

Timed Checkers and Testing

O.Koné, R.Castanet, B.Cousin.

LaBRI, Université Bordeaux I

351, Cours de La Libération, 33405 TALENCE-FRANCE

e-mail : kone@geocub.greco-prog.fr

Tel. (33)56 84 69 00 / Fax (33)56 84 66 69

Abstract.

This paper exhibits the need to specify *physical (or real) time constraints* in systems behaviour, and then proposes methods to check that a system satisfies such constraints. *Timed Petri Nets* are used to specify physical time constraints. These constraints can be checked by a module called *Timed Checker*. Such a module can be characterized from a Timed Petri Net and is able to detect *untimely events* which violate the physical time constraints. Existing methods for test derivation in TTCN format are augmented to test time dependant systems specified by Timed Petri Nets.

Key Words.

Physical time constraints, Untimely Event, Inopportune Action, Timed Checker, Tests

1 Introduction

The International Standardization Organisation(ISO) is interested in the design of Open Systems Interconnection (OSI) i.e. distributed systems integrating heterogeneous computer systems. For that purpose, a standardized architecture has been defined [15]. This architecture is composed of subsystems. OSI Protocol and Service *specifications* represent standards for these subsystems. They define the behaviour required from them. Coding a specification in an executable language provides an *implementation*. The behaviour of the latter can be observed in order to *check* that it satisfies some given properties. Moreover, *tests* can be performed. They consist of submitting some selected sequences of events to the implementation in order to check its behaviour. In the context of OSI, when the checking consists of knowing if an implementation satisfies (or *conforms to*) the specification, the tests are called *conformance tests*. It is important to note that Open Systems are just a particular type of distributed systems. So, methods used in the design of such systems (specification, testing techniques) are also suitable for other distributed systems.

Transition system models have demonstrated their ability to serve as a basis for OSI protocols specification [2], for trace checker derivation [6], for test sequences gen-

eration [3]. But *physical time* (sometimes called real-time) constraints are an important recognized aspect of specifications [2, 4, 9, 12], even if they are generally considered in terms of performance. In [11], the concept of physical time is defined. "Action *A* occurs after action *B*" specifies the temporal or logical ordering of actions *A* and *B*, while "Action *A* occurs at instant (or date) t_A and action *B* occurs at instant t_B " specifies a physical time aspect. In the latter, t_A and t_B are physical values (belonging to \mathbb{N} or \mathbb{R}) and $|t_B - t_A|$ is the duration between the two events (Note : event $\stackrel{\text{def}}{=}$ execution of an action at a given instant).

The checking [6, 5] and testing [3, 13, 14] of systems behaviour have mainly be studied according to logical time (ordering) aspects. We maintain that physical time is significant in the specification of a system, and therefore it must be taken into account for checking and conformance test. An example drawn from the Transport Protocol of ISO is provided to illustrate our point of view. In section 2, a Timed Petri Net (TPN) is proposed to model physical time aspects, after a short introduction of the modelling technique used. On the basis of this model, two kinds of invalid behaviours (untimely and inopportune outcomes) are characterized in section 3. Then, in the latter section, *Timed Checkers* are defined for checking time dependant behaviours. A case of Timed Checker is given. In section 4, we deal with a method for automatically deriving *Test sequences* in TTCN formats (see[3, 9]) from TPN oriented specification.

2 Timed Petri Nets for modelling Time dependant systems

2.1 A Transport entity example

Let us consider the behaviour of an OSI Transport implementation. A *CR* is a *Connection Request* and a *CC* is a *Connection Confirm*. A message sent (resp. received) is prefixed by a '!' (resp. '?'). The observed sequence $!CR.?CC$ is an authorized one. The sequence $!CR.!CR.?CC$ is also authorized. The second $!CR$ cor-

responds to a reexpedition after a timer (say T), set to the τ_T value, has expired. So its validity depends on the instant when it occurs, in other words, on the value of τ_T . In fact, τ_T is submitted to the following constraint : $\tau_T \geq A_{RL} + A_{LR} + A_R + X$, where

1. A_{RL} : Transit Delay of a PDU (Protocol Data Unit) from the Remote to the Local entity.
2. A_{LR} : Transit Delay of a PDU from the Local to the Remote entity.
3. X : Local entity computing Delay of a PDU.
4. A_R : Remote entity acknowledge Delay of a PDU.

