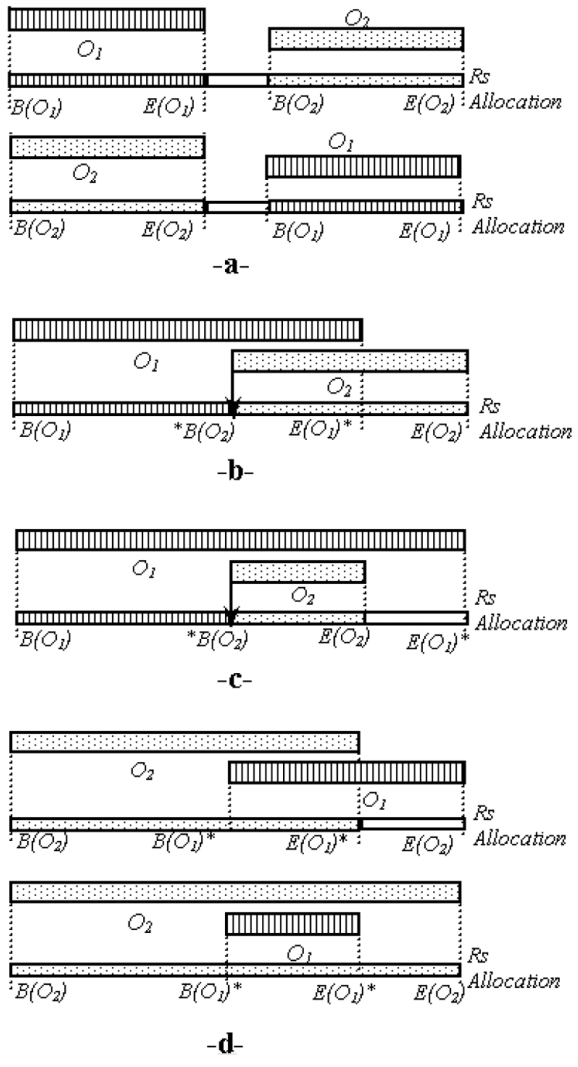

Figure 3. Resource allocation schemes.

— $B(O_1) \rightarrow {}^*B(O_2) \rightarrow E(O_2) \rightarrow E(O_1)^*$: This scenario is the same as the previous one, but we consider here that O_2 finishes first. As $bs_1(rs)$ becomes unmarked after firing ${}^*B(O_2)$; the generated event $E(O_1)^*$ denotes that the media O_1 has got its resource violated by O_2 during its presentation (see Figure 3.c).

— $B(O_2) \rightarrow B(O_1)^* \rightarrow E(O_1)^* \rightarrow E(O_2)$ and $B(O_2) \rightarrow B(O_1)^* \rightarrow E(O_2) \rightarrow E(O_1)^*$: Both scenarios describe the case where O_2 starts first. Consequently, the media O_1 can't get the resource required for its presentation i.e., the violated events $B(O_1)^*$ and $E(O_1)^*$ are both generated since $fr(rs)$ and $bs_1(rs)$ are unmarked denoting that O_1 could not get the resource rs from the start and therefore for the whole presentation (see Figure 3.d).

Notice that the preemption mechanism can be extended for more than two objects. For instance, let O_1, O_2,..., O_p be a set of media objects that are conflicting for the same resource rs, and let's assume that $Pr(O_1) \ldots \prec Pr(O_i) \ldots \prec Pr(O_p)$. Hence, to model a general preempting process, we need to consider first the resource allocation scheme given by $(fr(rs), bs_i(rs) i = 1..p)$ where $fr(rs)$ is marked initially and connected as an input to all transitions $B(O_i)$ $(i = 1..p)$ using strong arcs, denoting that the objects O_i $i = 1..p$ need the availability of rs to be performed. Therefore, each resource place $bs_i(rs)$ is connected as an output to transition $B(O_i)$ using standard arc, and as an input to $E(O_i)$ using a strong arc. Thus, if $bs_i(rs)$ is marked, it denotes that the resource rs is held by the object O_i. Besides, to model the preemption, we need to link each transition $B(O_j)$ $j = 1..p$ to all resource places $bs_i(rs)$ such $i \prec j$, by using violator arcs. Each

violator arc gives the right to $B(O_j)$ to violate transitions of lesser priority, when the required resource rs is held by one of them, namely, when there is one among the resources places $\{bs_i(rs)\ i = 1..j-1$ which is marked. Otherwise, if the resource is held by a higher priority object $(i \succ j)$, $B(O_j)$ cannot get the resource since there is no violator arc linking the resource place $bs_i(rs)$ to transition $B(O_j)$.

3.2. Modelling a Link Object

A link object is an anchor associated with the presentation of a media object. Therefore, an interval $[MinA, MaxA]$ delimiting the time constraints within which the anchor must be available can be specified. Thus, if the user "*may click*$''$ on the Anchor, the presentation targeted by the link is started and the behavior of the old presentation can take different courses [1]:

- The presentation is stopped and replaced by the linked one.
- The presentation continues to be shown parallelly with the new one, but in a different context.
- The presentation is paused and will be resumed once the new presentation ends.

The first case can be modelled as in Figure 4.a, where the current presentation O is linked to a presentation O'' through an anchor, which is available after starting O, during the time interval $[MinA, MaxA]$. The transition *Anc* models the user interaction event, which *may* fire only after *MinA*. Hence, to model the weak semantics of this event, the time interval associated here with the transition *Anc* is $[MinA, \infty]$. This is because the occurrence of the event *Anc* is not mandatory and may not occur. Thus, firing the transition *Anc* consumes the token in place p_1, implying to stop O and to start the new presentation O''. Otherwise, if the user interaction could not occur before *MaxA*, the event transition $Tout(Anc)$ denoting the timeout linked to the event *Anc* is fired, thus removing the availability of the latter (the anchor is no longer displayed).

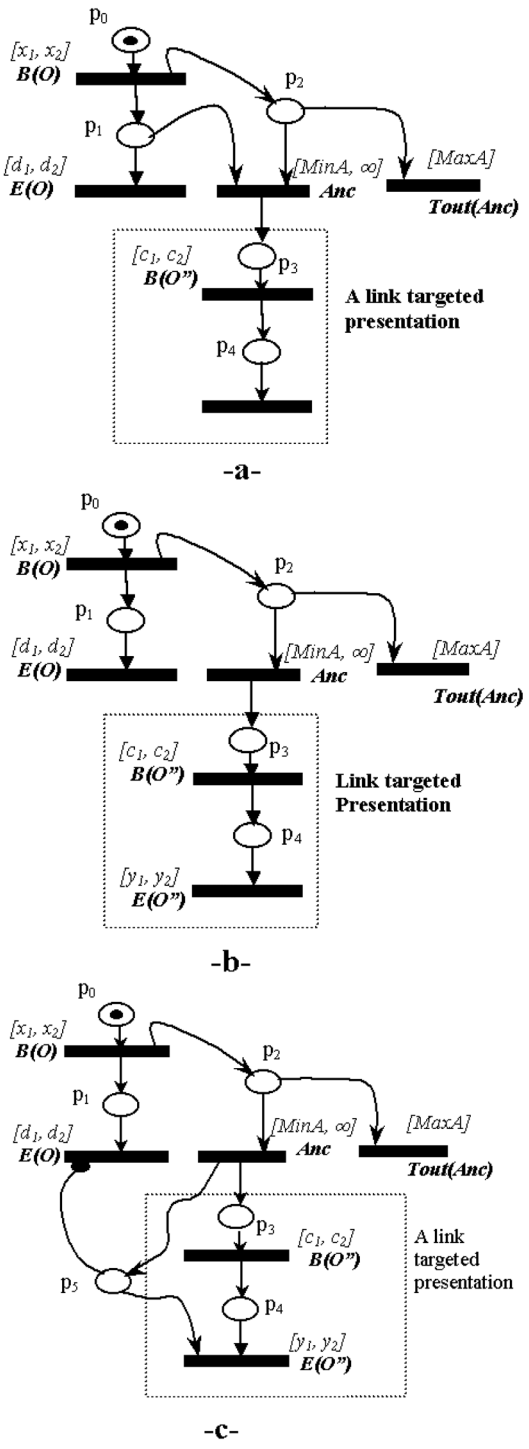

Figure 4. Modelling a link object.

