Analysis of the MPEG-2 Encoder Algorithm with Timed-Arc Petri Nets *

Valentín Valero, Fernando L. Pelayo, Fernando Cuartero, and Diego Cazorla

Departamento de Informática
Escuela Politécnica Superior de Albacete
Universidad de Castilla-La Mancha
Campus Universitario s/n. 02071. Albacete, SPAIN
{ valentin, fpelayo, fernando, dcazorla }@info-ab.uclm.es

Abstract. In this paper we analyse a parallel version of the MPEG algorithm for video encoding, by using Timed-Arc Petri nets. These are a timed extension of Petri nets, in which tokens have an age, and arcs connecting places with transitions are labelled with a time interval, thus establishing a restriction for the tokens that must be used in order to fire the transitions. With this model it is possible that some tokens become dead, in the sense that they cannot be used for the firing of any transitions in the future, and then we exploit this fact in order to extend the classical notion of safeness, thus defining the soft-safeness of a Timed-Arc Petri net. Then, we analyse a version of the MPEG–2 encoder, modelling it with a MTAPN. The main consequence that we can obtain from the model is that some parts of the algorithm can be made in parallel, and thus, provided that we have several processors, we can improve the performance of the encoder significatively.

1 Introduction

A survey of the different approaches to introduce time in Petri nets is presented in [5]. We can identify a first group of models, which assign time delays to transitions, either using a fixed and deterministic value [11, 12] or choosing it from a probability distribution [3]. Other models use time intervals to establish the enabling times of transitions [10]. Finally, we have also some models that introduce time on tokens [2, 4, 13]. In such a case tokens become classified into two different classes: available and unavailable ones. Available tokens are those that can be immediately used for firing a transition, while unavailable cannot. We have to wait for a certain period of time for these tokens to become available, although it is also possible for a token to remain unavailable forever (such tokens are said to be *dead*). More recently, Cerone and Maggiolo-Schettini [6] have defined a very general model (statically timed Petri nets), where timing constraints are intervals statically associated with places, transitions and arcs.

* This work has been supported by the CICYT project "Performance Evaluation of Distributed Systems", TIC2000-0701-C02-02.

Thus, models with timing constraints attached only to places, transitions or arcs can be obtained by considering particular subclasses of this general framework.

Timed-Arc Petri nets [1, 4, 7, 8, 13, 14] are a timed extension of Petri nets in which tokens have associated a non-negative real value indicating the elapsed time from its creation (*its age*), and arcs from places to transitions are also labelled by time intervals, which establish restrictions on the age of the tokens that can be used to fire the adjacent transitions. As a consequence of these restrictions some tokens may become *dead*, because they will be never available, since they are too old to fire any transitions in the future. The interpretation and use of Timed-Arcs Petri nets can be obtained from a collection of processes interacting with one another according to a rendez-vous mechanism. Each process may execute either local actions or synchronization ones. Local actions are those that the process may execute without cooperation from another process, and thus in the Petri net model of the whole system they would appear as transitions with a single precondition place, while synchronization actions would have several precondition places, which correspond to the states at which each one of the involved processes is ready to execute the action. Then, each time interval establishes some timing restrictions related to a particular process (for instance the time that a local processing may require). In consequence, the firing of a synchronization action can be done in a time window, which depends on the age of the tokens on its precondition places.

Therefore, Timed-Arc Petri nets are a very appropriate model for the description of concurrent systems with time restrictions, such as manufacturing systems, real-time systems, process control, workflow systems, etc. In this paper we show the use of TAPNs for the modeling of safe concurrent systems. But we also exploit the fact that some tokens may become dead in order to extend the classical notion of safeness, defining the *soft-safeness* of a TAPN. As illustration, we model the MPEG-2 Video Encoder by using TAPNs. The ISO/IEC 13818–2 standard [9], commonly known as MPEG–2, is a standard intended for a wide range of applications, including Video–on–Demand (VoD), High Definition TV (HDTV) and video communications using broadband networks.

The MPEG standards were designed with these two requirements:

- The need for a high compression, which is achieved by exploiting both spatial and temporal redundancies within an image sequence.
- The need for random access capability, which is obtained by considering a special kind of pictures (I pictures), which are encoded with no reference to other frames, only exploiting the spatial correlation in a frame.

This paper shows the specification and analysis of a singular version of parallel MPEG–2, in which instead of distribute the data among the processors by either distributing different partitions of the same frame (spatial parallelism) or different GoPs to the various processors (temporal parallelism), as usual, we will benefit from the intrinsic parallelism that we find in the MTAPN modelling this encoder for each GoP, and thus some improvements for the encoder can be proposed as a consequence of this.

The results so obtained improve significatively the performance of the standard MPEG–2, and we have studied the performance we can get both with two and ten processors.

The paper is structured as follows. In Section 2 we present timed-arc Petri nets and their semantics, and we also define the notion of *soft-safeness*. In Section 3 we show how to construct a state graph for bounded MTPANs, and how to extend this construction for soft-safe MTPANs. In Section 4 we describe how the MPEG-2 encoder works and we present a MTAPN that models this algorithm. Finally, the analysis of the algorithm and some conclusions are presented in Section 5.

2 Timed-Arc Petri Nets

We deal with timed-arc Petri nets, which have their tokens annotated with an age (a real value indicating the elapsed time from its creation) and arcs connecting places with transitions have associated a time interval, which limits the age of the tokens that must be consumed to fire the adjacent transition.