An implementation which τ_T value is too small will needlessly reexpedit a CR . This example exhibits the need to specify physical time constraints and therefore to take them into account in checking and test processes.

2.2 Modelling time dependencies in system behaviour

A system is submitted to physical time constraints. To perform an action, it may take a minimal time τ_1 , and a maximal time τ_2 . So $[\tau_1, \tau_2]$ specifies a validity interval and $\Delta\tau (= \tau_2 - \tau_1)$ specifies an incertitude of the execution time of an action, which is useful for actions such as sending a message via a network. The transit delay of a message via a network is variable.

A technique for modelling the duration of an action in an abstract manner is to define its beginning (say τ_1) and end (say τ_2) instants, because saying that the execution of an action consumes the time τ is equivalent to saying that $\tau_2 = \tau_1 + \tau$. An abstract specification only concerns the external behaviour of a system. In this kind of specification, the only way to know that an action has been executed is to observe an event (i), and the observation of an event is instantaneous (ii). These two previous reasons justify our specification technique which focuses mainly on the end of the execution of an action. In this paper, the execution instant of an action will exactly designate the end of its execution. In the following section, we try to formalize this specification technique.

2.3 Timed Petri Nets

Several kinds of Timed Petri Nets (TPN) have been introduced to model time dependencies in computer networks design[8, 10, 16]. In this paper, we use a TPN introduced in[16] called the Timed Condition-Event Nets (TCE Nets) which is suitable for our time constraint modelling technique. Let us start with defining a Petri net.

Definition 1 A Petri Net (or simply a net) is a 4-tuple $R = (P, T, Pre, Post)$, where

1. P is a finite set of places
2. T is a finite set of transitions
3. $Pre : P \times T \rightarrow \mathbb{N}$ is the backward incidence function
4. $Post : P \times T \rightarrow \mathbb{N}$ is the forward incidence function.

A marked net is a 2-tuple (R, M) where R is a net and $M : P \rightarrow \mathbb{N}$ is a function called marking. $M(p)$ is the number of tokens carried by the place p . In the case of Condition-Event net (CE net), $M(p)$ is less or equal to 1. And the presence of a token in a place indicates that the condition corresponding to that place holds. A transition t is *fireable* for a marking M (which is denoted $M(t >)$)

if $\forall p \in P, M(p) \geq Pre(p, t)$.

The firing of a sequence s (a word from T^*) is inductively defined by the successive firing of each transition (letter from T) from s . From a given marking M , the firing of s leads the Petri Net to another marking M' (which is denoted $M(s > M')$).

Let us now include physical time constraints on the firing of transitions.

Definition 2 A Timed Petri Net is defined as a 3-tuple $R_T = (R, f_{min}, f_{max})$ where

1. $R = (P, T, Pre, Post)$ is a Petri Net
2. $f_{min} : T \rightarrow \mathbb{N}, f_{max} : T \rightarrow \mathbb{N}$ are two time functions.

The f_{min} and f_{max} functions define the τ_1 and τ_2 values of the previous section.

Firing rule of the TPN

Let M, t, τ be respectively a marking, a transition and an instant (a time value).

t is fireable for M at instant τ if

1. $M(t >)$ and
2. $f_{min}(t) \leq \tau < f_{max}(t)$.

Note

1. The firing of t may only occur when the time τ has been elapsed after the reaching of M , which is also denoted $M((t, \tau) >)$. This firing is instantaneous, so the next marking of the net is instantaneously reached.