Note: According to *TPN* semantics, when assigning a strong interval $[MinA, MaxA]$ to transition *Anc* the latter is forced to fire when the progression of time reaches the upper bound *MaxA*. This semantics is inconsistent with the fact that the user interaction is not mandatory to occur. This is why we use the transitions *Anc* and $Tout(Anc)$ to model this weak semantics.

The second case can be modelled as the first one, but while removing the arc linking the place p_1 to the transition *Anc* (see Figure 4.b).

Finally, the third case is more complex since it implies the use of a mechanism which has the capacity to stop the time progression for some transitions and to resume it afterwards. To deal with it, we consider the stopwatch mechanism [10] which associates with each transition in the model a clock which could be started, stopped, and reset. Therefore, as in [9], inhibitor arc mechanism is used to decide about the action to apply on the transition's clock. This configuration can be modelled as in Figure 4.c, where the place p_5 is added to trigger the inhibition of $E(O)$, once marked. The inhibition of $E(O)$ provokes the time suspension of its stopwatch, which will be resumed once the transition $E(O'')$ is fired (i.e., the firing of $E(O'')$ will consume the token in the place p_5, thus removing the inhibition of $E(O)$).

3.3. Modelling Synchronisation Requirements

In addition to previous requirements, the model should be able to handle media presentations that require a synchronizing requirement. Thus, events' synchronization (which we call *rendezvous*) is performed when firing simultaneously their related transitions according to a predefined scheme. For instance, *Senac et al* [5, 6] have defined different synchronizing schemes between several actions which occur in parallel. This proposition, which is adopted in our model, offers three basic synchronization strategies (see Figure 5.b):

- Dynamic synchronization strategies *And, Weak-And, And-Master* driven by the latest transition that gets its minimum bound.
- Dynamic synchronization strategies, *Or, Strong-Or, Or-Master* driven by the earliest transition that gets its minimum bound.
- Static synchronization strategies *Master, Strong-Master, Weak-Master* driven by a selected transition also called the master transition.

For example, Figure 5.a shows three media presentations O, O' and O'' which have to synchronize as follows: The start of both O'' and O' has to be synchronized according to the *OR* scheme, which means that the transition between $B(O')$ and $B(O'')$ which fires first, provokes the firing of both, even if the time constraints of the other one are not satisfied. On the other hand, it is assumed that the end of the presentations O and O' must occur in synchronous manner, which means that $E(O)$ fires only if $E(O')$ can fire on the same date.

Note: Only the transition which is not inhibited can synchronize. Hence, a rendezvous which gets one of its transitions inhibited cannot fire. Therefore, a rendezvous can occur even if one of its transitions is violated, since the resource availability is not mandatory to perform the event synchronization.

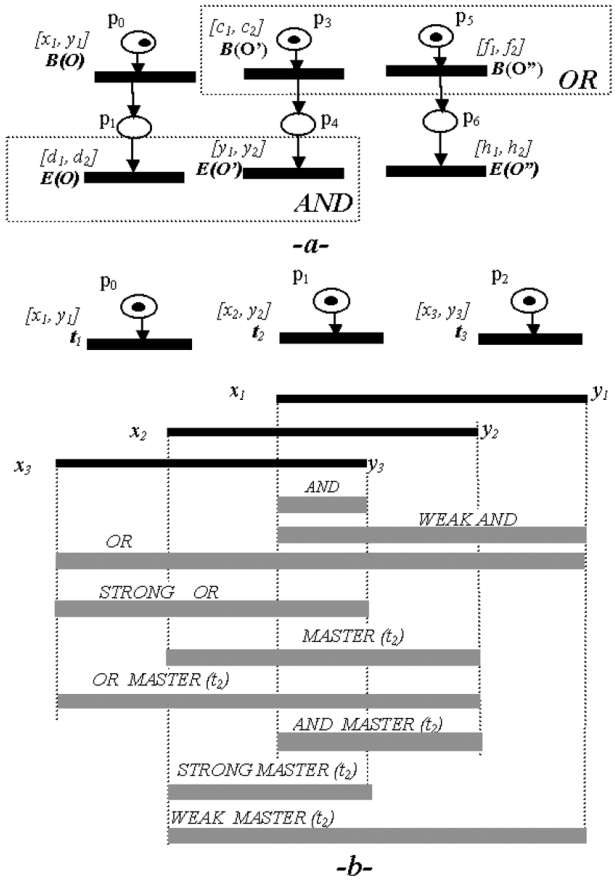

Figure 5. Synchronization schemes.

However, taking into account that transitions synchronizing for a same rendezvous are fired simultaneously, we have to adopt a priority order to determine which transition gets the resource by priority in case of conflict for the same resource. This priority policy follows the

order set previously to resolve the conflict between resource-conflicted transitions when firing a rendezvous.

On the other hand, additional places and transitions may be used to ease the modelling when necessary. For instance, when performing sequential and parallel composition of several objects. In these specific cases, synchronization and time requirements that concern more than one object (a block of objects) is held by these transitions (e.g. transitions *Syn* and $B(par_2)$ in the application example of Section 5.3.

4. Synchronizing Transitions Preemptive Time Petri Net

In this section we introduce a new Petri net model, dedicated to multimedia requirement specification. This proposition is based on the survey presented in the previous section and aims to provide a general framework which can be used to model a large range of multimedia applications. First we present the syntax of the model, and then its formal semantics.