However, a transition is not forced to be fired when all its preconditions contain tokens with an adequate age, and the same is true even if the age of any of these tokens is about to expire. More in general, in the model we consider[1] there is not any kind of urgency, what we can interpret in the sense that the model is *reactive*, as transitions will be only fired when the environment requires it. But then, it can be the case that the external context may lose the ability to fire a transition if some needed tokens become too old. Even more, it is possible that some tokens become *dead*, which means definitely useless because their increasing age will not allow in the future the firing of any of their postcondition transitions.

2.1 Definitions

Definition 1. (Timed-arc Petri nets)
We define a timed-arc Petri net (TAPN) as a tuple [2] $N = (P, T, F, times)$, where P is a finite set of *places*, T is a finite set of transitions ($P \cap T = \emptyset$), F is the *flow relation* ($F \subseteq (P \times T) \cup (T \times P)$), and *times* is a function that associates a closed time interval to each arc (p, t) in F, i.e.: $times : F|_{P \times T} \longrightarrow \mathbb{R}_0^+ \times (\mathbb{R}_0^+ \cup \{\infty\})$.

When $times(p, t) = [x_1, x_2]$ we write $\pi_i(p, t)$ to denote x_i, for $i = 1, 2$.

As we previously mentioned, tokens are annotated with real values, so markings are defined by means of multisets on $\mathbb{R}_0^+$. More exactly, a marking M is a function:

$$M : P \longrightarrow \mathcal{B}(\mathbb{R}_0^+)$$

[1] Other proposals of Timed-arc Petri nets [8] enforce the firing of transitions with an earliest and maximal firing rule.

[2] We consider only arcs with weight 1 to simplify some definitions, but the extension to general arcs with greater weights is straightforward.

where $\mathcal{B}(\mathbb{R}_0^+)$ denotes the set of finite multisets of non-negative real numbers. Thus, as usual, each place is annotated with a certain number of tokens, but each one of them has associated a non-negative real number (its *age*). We will denote the set of markings of N by $\mathcal{M}(N)$, and using classical set notation, we will denote the number of tokens on a place p by $|M(p)|$.

As initial markings we only allow markings M such that for all p in P, and any $x > 0$ we have $M(p)(x) = 0$ (i.e., the *initial age* of any token is 0). Then, we define *marked timed-arc Petri nets* (MTAPN) as pairs (N, M), where N is a timed-arc Petri net, and M is an initial marking on it. As usual, from this initial marking we will obtain new markings, as the net evolves, either by firing transitions, or by time elapsing. In consequence, given a non-zero marking, even if we do not fire any transitions at all, starting from this marking we get an infinite reachability set of markings, due to the token aging.

A timed-arc Petri net with an arbitrary marking can be graphically represented by extending the usual representation of P/T nets with the corresponding time information. In particular we will use the age of each token to represent it. Therefore, MTAPNs have initially a finite collection of zero values labelling each place. □

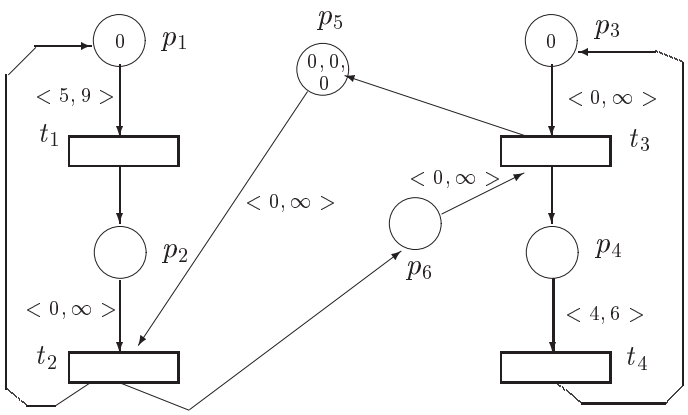

Fig. 1. Timed-arc Petri net modeling the PC-problem

In Fig. 1 we show a MTAPN modeling a producer/consumer system, where we have represented by transition t_1 the action corresponding to the manufacturing process of the producer, which takes between 5 and 9 units of time, and by t_2 the action of including the generated object into the buffer. Notice that the initial tokens on p_5 represent the capacity of the buffer (3), and the arc connecting this place with t_2 is labelled by the interval $< 0, \infty >$, because these tokens can be consumed at any instant in the future. Tokens on p_6 represent the objects on the buffer which have not been yet consumed. Transition t_3 models the action of taking out an object from the buffer, which can occur at any instant. Finally, transition t_4 models the processing that makes the consumer for the objects extracted from the buffer, and this action takes between 4 and 6 units of time.

Let us observe that if the enabling time for the firing of one of these transitions (t_1 or t_4) expires, the system eventually becomes deadlocked, because we obtain a *dead* token either on p_1 or p_4.

Let us now see how we can fire transitions, and how we model the time elapsing.

Definition 2. (Firing rule)
Let $N = (P, T, F, times)$ be a TAPN, M a marking on it, and $t \in T$.

(i) We say that t is *enabled* at the marking M if and only if: $\forall p \in {}^\bullet t \ \exists x_p \in \mathbb{R}_0^+$ such that $M(p)(x_p) > 0 \wedge x_p \in times(p,t)$, i.e., on each precondition of t we have some token whose age belongs to $times(p,t)$.

(ii) If t is enabled at M, it can be fired, and by its firing we reach a marking M', defined as follows:

$$M'(p) = M(p) - C^-(p,t) + C^+(t,p), \ \forall p \in P$$

where both the subtraction and the addition operators work on multisets, and:

- $C^-(p,t) = \begin{cases} \{x_p\} & \text{if } p \in {}^\bullet t, x_p \in times(p,t) \\ & \text{and } x_p \in M(p) \\ \emptyset & \text{otherwise} \end{cases}$
- $C^+(t,p) = \begin{cases} \emptyset & \text{if } p \notin t^\bullet \\ \{0\} & \text{otherwise} \end{cases}$

Thus, from each precondition place of t we remove a token fulfilling (i), and we add a new token (with age 0) on each postcondition place of t.