2. The TPN previously defined induces a net $(P, T_{\mathcal{T}}, Pre, Post)$, where the set of transitions is $T_{\mathcal{T}} = \{(t, \tau) \in T \times \mathbb{N} / f_{min}(t) \leq \tau < f_{max}(t)\}$. Let M_0 be the initial marking of a TPN $R_{\mathcal{T}}$. The following set defines all the possible sequences of events accepted by $R_{\mathcal{T}}$:
- $$L(R_{\mathcal{T}}, M_0) = \{s \in T_{\mathcal{T}}^* / M_0(s) > \}$$

A TPN can serve as a specification modelling the behaviour of a system. The set of events is $T_{\mathcal{T}}$. Each marking represents a global state of the system. The firing of a transition (resp. execution of an action) leads the TPN (resp. the system) from a marking (resp. a state) to another marking (resp. another state).

For protocol entities, physical time constraints are generally specified for outputs (e.g. !CR) only. After the reception of a given input ?a (or after the reaching of a given state), an output !b must be sent after (or within) a number of time units. Such specifications are typical in the Estelle language [7, 4]. The following Estelle transition states that

when State1 is reached, output B must be sent but no earlier than 3 time units.

```
FROM State1 TO State2
PROVIDED TRUE
DELAY(3)
BEGIN
```

```
...
OUTPUT B
```

```
END
```

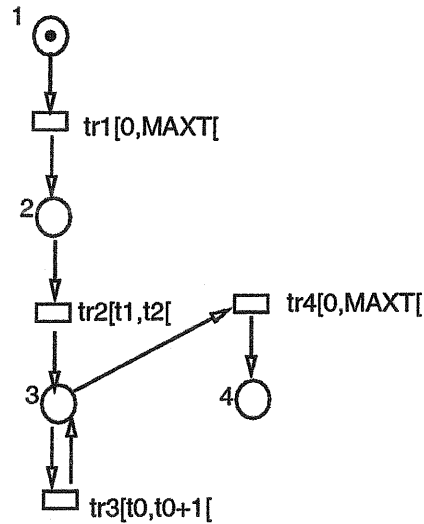
In fact, input actions do not depend on the protocol entity. They can occur at any time (one can associate them with the time interval $[0, \infty[$). But in practice, if a protocol entity does not receive the expected interaction within a given time, it executes some particular action. For instance, in the Transport entity example (section 2.1), a CR is reexpedited. In some cases, the protocol entity enters a disconnection phase after a "default response time" has been elapsed. So the default validity time interval of input interactions can be $[0, MAXT[$, where MAXT is a default response time. Figure 1 is a TPN which gives a simplified specification of a Transport entity.

The following section deals with observation and checking of implementations specified by TPNs.

3 Timed Checkers

3.1 The concept of Checker

A *checking process* consists of observing a system and checking if its behaviour satisfies a given property. This process is performed by a module called *Checker*. In [6], such a module is called *Trace Checker* while in [5]



Transition	Corresponding ASP or PDU
tr1	?TCOnreq
tr2	!CR
tr3	!CR
tr4	?CC

ASP Abstract Service Primitive

PDU Protocol Data Unit

Figure 1: TPN modelling a simplification of the Transport Protocol

it is called *Observer*. The checkers defined in both [6] and [5] only check logical time properties. They are characterized by sets of traces. Let \mathcal{L} be some set of valid traces of a specification. A trace checker can be derived by an automaton which recognizes the words (traces) corresponding to the elements of \mathcal{L} . Given a marked Petri net (R, M_0) , a Checker can therefore be characterized by the set $L(R, M_0)$, where $L(R, M_0)$ is the set of all possible sequences of actions executable by (R, M_0) .

Physical time constraints can also be checked. We call *Timed Checker* a module which includes physical time constraints checking. In section 3.3, such checkers are characterized from specifications based on TPNs.

First of all let us characterize two kinds of invalid outcomes that violate the expected behaviour specified by a TPN.