4.1. *STPTPN* Syntax

Definition 4.1. An *STPTPN* model (*synchronizing transitions preemptive time Petri net*) is defined by the tuple $(P, T, Res, B, F, M_0, FI, IH, PH, RDV_s, RDV_m, Pr)$, where:

— P and T are two finite disjoint sets of places and transitions respectively;

— Res is a finite set of special places representing the resources;

— B and F are two functions called backward and forward incidence functions such that: $B : P \times T \longrightarrow N$ and $F : (P \cup Res) \times T \longrightarrow N$. However, we assume that $F(r,t) \in \{0,1\}$ when $r \in Res$ i.e. the resource places are 1-marked:$\sum_{r \in Res} F(r,t) \preceq 1$, namely a transition is linked in its output to at most one resource place using a standard arc;

— M_0 is a function called the initial marking, so that: $M_0 : (P \cup Res) \longrightarrow N$. We assume, therefore, that $M_0(r) \in \{0,1\}$ when $r \in Res$;

— FI is a delay interval mapping function which associates with each transition t of T an interval defined on $Q^+ \times (Q^+ \cup \infty)$. Thus, FI gives the time interval within which the transition t can fire: $FI(t) = [EFT(t), LFT(t)]$ with $EFT(t) \preceq LFT(t)$. The interval is delimited by an earliest firing time $EFT(t)$, and a latest firing time $LFT(t)$, taking values in the set of rational numbers;

— IH is the inhibitor arc function defined from $P \times T \longrightarrow N$, so that there is an inhibitor arc linking the place p to the transition t, if $IH(p,t) \neq 0$;

— PH is a finite set of preemptor hyperarcs: each hyperarc is a tuple (srp, Vrp, t) where $t \in T$ is a transition, *srp* is an input resource place from *Res*, called *strong resource place*, and *Vrp* is a set of input resource places from *Res*, called *violator resource places*, where we assume that $Vrp \cap srp = \emptyset$, and the number of t-input strong resource places is at most equal to one i.e. the number of preemptor hyperarcs linked to each transition is at most equal to one.

— RDV_s denotes a finite set of simple *rendezvous*. A *rendezvous* R_s of RDV_s has the form $(typ, \{t_1, \ldots, t_r\})$, where $\{t_1, \ldots, t_r\}$ is a set of transitions from T and *typ* defines the synchronization rule and takes one of the following values: $typ \in \{And, Weakand, Or, Strong-Or\}$;

— RDV_m is a finite set of *master rendezvous*. A *master rendezvous* R_m is given by $(typ, \{t_m\} \Rightarrow \{t_1, \ldots, t_r\})$ so that $\{t_m\} \cap \{t_1, \ldots, t_r\} = \emptyset$, where $t_1, \ldots, t_r$ and t_m are transitions from T, and *typ* defines the synchronization rule which takes one of the following values: $typ \in \{Master, Or-Master, And-Master, Weak-Master, Strong-Master\}$. The transition t_m is denoted by $MA(R_m)$, and called the *master transition*;

— Pr: is a priority function assigning for each transition a priority level, $Pr : T \longrightarrow N$. This order operates when the transitions synchronizing for a rendezvous are in conflict for same resources. In this case, the defined order determines which transitions get the resources while firing the rendezvous.

Note: A *resource place* is linked to a transition as an input by using a preemptor hyperarc (given by PH) and as an output by using a standard arc (given by F).

4.2. *STPTP* Formal Semantics

The *STPTPN* model progresses at each step by firing a rendezvous, denoting the occurrence of an event's synchronization. Hence, firing a rendezvous is conditioned by its corresponding synchronization rule which entails to fire simultaneously all its transitions, taking therefore into account the time constraints as well as the availability of the required resources, in order to determine what event's types should be generated when performing the rendezvous.

Definition 4.2. Let M be an accessible marking:

— A transition t is said to be *enabled* for the marking M, if and only if the number of tokens in M in each t input standard place is greater or equal to the valuation of the arc between this place and t: $\forall p \in P, B(p,t) \preceq M(p)$. We denote thereafter by $Te(M)$ the set of transitions enabled for M.

— A transition is said to be *inhibited* by the marking M if it is enabled and one of its inhibitor arcs is enabled: $t \in Te(M) \wedge \exists p \in P,\ IH(p,t) \preceq M(p)$. We denote by $Ti(M)$ the set of transitions that are inhibited for M.

— A transition t is said to be *strongly enabled* for the marking M if it is *enabled*, not *inhibited*, and its input strong resource place is marked: $t \in Te(M) \wedge t \notin Ti(M) \wedge (\forall v \in PH,\ v = (srp, Vrp, t),\ M(srp) = 1)$. We denote by $Ts(M)$ the set of transitions strongly enabled for M.

— A transition t is said to be *violated* for the marking M if it is *enabled*, not *inhibited*, and both input strong and violator resource places are unmarked: $t \in Te(M) \wedge t \in Ti(M) \wedge (\forall v \in PH,\ v = (srp, Vrp, t),\ \forall r \in \{srp\} \cup Vrp, M(r) = 0)$. We denote by $Tv(M)$ the set of transitions violated for M.

— A transition t is said to be *violating* or (*preempting*) for the marking M if it is *enabled*, not *inhibited*, its strong resource place is unmarked and at least one of its violator resource places is marked: $t \in Te(M) \wedge t \notin Ti(M) \wedge t \notin Tv(M) \wedge t \notin Ts(M)$ We denote thereafter by $Tp(M)$ the set of violating transitions for M.

Note: We can notice that both strongly enabled and violated transitions are decided by the availability of tokens in their input resource places whereas violating transitions deal in addition to its input resource place with the output resource places linked to transition with lesser priority.

On the other hand, we suppose that the model is a $T-$*safe* net: for any marking, a transition t is at most $1-$*enabled* (A marking cannot enable the same transition several times).

Notation 4.1. Let RDV be the set of *rendezvous* that might fire in the model. This set contains rendezvous from RDV_s and RDV_m, as well as the set of asynchronous rendezvous denoted RDV_a. An asynchronous rendezvous R_a is a single transition of T, which is not involved in any rendezvous of $RDV_s \cup RDV_m$.

Let R be a rendezvous from RDV, we denote thereafter by $Trans(R)$ the set of synchronizing transitions in R.

Definition 4.3. Let M be an accessible marking of an *STPTPN*:

— A rendezvous R is said to be *enabled* for the marking M if all its transitions are enabled for M and we denote by $enable(M)$ the set of all rendezvous of RDV that are enabled for M. $enable(M) = \{R \in RDV \mid Trans(R) \subseteq Te(M)\}$.

— A rendezvous R is said to be *inhibited* for the marking M if it is enabled for M and at least one of its transitions is inhibited for the marking M. We denote by $inhibit(M)$ the set of rendezvous of RDV, that are inhibited for M. $Inihibit(M) = \{R \in RDV \mid R \in enable(M) \wedge (\exists t \in Trans(R),\ t \in Ti(M))\}$.

— A rendezvous R is said to be *strongly enabled* for the marking M if it is enabled and not inhibited for M. We denote by $Senable(M)$ the set of rendezvous that are strongly enabled for M. $Senable(M) = \{R \in RDV \mid R \in enable(M) \wedge R \notin inhibit(M)\}$.