As usual, we denote these evolutions by $M[t\rangle M'$, but it is noteworthy that these evolutions are in general non-deterministic, because when we fire a transition t, some of its precondition places could hold several tokens with different ages that could be used to fire it. Besides, we see that the firing of transitions does not consume any time. Therefore, in order to model the time elapsing we need the function *age*, defined below. By applying it we age all the tokens of the net by the same time:

(iii) The function $age : \mathcal{M}(N) \times \mathbb{R}_0^+ \longrightarrow \mathcal{M}(N)$ is defined by:

$$age(M,x)(p)(y) = \begin{cases} M(p)(y-x) & \textit{if } \ y \geq x \\ 0 & \textit{otherwise} \end{cases}$$

The marking obtained from M after x units of time without firing any transitions will be that given by $age(M,x)$. □

Although we have defined the evolution by firing single transitions, this can be easily extended to the firing of *steps* or *bags* of transitions; those transitions that could be fired together in a single step could be also fired in sequence in any order, since no *aging* is produced by the firing of transitions. In this way we

obtain step transitions that we denote by $M[R\rangle M'$. Finally, by alternating step transitions and time elapsing we can define a timed step semantics, where timed step sequences are those sequences $\sigma = M_0[R_1\rangle_{x_1} M_1 \ldots M_{n-1}[R_n\rangle_{x_n} M_n$, where M_i are markings, R_i multisets of transitions and $x_i \in \mathbb{R}_0^+$, in such a way that $M_i[R_{i+1}\rangle M'_{i+1}$ and $M_{i+1} = age(M'_{i+1}, x_{i+1})$. Note that we allow $x_i = 0$ in order to capture the execution in time zero of two causally related steps.

Then, given a MTAPN (N, M_0), we define $[M_0\rangle$ as the set of reachable markings on N starting from M_0, and we say that N is bounded if for every $p \in P$ there exists $n \in \mathbb{N}$ such that for all $M \in [M_0\rangle$ we have $|M(p)| \leq n$.

A token on a place p at a marking M is said to be *dead* if it can never be used to fire any transitions, i.e., it will remain on its place forever, just growing up. Thus, we say that a marking is *dead* when all its tokens are *dead*.

In a previous paper [13] we have shown that TAPNs have a greater expressiveness than PNs, even although TAPNs are not Turing complete, because they cannot correctly simulate a 2-counter machine. In that paper we proved that reachability is undecidable for TAPNs. Other properties that we have studied in a more recent paper [7] are coverability, boundedness and detection of dead tokens, which are all decidable for TAPNs. Decidability of coverability has been also proved in [1] for an extended version of TAPNs, in which all arcs can be annotated with bags of intervals in $\mathbb{N} \times (\mathbb{N} \cup \{\infty\})$.

2.2 Safe Timed-Arc Petri Nets

We say that (N, M_0) is safe if $|M(p)| \leq 1,\ \forall p \in P$, for every $M \in [M_0\rangle$. But let us observe that once a token becomes dead, it cannot be used for the firing of any transition in the future, and thus, we can extend the notion of safeness of a MTAPN taking into account that these tokens will not affect the behaviour of the net. Then, we may define a *soft-safe Timed-Arc Petri Net* as a MTAPN for which there is at most one non-dead token on each place for every reachable marking.

It is quite obvious that if N is a safe MTAPN, then it is also soft-safe, but the converse is not true in general. Furthermore, given a MTAPN N, if the underlying untimed net, $untimed(N)$ is safe, then N is also safe. However, the safeness of a MTAPN N does not imply the safeness of $untimed(N)$ (see Fig. 2).

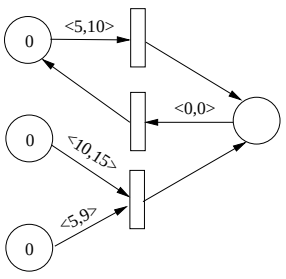

Fig. 2. A safe MTAPN N for which $untimed(N)$ is not safe

Moreover, as we can see in Fig. 3 a MTAPN N can be soft-safe, being unsafe its underlying untimed net.

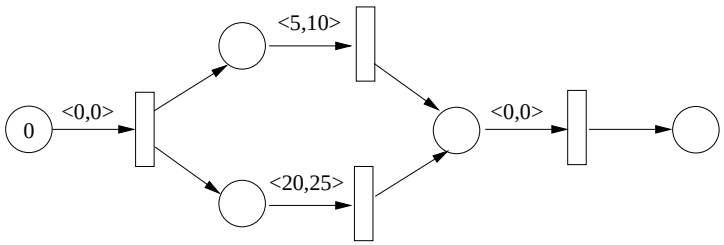

Fig. 3. A soft-safe MTAPN N for which $untimed(N)$ is not safe

3 State Graph

Let us observe that even in the case of bounded nets the number of reachable markings is infinite, because of the growing of the token ages. If we restrict ourselves to bounded timed-arc Petri nets with time intervals in $\mathbb{N} \times \mathbb{N} \cup \{\infty\}$, we may follow the same ideas in [8], in order to construct a state graph, by using the fact that for every place p of a timed-arc Petri net we can find a maximal value $Max(p)$ for the age of its tokens to influence the activation of its postcondition transitions, because the tokens on that place with an age exceeding that maximal value can only be consumed by the firing of some transition $t \in p^\bullet$ for which $\pi_2(p,t) = \infty$.