3.2 Untimely & Inopportune Outcomes

Let $R_{\mathcal{T}} = (P, T_{\mathcal{T}}, Pre, Post)$ be a TPN ($T_{\mathcal{T}} \subset (T \times \mathbb{N})$).

An event is said to be *untimely* for $R_{\mathcal{T}}$ if $R_{\mathcal{T}}$ has reached a marking for which this event is not accepted. Formally, let M be the current reached marking of $R_{\mathcal{T}}$.

Definition 3 *The set of untimely events of $R_{\mathcal{T}}$ is given by*

$$UNTIMELY(R_{\mathcal{T}}, M) = \{(t, \tau) \in T_{\mathcal{T}} \mid M((t, \tau) > \text{is false})\}$$

Example

Let us consider the TPN of figure 1.

Let $M = (0, 0, 1, 0)$ be a marking of this TPN (We have $M(1) = 0$; $M(2) = 0$; $M(3) = 1$; $M(4) = 0$), where the set of places is $P = \{1, 2, 3, 4\}$). Once M is reached, if a $!CR$ is executed before t_0 time units (e.g. at t_0-1), this execution is untimely. So $(!CR, t_0-1) \in UNTIMELY(R_{\mathcal{T}}, M)$.

It is also possible to characterize a set of actions which can never be executed for the current marking of $R_{\mathcal{T}}$. Such actions are called *inopportune*.

Definition 4 *The set of inopportune actions of $R_{\mathcal{T}}$ is given by*

$$INOPPORTUNE(R_{\mathcal{T}}, M) = \{t \in T \mid \forall \tau \in \mathbb{N}, (t, \tau) \in UNTIMELY(R_{\mathcal{T}}, M)\}.$$

If an action is inopportune any execution of it will be untimely.

Example

From the specification of figure 1, when $M = (0, 0, 1, 0)$ is reached, only $!CR$ and $?CC$ are executable. Any other action is inopportune. For instance a $!DT$ (Data request) is not allowed for this marking. So $!DT \in INOPPORTUNE(R_{\mathcal{T}}, M)$.

From the previous examples we get

$!CR \notin INOPPORTUNE(R_{\mathcal{T}}, M)$ but

$(!CR, t_0-1) \in UNTIMELY(R_{\mathcal{T}}, M)$. That shows that even if an action is not inopportune, its execution at some instants can be untimely.

3.3 A case of Timed Checker characterization

Let us consider a specification given by marked TPN $(R_{\mathcal{T}}, M_0)$. In order to check the validity of systems executions according to $(R_{\mathcal{T}}, M_0)$, we characterize a Timed Checker by the set $L(R_{\mathcal{T}}, M_0)$. Any suite of executions not belonging to that set is invalid.

The Timed Checker characterized has the power of detecting any untimely event.

Proof

Let M be the current marking of $(R_{\mathcal{T}}, M_0)$.

$\exists s \in L(R_{\mathcal{T}}, M_0)$ such that $M_0(s > M)$. (M is reachable from M_0)

Let $\mu = (t, \tau)$ be an event.

$\mu \in UNTIMELY(R_{\mathcal{T}}, M)$

$\iff M(\mu > \text{is false})$

$$\iff \begin{cases} \exists s \in L(R_{\mathcal{T}}, M_0) \text{ such that } M_0(s > M \\ \wedge \\ s.\mu \notin L(R_{\mathcal{T}}, M_0) \end{cases}$$

□

An untimely event violates the specification by involving a suite of executions which is not authorized.

In the following section we present test methods aimed to check the validity of an implementation with respect to a given specification, in the context of conformance. The proposed methods permit untimely and inopportune outcomes to be detected.

4 Test sequences generation in TTCN format

Several methods have been proposed for deriving test suites from Finite States Machine (FSM) based specifications [3, 13, 14]. The reader may report to [3] for a general view of our test derivation methodology in TTCN formats.