Definition 4.4. (*STPTPN* state). A state of an *STPTPN* is a couple $S = (M, V)$, where M is the current marking, and V is a function associating with each enabled transition, a dynamic interval representing its time constraints: $V(t) = [EFT_s(t), LFT_s(t)]$. The initial state of the model is given by the pair (M_0, V_0), where M_0 is the initial marking, and V_0 associates with each enabled transition, its static firing interval:

$$\forall t \in Te(M_0) \qquad V_0(t) = FI(t).$$

Definition 4.5. (A temporal validity interval of a rendezvous). Let S be an accessible state and R a rendezvous *strongly enabled* for S. We associate with R its temporal validity interval

given by $TVI(R) = [Low_s(R), Up_s(R)]$, where $Low_s(R)$ and $Up_s(R)$ represent the lower bound and the upper bound of R, respectively. The interval $TVI(R)$ determines the interval of potential dates within which R has to fire from state S. Therefore, R can occur from state S at relative date δ if $Low_s(R) \preceq \delta \preceq Up_s(R)$. Where $Low_s(R)$ and $Up_s(R)$ are computed according to the synchronization rule provided by $typ(R)$:

Low_s(R):=

$$\begin{cases} if\ typ(R) \in \{And, Wand, Amas, Async\} then \\ \quad MAX_{t \in\ Trans(R)}\ EFT_s(t) \\ if\ typ(R) \in \{Or,\ Sor,\ Omas\}\ then \\ \quad MIN_{t \in\ Trans(R)}\ EFT_s(t) \\ if\ typ(R) \in \{Mas,\ Smas,\ Wmas\}| \\ \quad \wedge(t_m = MA(R))\ then\ EFT_s(t_m) \end{cases}$$

Up_s(R):=

$$\begin{cases} if\ typ(R) \in \{And,\ Sor,\ Smas\}\ then \\ \quad MIN_{t \in Trans(R)}\ LFT_s(t) \\ if\ typ(R) \in \{Wand,\ Or,\ Wmas\}\ then \\ \quad MAX_{t \in Trans(R)}\ LFT_s(t) \\ if\ typ(R) \in \{Mas,\ Omas,\ Amas\} \\ \quad \wedge(t_m = MA(R))\ then\ LFT_s(t_m) \end{cases} .$$

However, the above formulae can compute three types of intervals according to the time constraints of the transitions that are synchronizing for the rendezvous:

— A temporal validity interval of a rendezvous is *coherent*, if $0 \preceq Low_s(R) \preceq Up_s(R)$ thus providing the interval of dates within which R might occur.

— Otherwise, the computed interval might be *empty* (incoherent) if $0 \preceq Up_s(R) \preceq Low_s(R)$, describing the behavior wherein R cannot fire since its synchronizing scheme could not be performed, being given the time constraints of its synchronizing transitions.

— The interval might be temporally *inconsistent* if $Up_s(R) \prec 0 \preceq Low_s(R)$, denoting the case where a rendezvous has its interval overtaken by the time passage before its enabling. This situation models a blocking state wherein a synchronizing event could not be provided in due time for its rendezvous, leading to a temporal violation. Hence, states modelling such inconsistencies are called temporally inconsistent states, and are determined formally within the space of accessible states, using next definition.

Definition 4.6. (temporally inconsistent state). A state $S = (M, V)$, is said to be *temporally inconsistent* if the upper bound of the temporal validity interval of a strongly enabled rendezvous has been overtaken by the time passage (i.e., its value is negative): $\exists R \in Senable(M) \quad Up_s(R) < 0$.

Contrary to *TPN* where time constraints related to transitions are strong, in *STPTPN* these constraints are weak, namely, a transition is never forced to fire within its interval. It depends on the synchronizing rule of the corresponding rendezvous. However, a rendezvous is forced to occur within its validity interval and firing R implies to have all its transitions enabled. Thus, time passage can overtake the transition interval before the enabling of its related rendezvous. As the temporal validity interval of R is computed from the intervals of its own transitions, the upper bound of the computed *TVI* might be surpassed when R becomes strongly enabled, which yields a time constraint violation represented by the accessible state called *temporally inconsistent state*. Thus, a temporally inconsistent state models a blocking state wherein no rendezvous can fire since it models a time constraints violation, thus denoting the impossibility for the system to progress.

For instance, in the *STPTPN* in Figure 6, the transitions t_2 and t_3 should fire in synchronous manner whereas t_1 evolves asynchronously. Initially, only the asynchronous rendezvous $R_1 = \{t_1\}$ is strongly enabled and can occur within $[3, 4]$. Then all the states accessible are temporally inconsistent since the upper bound of the *TVI* related to rendezvous $R_2 = (And, \{t_3, t_2\})$ is surpassed by the time passage. $TVI\ (R_2) = [Max(1-\delta, 0),\ Min(1-\delta, 3)] = [0, 1-\delta]$ so that $d \in [3, 4]$, which leads us to state that: $1 - \delta < 0$.

Taking into account the previous hypotheses, next we give a necessary and sufficient condition to fire a rendezvous from a temporally consistent state.

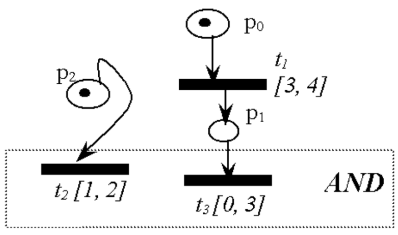

Figure 6. An *STPTPN* with temporally inconsistent states.

Definition 4.7. (firing a rendezvous). A rendezvous R is *firable* from a temporally consistent state $S = (M, V)$ at relative date δ, if: (i) R is strongly enabled for the marking M; (ii) The time passage reaches the lower bound of R; (iii) No upper bound of a strongly enabled rendezvous has been overtaken and (iv) no lower bound of a strongly enabled rendezvous embedding R, is reached: $\exists \delta \in Q^+$ such that:

(i) $R \in Senable(M)$

(ii) $0 \preceq Low_s(R) \preceq \delta$

(iii) $\delta \preceq MIN_{R' \in Senable(M)}\{Up_s(R')\}$

(iv) $\delta \prec MIN^{R' \in Senable(M)}_{Trans(R) \subset Trans(R')}\{Low_s(R)\}$.

The hypothesis (iv) handles the case where several rendezvous are sharing same transitions (namely a transition event can occur according to different synchronisation schemes). Therefore, the model can fire a rendezvous R only if there is not a strongly enabled rendezvous R' embedding R that can fire at the same instant. Otherwise, if the two rendezvous are conflicting for the same transition, then the model fires one of the rendezvous in a non deterministic manner. For instance, let's consider the example given in Figure 7. Initially, t_1, t_2, t_3 and t_4 are strongly enabled, and can fire according to three schemes: $R_1 = (Or, \{t_1, t_2\})$, $R_2 = (AND, \{t_1, t_2, t_3\})$ and $R_3 = (Wand, \{t_3, t_4\})$. Therefore, the computed *TVI* of these rendezvous are: $TVI(R_1) = [0, 5]$, $TVI(R_2) = [2, 2]$ and $TVI(R_3) = [2, 4]$. Hence, R_1 can fire only when R_2 is not firable, namely $[0, 2]$; the priority is given to R_2 since it embeds R_1. However, R_2 and R_3 are conflicting for t_3 and can fire within $[2, 2]$ and $[2, 2]$ respectively.

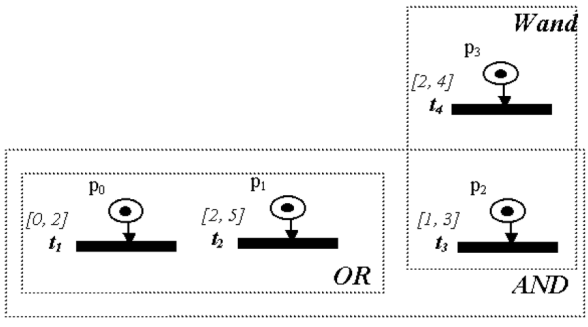

Figure 7. Conflicting rendezvous.