Concretely we may define:

$$Max(p) = Max\{\pi_i(p,t) \mid t \in p^\bullet,\ \pi_i(p,t) < \infty,\ i = 1,2\}$$

and $S(p) = 1 + Max(p)$. Then, it follows that once the age of a token on p exceeds $Max(p)$ the only postcondition transitions $t \in p^\bullet$ that could be fired by using that token are those for which $\pi_2(p,t) = \infty$. Obviously, in order to fire such a transition t the age of the involved token on p is unimportant once it exceeds $S(p)$. This means that in order to construct the state graph we can represent by the single value $S(p)$ the whole interval $[S(p), \infty]$.

Definition 3. (State graph)
Given a bounded MTAPN $N = (P, T, F, times, M_0)$ with time intervals in $\mathbb{N} \times (\mathbb{N} \cup \{\infty\})$, we define its state graph $G(N) = (V, E, M_0)$, where V is the set of markings of N such that the age of the tokens of every place $p \in P$ belongs to $\{0, \ldots, S(p)\}$, M_0 is the initial state and $E \subset V \times T \times \mathbb{N} \times V$.

An arc $(M, t, r, M') \in E$ means that t can be fired at the marking $age(M, r)$ applying Def. 2, and M' is the reached state, but changing for every place p the age of the tokens exceeding $Max(p)$ to $S(p)$.

Thus, we start from M_0 and we apply the modified firing rule to successively get the new reachable states of the graph. We must observe that as we are not imposing the urgent firing of transitions, in principle we could have infinitely many arcs leaving each state. However, for each state $M \in V$ we have that from a certain instant $r \in \mathbb{N}$ onwards, all its outgoing arcs will reach the same state, because the ages of the tokens in each place $p \in P$ at the markings $age(M, s)$, for $s \geq r$, would be greater than their corresponding maximal value, $Max(p)$. As a consequence, for each state $M \in V$ we can limit the application of the firing rule to the markings obtained aging M up to time $r_{max} = Max\{S(p) \,|\, p \in P\}$.

According to this definition, taking into account that V is finite and that we have a finite number of arcs leaving each state, we conclude that $G(N)$ is finite.

□

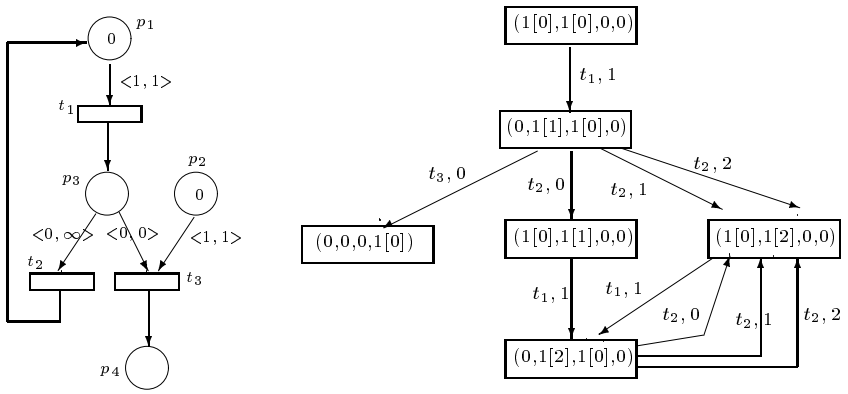

Fig. 4. A bounded MTAPN and the corresponding state graph

In Fig. 4 we can see a bounded MTAPN and its corresponding state graph. For this net we have that $S(p_1) = 2, S(p_2) = 2, S(p_3) = 1, S(p_4) = 0$ and thus $r_{max} = 2$. Graph states are shown by indicating for each place the number of tokens of a certain age that we have on this place, for instance 1[0] means that we have one token with age 0 in the corresponding place.

Then, we have the following result which relates reachable markings of N with states in $G(N)$.

Proposition 1. Given a bounded MTAPN $N = (P, T, F, times, M_0)$ with time intervals in $\mathbb{N} \times (\mathbb{N} \cup \{\infty\})$, and its corresponding state graph $G(N) = (V, E, M_0)$. If $M \in [M_0\rangle$, then there is a state $M' \in V$ such that $\varphi(M) = \varphi(age(M', r))$, for a certain $r \in \mathbb{N}$, where φ is a function that for every place p changes the age of the tokens on p exceeding $Max(p)$ to $S(p)$:

$$\varphi : \mathcal{M}(N) \longrightarrow V$$

$$\varphi(M)(p)(n) = \begin{cases} M(p)(n) & \text{if } n \leq Max(p) \\ \sum_{m \geq S(p)} M(p)(m) & \text{if } n = S(p) \\ 0 & \text{if } n > S(p) \end{cases}$$

Proof. By induction on the timed step sequence σ such that $M_0[\sigma\rangle M$. The base case ($M = M_0$) is trivial, so let us consider a reachable marking M_1 for which we have a state $M_1' \in V$ such that $\varphi(M_1) = \varphi(age(M_1', r_1))$, and both possibilities of evolution, either by the simple passage of time or by firing a transition:

- If $M = age(M_1, r_2)$, we have that $\varphi(M) = \varphi(age(M_1', r_1 + r_2))$, as desired.
- If $M_1[t\rangle M$, we have that t is also enabled at $\varphi(age(M_1', r_1))$, because the change of age that φ makes for the tokens exceeding their maximal value does not affect the enabledness of transitions. Besides, we may use for firing t the adequate tokens to get a state M' such that $\varphi(M) = \varphi(M') = \varphi(age(M', 0))$.