Besides, notice that firing a rendezvous R implies simultaneous firings of all its transitions (i.e., $t \in Trans(R)$), which yields a new accessible state S' where the corresponding marking M' is obtained from M after firing the transitions of $Trans(R)$. As the resource place marking denotes the resource availability, the transitions involved in a firable rendezvous might be in conflict for the current marking; namely, they require the same resources. Therefore, firing a rendezvous implies to resolve the conflict by determining which transition will get the resource needed. The evolution of the marking is carried out by firing the synchronizing transitions sequentially from the highest priority one to the least priority one, according to Pr. Then, the sub marking computed at each step allows to determine the status of the transition (strongly-enabled, violated, violating) and hence, the event to be generated.

Definition 4.8. Let $S = (M, V)$ be an accessible temporally consistent state. $S[(L(R), \delta) > S'$ denotes the firing of the rendezvous R from state S at relative date δ, so that $S' = (M', V')$ is the state accessible, and $L(R)$ is the generated events-label:

— M' is computed from M, after firing the transitions of $Trans(R)$ following their priority order given by the function Pr, starting from the highest priority transition to the lowest priority one.

Let $Trans(R) = \{t_1, \ldots, t_n\}$ where, $Pr(t_1) \succeq Pr(t_2) \ldots \succeq Pr(t_n)$, and we denote by:

$$M'_0[event(t_1)] > M'_1 \ldots M'_{n-1}[event(t_n)] > M'_n$$

the sequence of accessible markings after firing sequentially the transitions $t_1, \ldots, t_n$, where $M = M'_0$ and $M' = M'_n$. Therefore, $event(t_i)$ represents the event induced when firing the transition t_i, and could be: a standard event noted t if the transition is *strongly enabled* for M_{i-1}; a *violated event* noted t^* if the transition is violated for M_{i-1}; a *violator event* noted ${}^*t$ if the transition is violating for M_{i-1}:

$$\begin{cases} events(t_i) := \text{“}t_i\text{”} & if\ t_i \in Ts(M_{i-1}) \\ events(t_i) := \text{“}t_i^*\text{”} & if\ t_i \in Tv(M_{i-1}) \\ events(t_i) := \text{“}{}^*t_i\text{”} & if\ t_i \in Tp(M_{i-1}) \end{cases}.$$

Each marking M'_i is computed from M'_{i-1}, as a standard way for standard places, and when considering the status of each fired transition for resource places. We assume that the number of tokens in each resource place is bounded by one and that a violated transition cannot produce tokens in its output resource places:

$\forall p \in P$,
$\quad M'_i(p) := M'_{i-1}(p) - B(p, t_i) + F(p, t_i)$
$\forall r \in Res$,

If $F(r,t_i) \neq 0 \vee \exists\,(srp, Vrp, t_i) \in PH$, $r \in \{srp\} \cup Vrp$
then

$$\begin{cases} If\ t_i \in Ts(M'_{i-1})\ then\ M'_i(r) := F(r,t) \\ If\ t_i \in Tv(M'_{i-1})\ then\ M'_i(r) := 0 \\ If\ t_i \in Tp(M'_{i-1})\ then\ M'_i(r) := F(r,t) \end{cases}$$

else
$M'_i(r) = M'_{i-1}(r)$.

— The function V' is computed in three steps:

a) Remove the transitions that are disabled for M'; these disabled transitions are those enabled by M and not enabled by M', or those enabled for both markings while being in conflict with transitions of rendezvous R.

b) For each transition t that was enabled for M and remains enabled in M', two cases can follow:

* If t is not inhibited for M, then shift $V(t)$ by the value δ towards the origin of time, and truncate $V(t)$ on its lower bound when necessary to non negative value:
$V'(t) = [Max(0, EFT_s(t)-\delta),\ LFT_s(t)-\delta]$
* If t is inhibited for the marking M, then its firing interval remains the same: $V'(t) = V(t)$.

c) Introduce the transitions that are newly enabled for M' by assigning their static firing intervals: $V'(t) = FI(t)$.

Note: Notice that the upper bound of $V'(t)$ is not truncated to a non negative value, thus allowing to detect temporally inconsistent states.

— The label $L(R)$ is given by the set of events generated when firing the transitions synchronizing in R: $L(R) = \{event(t) \mid t \in Trans(R)\}$.

5. Application

The *SMIL* language, which has become the dominant reference for the characterization of multimedia presentations, suggested several works, in particular those related to the study of the consistency of its documents [28]. Algorithms based on formal methods were integrated into *SMIL* authoring tools and *SMIL* players too, in order to control the consistency of their presentations.

Within this context, in this section we show how to specify *SMIL* document requirements thanks to the *STPTPN* model. The resource requirements are particularly taken into account, emphasizing a new cause of inconsistency which has not been dealt with in existing approaches that perform *SMIL* document consistency checking.

5.1. SMIL Language

SMIL (Synchronized Multimedia Integration Language) [1] is a declarative language based on *XML* describing the spatial and the temporal aspects of different media-object presentations. A basic object can be an *image*, a *text*, a *video* or an *audio clip*. Each object is declared using its associated tag, and it is mainly characterized by its own parameters like its identification *Id*, its *URL*, its duration, its start, its end, and its target display area (region), etc.

Constructor tags are also provided allowing to compose together different presentations according to various synchronization schemes (e.g. the parallel tag $< par >$ or the sequential tag $< seq >$). The non-determinism is introduced through two link tags. The tag *a*, which specifies a *URL* link within a presentation, and the tag *Anchor*, which is similar to *a*, but allows characterizing the dimension of its target layout as well as the display time constraints thereto.

Thereafter, we associate with each *SMIL* element noted O, two events, its starting event $B(O)$ and its ending event $E(O)$. Notice that the start and the end parameters of an object O are expressed as clock values or as event values. An event value expresses a synchronization relation between two events. An event value has the form $Id(C)\ (e)$, where e specifies either an event or a clock value: e.g. $End :=$ "$id(C)$ (3)" means that O ends "3" time units after C began, whereas $End :=$ "$id(C)$ (end)" means that O finishes when C ends. The previous synchronization relation is a *master* one, namely the occurrence of the first event (named the *slave*) is decided by the occurrence of the second event (named the *master*). However, the master event can occur even if the slave event is unavailable. Hence, to model this mechanism with an *STPTPN*, we need to consider two rendezvous, the first one will model the master synchronization between the two events and the second one will model the asynchronous evolution of the master event, (i.e., in case where the slave event cannot occur).

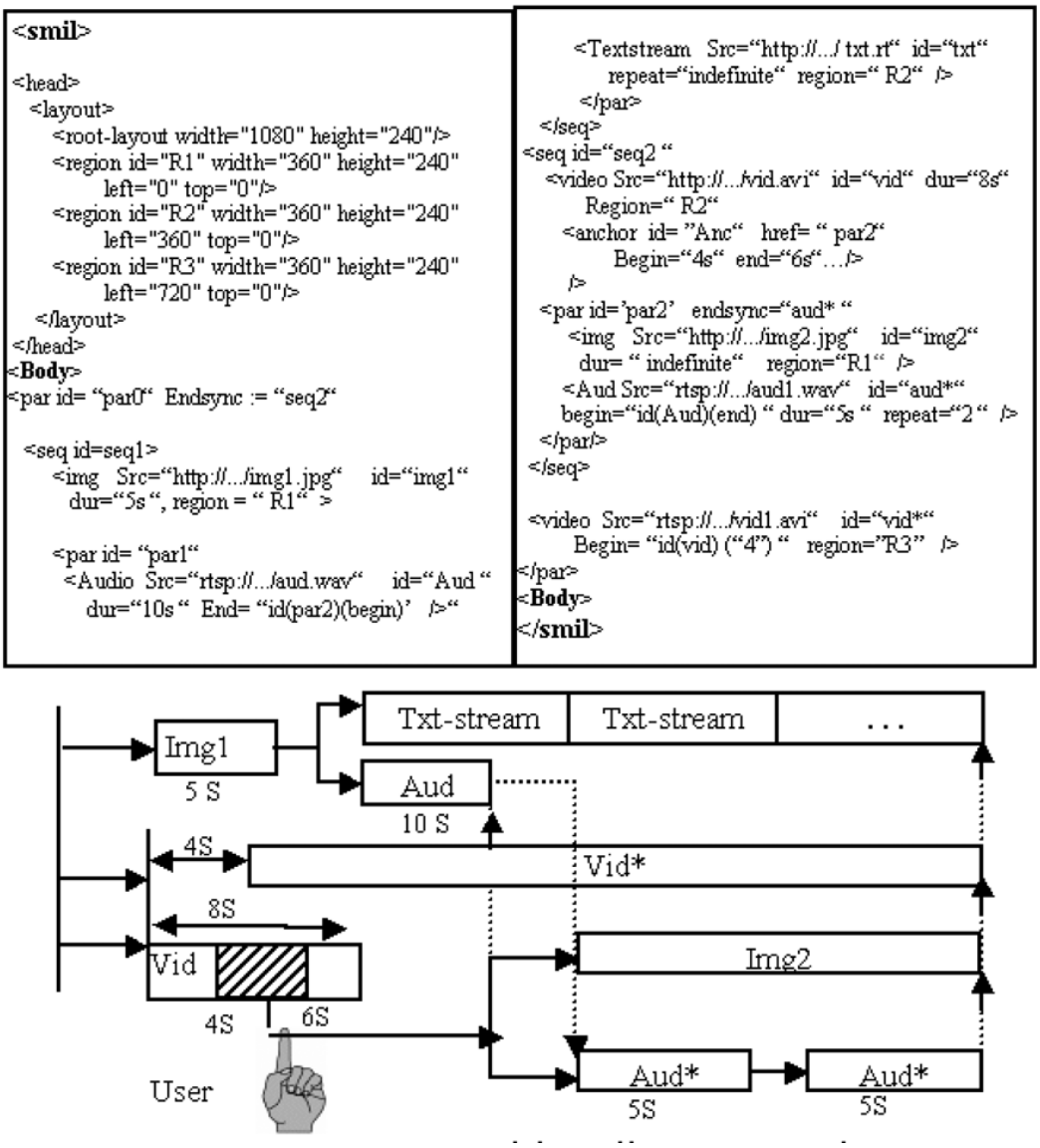

Figure 8. A SMIL multimedia presentation.

5.2. Example

Let's consider the *SMIL* document described in Figure 8, which consists of a sequence seq_1, of an image Img_1 followed by a parallel presentation of an audio clip *Aud* and of an undefined number of a text stream *Txt*. This sequence must be presented simultaneously with two other sequences: The first of the two sequences seq_2 shows a video clip *Vid* followed by the parallel presentation par_2 of an image Img_2 and a double repetition of an audio clip Aud^*. The other sequence presents a video clip Vid^* that begins 4*s* after the start of *Vid*. We assume that the duration of Img_1, *Aud*, *Vid*, and Aud^* are, respectively, 5, 10, 8 and 5 seconds, and that the medias Img_2, *Txt* and Vid^* do not have an explicit duration.

Besides, notice that the start of the block par_2 entails the end of the media *Aud* and that Aud^* must begin when the media *Aud* ends. Furthermore, the presentation of *Vid* can be stopped by a user interaction, which may occur between 4 and 6 seconds after starting *Vid*. Therefore, it is assumed that the end of the sequence seq_2 is determined by the end of Aud^* and the end of the whole presentation is determined by the end of the sequence seq_2. Hence, the *SMIL* presentation finishes when Aud^* ends. On the other hand, the presentation uses three regions R_1, R_2 and R_3 wherein the media to visualize must be displayed. The region R_1 receives Img_1

and Img_2 whereas the region R_2 is allocated to media Vid and Txt. Finally, the region R_3 is dedicated to media Vid^*.

5.3. Modelling the *SMIL* Requirements

The requirements that concern the *SMIL* document given in Figure 8 can be modelled by the *STPTPN* in Figure 9. The requirements related to each media are described as a single Petri net unit. Then, the various units are organized according to the structuring schema given in the *SMIL* document. Initially, the standard places P_{14}, P_0 and P_{17} are marked thus enabling the starts of the units Vid, Img_1 and Vid^*, respectively. As regards resource requirements, they are expressed using preemptor hyperarc mechanism. Note that the policy of order is established by $Real-player$ [7] by giving the priority to the media tag introduced in the last among the tags of the same block. For instance, media Vid has priority over the other media for the resources of which it requires for its presentation. By the way, the resources liable to conflict in the *SMIL* document are of two types:

— The layout resources introduced here, through the three regions denoted by R_1 R_2 and R_3, the allocation schemes of which are modelled using resources places. For instance, the region R_1 is modelled using the set of resource places $(fr(R_1), bs_1(R_1), bs_2(R_2))$. Notice that $fr(R_1)$, once marked, denotes that the region R_1 is available whereas $bs_1(R_1)$ respectively $bs_2(R_1)$, once marked, denotes that R_1 is used to display the media Txt respectively the media Vid. However, Vid has priority on Txt, which is set using the violator arc going from the resource place $bs_1(R_1)$ to transition $B(Vid)$.

— The audio device denoted by "A" is specified using the resource places $(fr(A), bs_1(A), bs_2(A))$, and is liable to conflict between medias Aud and Aud^*.

The *SMIL* synchronization requirements are specified by the set of rendezvous using the rules *AND* and *MAS* regarding the semantics of the *SMIL* language. For this effect, the transition Syn is added in unit Vid^*, to synchronize the starts of both media Vid and Vid^*, since Vid^* should start 4 seconds after Vid. Moreover, to model the synchronization between the start of both Img_2 and Aud^* (i.e., block Par_2), and the end of Aud, we add the transition $B(par_2)$ denoting the start event of the parallel block that has to synchronize with the transition $E(Aud)$, as required in the *SMIL* document.

Notice that the transition, $B(par_2)$ is connected in its input to unit Vid and in its output to units Img_2 and Aud^*, following the structuring of the presentation shown in Figure 8. Moreover, the unit Img_1 is connected on its output to unit Aud and Txt.