□

Thus, the state graph provides us with a necessary condition for a marking to be reachable in a bounded MTAPN, but in general it will not allow us to determine if a given marking is reachable or not.

In the particular case of safe MTAPNs, the state graph will be a little smaller, because we will have at most one single token on each place for every reachable marking. For soft-safe MTAPNs the previous construction will not generate in general a finite graph (if the net is unbounded). However, dead tokens do not affect the future behaviour of the net (in terms of the transitions that we may execute), and thus, we may adapt the previous construction for the particular case of soft-safe MTAPNs.

Definition 4. (State graph for soft-safe MTAPNs)
Given a MTAPN $N = (P, T, F, times, M_0)$ with time intervals in $\mathbb{N} \times (\mathbb{N} \cup \{\infty\})$, we may define its state graph $G^*(N) = (V, E, M_0)$, in a similar way to that followed in Def. 3, but in this case, once we have obtained an arc (M, t, r, M') according to Def. 3, we check for every place $p \in P$ if there are two or more tokens with the same age on p at M', and in this case we only consider one of them, i.e., we will finally take an arc (M, t, r, M''), where:

$$M''(p)(j) = Min\{1,\, M'(p)(j)\} \;\; \forall j \in \{0, \ldots, S(p)\}$$

With this construction we can generate only a finite number of possible states, and thus the graph is finite. □

With this new construction we have removed some tokens which appear across the normal evolution of the net, but the important fact is that these tokens are *innocuous*, in the sense that they do not affect the future behaviour of the net, because we are supposing that it is soft-safe. Of course, we must pay an additional price for that: we are losing some more information concerning reachability.

Proposition 2. Given a soft-safe MTAPN $N = (P, T, F, times, M_0)$ with time intervals in $\mathbb{N} \times (\mathbb{N} \cup \{\infty\})$, and its corresponding state graph $G^*(N) = (V, E, M_0)$. If $M \in [M_0\rangle$, then there is a state $M^* \in V$ such that $\varphi(M) \geq \varphi(age(M^*, r))$, for a certain $r \in \mathbb{N}$, where:

- $M_1 \geq M_2$ if and only if $\forall p \in P : M_1(p)(n) \geq M_2(p)(n)$, $\forall n \in \{0, \ldots, S(p)\}$.
- φ is defined as in Prop. 1.

Moreover, for every place $p \in P$, if there is one non-dead token in p at M, then we have one token in $age(M^*, r)$ with either the same age or both have an age greater or equal than $S(p)$.

Proof. We use again induction on the timed step sequence σ such that $M_0[\sigma\rangle M$. The base case ($M = M_0$) is trivial, so let us consider a reachable marking M_1 for which we have a state $M_1^* \in V$ fulfilling:

- $\varphi(M_1) \geq \varphi(age(M_1^*, r_1))$.
- Every non-dead token at M_1 has its counterpart in $age(M_1^*, r_1)$.

Then, let us consider both possibilities of evolution, either by time elapsing or by firing one transition:

- If $M = age(M_1, r_2)$, it follows that $\varphi(M) = \varphi(age(M_1, r_2)) \geq \varphi(age(M_1^*, r_1 + r_2))$, and every non-dead token in M was non-dead in M_1 too, thus we can apply the induction hypothesis for it, obtaining its counterpart in $age(M_1^*, r_1)$. We just need to age this token in r_2 units of time in order to obtain the token in $age(M_1^*, r_1 + r_2)$.
- If $M_1[t\rangle M$, it follows that t is also enabled at $\varphi(age(M_1^*, r_1))$, because every non-dead token of M_1 has its counterpart in $age(M_1^*, r_1)$ and N is soft-safe. Let M^* be the state that we obtain by applying Def. 4, it can be easily checked that:
  - $\varphi(M) \geq \varphi(M^*) = \varphi(age(M^*, 0))$.
  - If we generate a new non-dead token on a place $p \in P$, this token will have its counterpart in M^*, because the firing of t would generate a new token of age 0 there, unless we already had one token of age 0 in p.

□

Therefore, with this proposition we have again a necessary condition for a marking to be reachable in a soft-safe MTAPN. Notice that safeness of a MTAPN can be effectively decided by using an enumerative algorithm, taking into account the normalization introduced in Def. 3. Specifically, we generate the state graph until reaching a node in which a place has two tokens, in this case the net will not be safe, and the algorithm finishes. If the construction terminates with a state graph in which all the states have a single token on each place, we will have a safe net. This algorithm can be slightly modified to effectively decide whether a MTAPN is soft-safe or not. We annotate a maximum of two tokens with the same age on each place when constructing the graph, and each token is labelled with a *boolean value*, indicating if the token can be consumed in the future. When

the token is generated this value is *false*, but if we detect along the generation that one token can be used to fire one transition, we relabel the token with the value *true* all along the graph. If the net is soft-safe we will never generate two tokens in the same place with values *true* for a same state, and the algorithm will terminate because the number of possible states is finite. On the other hand, when the net is not soft-safe, there must be a sequence of transitions reaching a state in which two tokens staying in a place can be eventually consumed, and thus, we will terminate the execution saying that the net is not soft-safe.

4 MPEG-2 Encoding Algorithm

MPEG standards were designed with two requirements in mind, namely, the need for a high compression, and the need for random access capability. These techniques exploit the fact that video sequences usually contain statistical redundancies in both temporal and spatial directions. Thus, MPEG digital video coding techniques are statistical in nature. Specifically, the basic statistical property upon which MPEG compression techniques rely is inter–pixel region correlation. The contents of a particular pixel region can be predicted from nearby pixel regions within the same frame (intra–frame coding) or from pixel regions of a nearby frame (inter–frame coding).