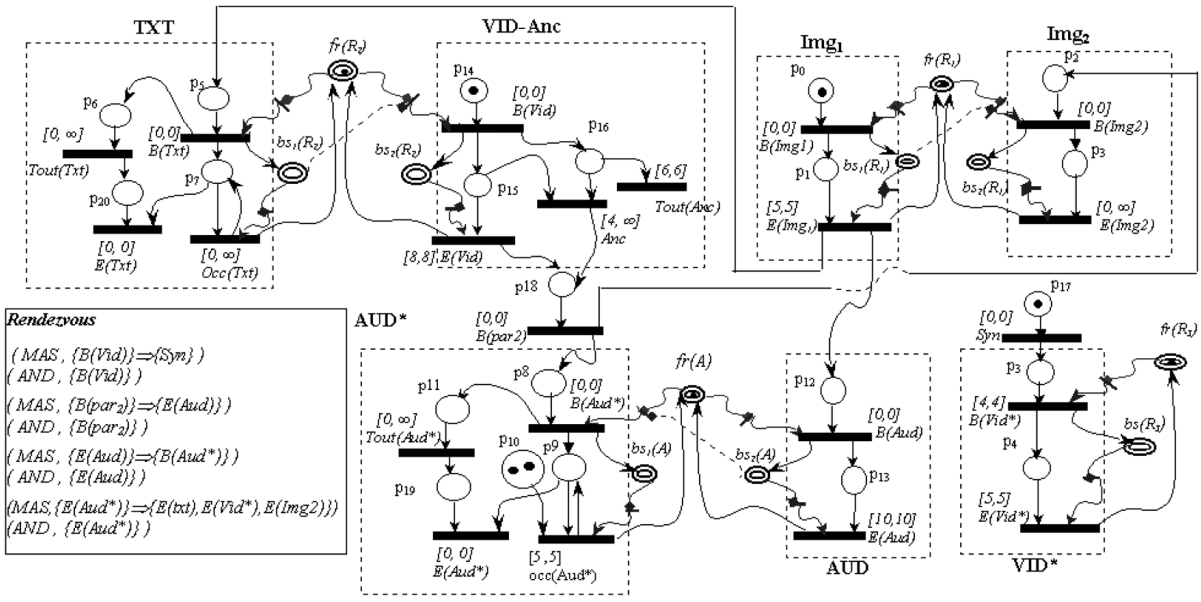

Figure 9. The STPTPN modelling the SMIL presentation requirements.

It should be noted that the *SMIL* synchronization requirement of type $A :=$ "B" denotes that event A can occur only if the event B can occur too. However, B does not need the availability of A to be performed. This semantics does not exactly match the *STPTPN* one. Consequently, we need to express each *SMIL* synchronization, using two rendezvous, one for modelling the occurrence of the master synchronization when all events are provided, the second one modelling the case where the master event has to occur alone, because of the unavailability of the slave event. For instance, the start of media Aud^* is decided by the end of media Aud, which is specified by the master rendezvous $(MAS,\ \{E(Aud)\} \Rightarrow \{B(Aud^*)\})$. However, if the master scheme cannot be performed, then $E(Aud)$ can evolve in asynchronous manner, according to the rendezvous $(AND, E(Aud))$.

To summarize, the purpose of this modelling process is to specify the different constraints imposed on the document. Hence, the availability of an algorithm enumerating the system behaviors based on the semantics given in Section 4.2 will make it possible to clarify the dysfunctions of the *SMIL* presentation, the origin of which is a bad characterization of the time and synchronization constraints, even of resource conflicts. Identification of the cause of these inconsistencies will permit to cure it by carrying out the formatting of the *SMIL* document. Besides, an algorithm deriving the reach-ability graph of the model will allow deducing an efficient access pattern which could be used in the management of a pre-fetching scheme that could be set up at different levels (client, Proxy and server)[29].

6. Comparison with Other Models

In this section, we compare the *STPTPN* model with other existing models, particularly those which have been dedicated to model multimedia requirements.

— **Timed automata [24]:** Timed Automata [24, 25] seem to be a powerful model, since they allow expressing many of the semantic features introduced in *STPTPN*, as for instance the *stopwatch* one [10]. However, timed automata are not dedicated to specification, since they have not the natural expressiveness of algebraic and Petri net models. They are used specifically as an intermediate model, whereon temporal logics can be applied for analysing the real time properties of the basic specification [26].

— **RT-LOTOS language [27]:** Algebraic specifications have been particularly used to model the time requirements of multimedia systems. For instance, $RT-LOTOS$ language, which is one of the most powerful algebraic languages, has been widely used to model multimedia systems [4, 27]. However, $RT-LOTOS$ is not provided with mechanisms like time suspension, as well as resource preempting mechanisms. Hence, the $RT-LOTOS$ language has been only dedicated to temporal consistency analysis of multimedia systems, without taking into account the issues considered in this paper.

— **TSPN MODEL [5, 6]:** *TSPN*, which has been defined to model weakly synchronous multimedia systems, seems to be the closest model to *STPTPN*. Thus, the comparison between both models allows us to point out some differences: In *TSPN*, time intervals are associated with ingoing arcs. The *TSPN* places model the different processes of the system called *streams* (see Figure 1.h). Each synchronization scheme between different processes is given by a rule associated with the related transition. Consequently, *TSPN* cannot be considered as a straightforward extension of *TPN*, Thus, it is not adapted to model event-based specification since each process is modelled as a place. Moreover, some behaviors cannot be modelled easily and naturally with *TSPN*, leading to ambiguous specification [17]. As a particular case, modelling the resource requirements of a process in *TSPN* is quite fuzzy since the place is used to model a process. Consequently, as *STPTPN* allows to take into account the synchronizing schemes introduced in *TSPN*, and this in addition to some features like time suspension and resource preempting mechanism, which have not been considered in *TSPN*, it provides a more powerful and adapted model in the general purpose to model a large range of multimedia requirements, while preserving all the accuracy and the natural expression power that made *TPN*, a widely used practical model.

However, the previous languages are provided with tools and techniques allowing their analysis and their authoring, whereas *STPTPN* is still under construction. Consequently, in order to make the *STPTPN* a practical model, we

must provide the latter with techniques allowing its analysis and an environment performing the different stages of the verification:

- Proposing an algorithm that allows deriving the reachability graph of the *STPTPN* model. Within this intention, two issues have to be considered: the first issue deals with the stopwatch semantics which is well known to harden the construction of the *TPN* reachability graph. Within this context, different approaches [10, 18, 30, 31] have been addressed in the literature, aiming to approximate and to ease the graph construction. The latter could be adapted in the case of the *STPTPN* model. The second issue deals with the synchronization semantics that implies to extend the existing algorithms [32] to handle more complex firing semantics,
- An authoring module with a graphical interface,
- A translator module which parses the multimedia document into an *STPTPN* model,
- An interpreting module allowing to parse the *STPTPN* model into its corresponding reach-ability graph and to check therein the properties of the specification,
- A simulator performing simulation tests, by processing a given *STPTPN* specification, based on the formal semantics of the model.

7. Conclusion

In this paper we defined a new model called *STPTPN* (*Preemptive time Petri nets with synchronizing transitions*) towards specifying multimedia requirements. In this model, the resources are modelled as special places while adapting the firing-rule semantics. Thus, we introduced three special events when firing a transition t: (i) A *strong event* noted t, denoting that t was fired while getting the resource required; (ii) a *violated event* noted t^*, denoting that t has its resource violated by a higher priority transition; and (iii) a *violator event* noted ${}^*t$, meaning that t is violating the resource of a lesser priority transition. The priority is set by using a new mechanism called *preemptor hyperarc*, which determines the event type generated when firing a transition. Furthermore, to express time suspension mechanism, a *stopwatch* is associated with each transition which lets time-passage be stopped, resumed and started, by using classical arcs and inhibitor arcs. Therefore, to model the different synchronization schemes as those defined in *TSPN*, we consider the simultaneous firing of a set of transitions (called *rendezvous*), according to different synchronizing rules.

Finally, we have presented the *STPTPN* syntax and formal semantics. Then, we compared our model to other existing models. Further works will lead us to develop an algorithm by adapting existing methods [18, 30, 31, 32]for building the *STPTPN* reachability graph in order to perform some analysis, thus allowing to check the real time properties and the consistency of multimedia systems.

8. Acknowledgment

Author A. Abdelli would like to thank Professor Daniel Kayser of Paris XIII university (France), for his support in writing this paper during author's stay in the LIPN laboratory.

References

[1] SMIL 2.0: W3C Recommendations, SMIL 2.0 Specification.
`http://www.w3.org/TR/SMIL20`

[2] G. BLAKOWSKI, R. STEINMETZ. A media synchronization survey: Reference model, specification, and case studies. *IEEE JSAC*, Vol. 14, No. 1, Jan. 1996, pp. 5–35.

[3] M. HAINDL. A new Multimedia synchronization model. *IEEE Journal on selected area in communications*, Vol. 14, No. 1, Jan. 1996, pp. 73–88.

[4] J. P. COURTIAT, M. DIAZ, R. C. DE OLIVEIRA, P. SENAC. Proving temporal consistency in a new multimedia synchronization model. *In Proc of ACM Multimedia*, Boston Nov. 1996.

[5] P. SENAC. Modeling logical and temporal synchronizations. *In IEEE JSAC*, Vol. 14, No. 1, Jan. 1996, pp. 84–103.

[6] M. DIAZ, P. SENAC. Time Stream Petri Nets: A model for timed multimedia information. *In Proc of the 15th ICATPN*, Vol. 815 of LNCS, Zaragosse, Spain, June 1994, pp. 219–238.

[7] Real Player Home page `http://real.com`

[8] B. WALTER. Timed net for modelling and analysing protocols with time. *In Proceedings of the IFIP Conference on Protocol Specification Testing and Verification*, North Holland, 1983.

[9] O. H. ROUX, D. LIME. Time Petri Nets with Inhibitors Hyperarcs, Formal Semantics and State Space Computation. *In Proc ICATPN 2004 LNCS Bologna*, Italy.

[10] F. CASSEZ, K. G. LARSEN. The Impressive Power of Stopwatches. *LNCS*, Vol. 1877, Aug. 2000, pp. 138–152.

[11] T. LITTLE, A. GHAFOOR. Synchronization and storage models for multimedia objects. *IEEE JSAC*, Vol. 8, No. 3, March 1990, pp. 413–427.

[12] J. L. PETERSON. Petri Nets. *ACM Computing Surveys*, Vol. 9, No. 3, Sept. 1977.

[13] MHEG: Information technology: coded representation of multimedia and hypermedia objects, DIS ISO/IEC 13522–1, 1994.

[14] ISO/IEC IS 10744. Hypermedia/Time-based Document Structuring Language Hy Time, 1992.

[15] S. FISCHER. Implementation of multimedia systems based on real time extension of ESTELLEs. *In Formal Description Techniques VIII FORTE /PST V'96 Kaiserslautern Germany*, October 1996.

[16] P. MERLIN. A study of the recoverability of computer system. *PHD Thesis Dep. Computer. Science*, Univ. California, Irvine, 1974.

[17] M. BOYER. Contribution á la modèlisation des systmes á temps contraint et application au multimèdia. *PHD Thesis–LAAS*, Toulouse III–2001.

[18] G. BUCCI, A. FEDELI, L. SASSOLI, E. VICARIO. Timed State Space Analysis of Real-Time Preemptive Systems. *IEEE Transaction on Software Engeneering*, Vol. 30, No. 2, Feb. 2004.

[19] M. HACK. PETRI nets language. *Computation Structures Group Memo 127*, MIT 1975.

[20] T. AGERWALA. A complete model for representing the coordination of asynchronous processes. *Technical report*, John Hopkins University, Baltimore Maryland, 1974.

[21] R. JANICKI, M. KOUTNEY. On causality semantics of nets with priority. *Fundamenta Informaticae*, Vol. 38, No. 3, 1999, pp. 233–255.

[22] A. ABDELLI, M. DAOUDI. Un nouveau mècanisme pour la rèsolution du non dèterminisme dans les rèseaux de Petri temporels. *In proc of IEEE Inter symposium SETIT*, Sousse Tunisia, March 2005.

[23] Ambulant Player `http://ambulantplayer.org`

[24] R. ALUR, L. DILL. A Theory of Timed Automata. *Theoretical Computer Science*, Vol. 126, 1994, pp. 183–235.

[25] R. ALUR, C. COURCOUBETIS, N. HALBWACHS, T. A. HENZINGER, P. H. HO, X. NICOLLIN, A. OLIVERO, J. SIFAKIS, AND S. YOVINE. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, Vol. 138, 1995, pp. 3–34.

[26] S. YOVINE. Kronos: A verification tool for real time systems. *International Journal of soft tools for tech transfer*, Vol. 1, No. 1, 1997.

[27] J. P. COURTIAT, C. A. S. SANTOS, C. LOHR, B. OUTTAJ. Experience with RT-LOTOS, a temporal extension of the LOTOS formal description technique. *In Computer Communication*, 23. July 2000.

[28] J. P. COURTIAT. Providing consistent SMIL2.0 documents; *ICME'2002*, Suisse, 26–29 August 2002, 4p.

[29] A. ABDELLI AND N. BADACHE. A Semantics Based Pre-fetch Scheme for SMIL presentation proxy-delivery; *In proc of 12th IEEE Multimedia Modeling conference*, Beijing, China, Jan. 2006.

[30] B. BERTHOMIEU, D. LIME, O. H. ROUX, F. VERNADAT. Problèmes d'accessibilitè et espaces d'ètats abstraits des rèseaux de Petri temporels chronomètres, RS – JESA, 39/2005. MSR, 05, pp. 223–238.

[31] G. BUCCI, L. SASSOLI, E. VICARIO. ORIS: a tool for state-space analysis of real-time preemptive systems, *Proc. QEST 2004*, Enschede (Olanda), September 2004, pp. 70–79.

[32] B. BERTHOMIEU, M. DIAZ. Modeling and verification of time dependant systems using Time Petri Nets. *IEEE Transactions on Software Engineering*, Vol. 17, No. 3, March 1991, pp. 259–273.

Received: December, 2005
Accepted: July, 2006

Contact addresses:

Abdelkrim Abdelli
Computer Science Department
USTHB University
Bp 32, El Alia Babezzouar Algiers, Algeria
e-mail: `Abdelli@lsi-usthb.dz`

Nadjib Badache
Computer Science Department
USTHB University
Bp 32, El Alia Babezzouar Algiers, Algeria
e-mail: `Badache@lsi-usthb.dz`

ABDELKRIM ABDELLI is an assistant professor of computer science in the Department of University *USTHB* of Algiers. His areas of interest include modelling and analysis of real time systems using Petri nets, and also multimedia systems.

NADJIB BADACHE is a full professor in the Department of Computer Science, University *USTHB* of Algiers, and runs the *LSI* laboratory in the same department. His areas of interest include mobile adhoc network and security issues.