Perhaps the ideal method for reducing temporal redundancies is one that tracks every pixel from frame to frame. However, this extensive search is computationally expensive. Under the MPEG standards, this search is performed by tracking the information within 16×16 pixels regions, called *macroblocks*. Given two contiguous frames, $frame(t)$ and $frame(t-1)$, for each macroblock in $frame(t)$, the encoder determines the best matching macroblock in $frame(t-1)$ and calculates the *motion vector*, which captures the macroblock translation information. Therefore, the *temporal redundancy reduction processor* generates a representation for $frame(t)$ by using the corresponding macroblock from $frame(t-1)$, and this representation only contains the motion vector and the prediction error (changes between both frames). This technique is called *motion compensated prediction*.

In order to reduce spatial redundancies a DCT (Discrete Cosine Transform) is used. With this coding process some subjective redundancies in the image are removed, on the basis of human visual criteria.

The combination of these two techniques described above are the key elements of the MPEG encoding process. Furthermore, in order to achieve the requirement of random access and high compression, the MPEG–2 standard specifies three types of compressed video frames: I pictures, P pictures and B pictures. I pictures (intracoded pictures) are coded with no reference to other frames, exploiting only spatial correlation in a frame. P pictures (predictive coded pictures) are coded by using motion compensated prediction of a previous I or P picture. Finally, B pictures (bidirectionally–predictive coded pictures) are obtained by motion compensation by using past and future reference frames (I or P pictures).

A group of consecutive I, P and B pictures constitute a structure called Group of Pictures (GoP). Therefore, a video sequence may be seen as a sequence of GoPs.

The block diagram of the MPEG encoder is depicted in Figure 5. In order to understand how the MPEG–2 encoder works, we will consider a typical GoP consisting on the frames IBBP. Despite B pictures appear before P pictures, the encoding order is IPBB because B pictures require both past and future frames as references.

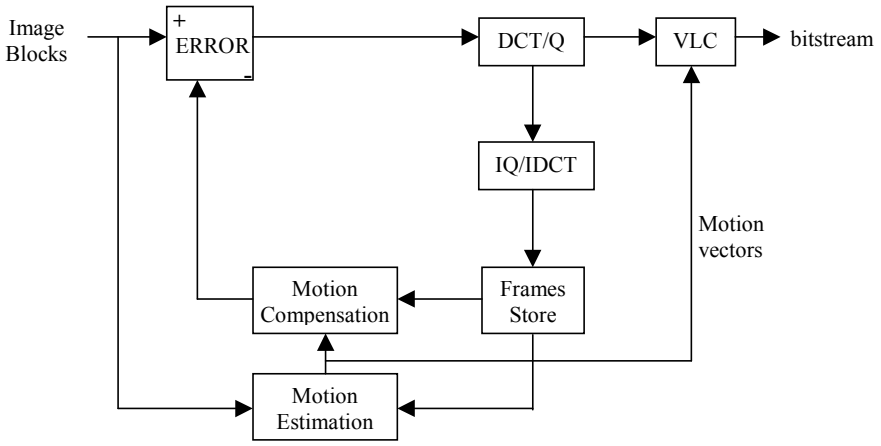

Fig. 5. Block diagram of the MPEG–2

The first frame in a GoP (I picture) is encoded in intra mode without references to any past or future frames. The DCT is applied to each macroblock and then it is uniformly quantized (Q). After quantization, it is encoded using a variable length code (VLC) and it is sent to the output buffer. At the same time the reconstruction (IQ) of all non–zero DCT coefficients belonging to one macroblock and the Inverse DCT (IDCT) give us a compressed I picture which is stored temporarily in the *Frame Store (FS)*.

When the input is coded either as P or B pictures, the encoder does not code the picture macroblocks directly. Instead, it codes the prediction errors and the motion vectors. With P pictures, for each macroblock in the current picture, the motion estimation gives us the coordinates of the macroblock in the I picture that best matches its characteristics and thus, the motion vector may be calculated. The motion compensated prediction error is obtained by subtracting each pixel in a macroblock with its motion shifted counterpart in the previous frame. The prediction error and the motion vectors are coded (VLC) and sent to the output buffer. As in the previuos case, a compressed P picture is stored in the Frame Store.

With B pictures, the motion estimation process is performed twice: for a past picture (I picture in this case), and for a future picture (P picture). Prediction errors and both motion vectors for each macroblock are coded (VLC) and sent to the output buffer. Notice that the compressed B pictures are not stored in the Frame Store, since they are not needed to calculate any other pictures.

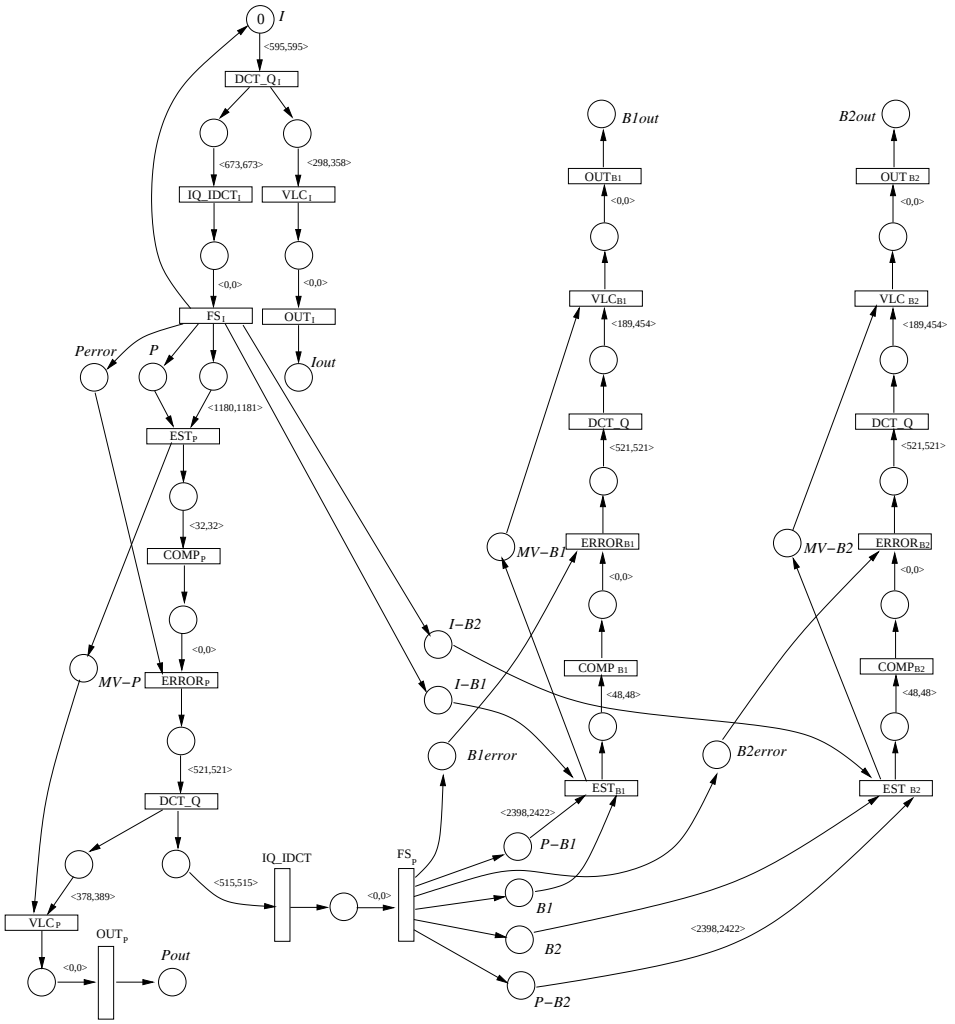

Fig. 6. TAPN which models the MPEG–2 codification process of a GoP

4.1 Timed-Arc Petri Net modeling the MPEG-2

Figure 6 shows the timed-arc Petri net modeling the MPEG-2 encoding algorithm[3]. The left-hand side describes the first part of the encoding algorithm, which corresponds to the generation of both I and P encoded pictures (but remember that even if P pictures are generated before the B pictures, they will appear the last in the final video sequence). Once the places *I-B1, I-B2, P-B1, P-B2* are marked the second part of the net becomes activated (right-hand side),

[3] For simplicity, we omit in this picture the label of the arcs when they are labelled by $<0, \infty>$.

which models the *B1* and *B2* picture encoding process. *Iout* (*Pout* resp.) represents the output of an encoded *I-picture* (*P-picture*), while places *B1out* and *B2out* represent the output of B1 and B2 pictures.

The time intervals that label the arcs connecting places with transitions have been obtained from several real measurements, by coding the "Composed" Video sequence. This experiment has been repeated a number of times, and the results being reported below are therefore the minimum and the maximum of all these trials[4]. During these trials, no other operations were taken place in our experimental setup. The "Composed" Video sequence (format PAL CCIR601, 720x576 pixels) is a representative video sequence which has several different motion levels, and we have encoded it by using a completely software-based MPEG-2 video encoder derived from that developed in Berkeley, which is freely available in the MPEG Home Page: *http://drogo.cslet.stet.it/mpeg*.

In order to get the real values for the different elements of the encoder we have included some patches into the source code, which correspond with the beginning and the end of the elementary actions that we have described in the specification of the algorithm. The real values thus obtained for I and P pictures being in the output buffer are shown in Table 1, as well as the times for encoding the complete GoP.

Picture	Min	Max
I	893 ms	953 ms
P	3688 ms	3738 ms
GoP	11567 ms	12085 ms

Table 1. Measured real values

5 Analysis and Conclusions

We may compute the times to reach every place of a TAPN, by constructing a state reachability graph. With that process we obtain a time interval for every place. In our case, for the TAPN modeling the MPEG–2 encoder we have obtained the results shown in Table 2.

Picture	Min 1st GoP	Max 1st GoP	Min Nth GoP	Max Nth GoP
I	893 ms	953 ms	893 + 1268 * (N-1) ms	953 + 1268 * (N-1) ms
P	3379 ms	3391 ms	3379 + 1268 * (N-1) ms	3391 + 1268 * (N-1) ms
GoP	6672 ms	6962 ms	6672 + 1268 * (N-1) ms	6962 + 1268 * (N-1) ms

Table 2. Values obtained from the TAPN

[4] These values were obtained in a single processor Pentium II - 350MHz platform with 64MB RAM.

Comparing these results with those shown in Table 1 we can see that there are strong differences in order to complete the encoding process of a GoP. For instance, to complete the first GoP the measured values are $[11567\ ms, 12085\ ms]$ while in the TAPN we have obtained $[6672\ ms, 6962\ ms]$. These strong differences are due to the important fact that the encoder only uses a single processor, whereas our TAPN model captures all the intrinsic parallelism of the encoding process, so that with the analysis of the TAPN we are obtaining the required times provided that we have as many processors as needed to take advantage of this parallelism (for a single GoP two processors would be enough).

Let us now consider the encoding process of N GoPs, according to the TAPN the encoding would take $[6672 + 1268 * (N-1)\ ms, 6962 + 1268 * (N-1)\ ms]$, and thus, the average time for encoding each GoP would converge to 1268 ms. This can be interpreted as the time needed to encode a GoP for an infinite video sequence provided that the number of processors is big enough.

Fig. 7 shows the number of processors required in order to process any sequence of GoPs, according to the TAPN of Fig. 6. From that Figure we may conclude that the best performance could be obtained with **10 processors**.

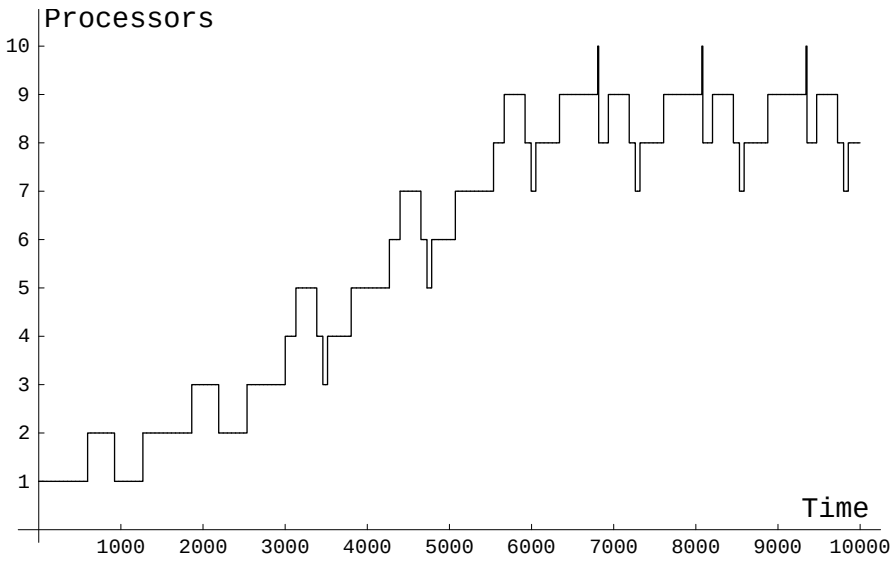

Fig. 7. Graph relating the number of required processors and time (ms)

Consequently, the main conclusion from this analysis of the MPEG–2 encoder by using TAPNs is that the performance of the encoding algorithm for a single GoP can be improved in a factor near to 50% by using two processors. Moreover, in longer sequences of GoPs this factor can be increased, and we have obtained an average time of 1268 ms for encoding each single GoP of an infinite video sequence, provided that we have as many processors as required (10 in this particular case).

References

1. P.A. Abdulla and A. Nylén. *Timed Petri Nets and BQOs.* In Proc. ICATPN 2001, Lecture Notes in Computer Science, vol. 2075, pp. 53-70, 2001.
2. W.M.P. van der Aalst. *Interval Timed Coloured Petri Nets and their Analysis.* Lecture Notes in Computer Science, vol. 691, pp. 451-472. 1993.
3. M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte and A. Cumani. *On Petri Nets with Stochastic Timing.* Proc. of the International Workshop on Timed Petri Nets, IEEE Computer Society Press, pp. 80-87. 1985.
4. T. Bolognesi, F. Lucidi and S. Trigila. *From Timed Petri Nets to Timed LOTOS.* Proc. of the Tenth International IFIP WG6.1 Symp. on Protocol Specification, Testing and Verification. 1990.
5. Fred D.J. Bowden. *Modelling time in Petri nets.* Proc. Second Australia-Japan Workshop on Stochastic Models. 1996.
6. Antonio Cerone and Andrea Maggiolo-Schettini. *Time-based expressivity of time Petri nets for system specification.* Theoretical Computer Science (216)1-2, pp. 1-53. 1999.
7. D. de Frutos, V. Valero and O. Marroquín. *Decidability of Properties of Timed-Arc Petri Nets.* Proc. ICATPN 2000, Lecture Notes in Computer Science, vol. 1825, pp. 187-206. 2000.
8. Hans-Michael Hanisch. *Analysis of Place/Transition Nets with Timed-Arcs and its Application to Batch Process Control.* Application and Theory of Petri Nets, LNCS vol. 691, pp:282-299. 1993.
9. ISO/IEC 13818-2. *Draft International Standard Generic Coding of Moving Pictures and Associated Audio.* Recommendation H.262.
10. P. Merlin. *A Study of the Recoverability of Communication Protocols.* PhD. Thesis, Univ. of California. 1974.
11. C. Ramchandani. *Performance Evaluation of Asynchronous Concurrent Systems by Timed Petri Nets.* PhD. Thesis, Massachusetts Institute of Technology, Cambridge. 1973.
12. J. Sifakis. *Use of Petri Nets for Performance Evaluation.* Proc. of the Third International Symposium IFIP W.G.7.3., Measuring, Modelling and Evaluating Computer Systems. Elsevier Science Publishers, pp. 75-93. 1977.
13. V. Valero, D. de Frutos and F. Cuartero. *On Non-decidability of Reachability for Timed-Arc Petri Nets.* Proc. 8th Workshop on Petri Nets and Performance Models, PNPM'99, pp. 188-196. 1999.
14. B. Walter. *Timed Petri-Nets for Modelling and Analysing Protocols with Real-Time Characteristics.* Proc. 3rd IFIP Workshop on Protocol Specification, Testing and Verification, North-Holland. 1983.