There also exist methods to derive test suites from Petri nets models[1]. Such methods generally consist of deriving an automaton from the reachability graph of the Petri net, and then deriving the tests from that automaton.

To derive tests from a marked net (R, M_0) , the following steps can be used :

1. calculate the reachability graph of the marked net (R, M_0)

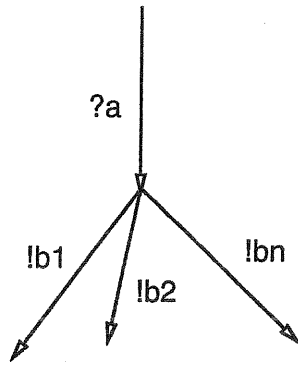


Figure 2: Several possible outputs b_1, b_2, \dots, b_n upon reception of input a

2. derive (with classical algorithms) the reduced automaton from the reachability graph such that $L(R, M_0)$ is the set of *words* recognized by this automaton
3. rename the transitions by the corresponding Service (or Protocol) Data Unit name as it is defined in the standard, while adding a prefix '?' or '?' for sending and receiving actions respectively.
4. as we proposed in [3], the derivation of test suites from FSM in TTCN format is performed by inverting the prefixes ('!' becomes '?' and vice versa) while adding L or U which corresponds to the Lower or Upper interface with the tester. So an output of the specification becomes an input of the tester and vice versa.

In order to take physical time constraints into account, the instants at which actions occur must be checked. The inclusion of validity time interval in the test derivation procedure (see the 4 steps above) is made by adding a step 3' after step 3 :

3'. associate the transitions of the automaton with the validity time interval corresponding to the transitions in step 3.

The test derivation method presented in [3] did not include physical time constraints. That is done in this section.

In the test cases generation, interactions are sent to the IUT (Implementation Under Test) without delay, and the instants at which outputs are generated are checked. After the reception of an input (?a), if several outputs (!b1, !b2, ... !bn) are possible (see figure 2), they become alternatives in the TTCN test tree. The derivation of TTCN test cases has the structure given by figure 3.

1. As soon as an input (L or U)!a is sent (by the tester) to the IUT, a timer T is started with a Tmax value. Tmax must be greater than $f_{max}(tr)$

BEHAVIOUR DESCRIPTION	VERDICT
(L or U)!a START T, Tmax	
(L or U)?b1 READTIMER T, Curtime [Curtime < $f_{min}(tr_1)$ OR Curtime >= $f_{max}(tr_1)$] +subtree 1(of the end of the sequence)	FAIL (P)
(L or U)?b 2 READTIMER T, Curtime [Curtime < $f_{min}(tr_2)$ OR Curtime >= $f_{max}(tr_2)$] +subtree 2(of the end of the sequence)	FAIL (P)

(L or U)?b n READTIMER T, Curtime [Curtime < $f_{min}(tr_n)$ OR Curtime >= $f_{max}(tr_n)$] +subtree n(of the end of the sequence)	FAIL (P)
?OTHERWISE	FAIL
?TIMEOUT T	FAIL

Figure 3: General structure of a test case including physical time constraints

i) ($i=1, n$), where tr_i is the transition corresponding to output (L or U)?b_i (see step 3') of the test derivation procedure.

2. Whenever an output (L or U)?b_i is received by the tester, the current value of the timer T (say Curtime) is read and compared to the [$f_{min}(tr_i), f_{max}(tr_i)$] interval. If this current value does not match the interval, the event (!b_i, Curtime) is untimely, therefore a FAIL is declared in the verdict part of the TTCN table tree.
3. If an output other than the ones specified (i.e. b₁, b₂, ... b_n) is received, then such an output is inopportune and a FAIL is declared.
4. If T expires and nothing has occurred, a FAIL is declared too.

The table of figure 4 is an application of the method to the specification of figure 1.

5 Conclusion

In this paper we proposed a type of Timed Petri Net for modelling both ordering and physical time dependencies in system behaviour. Two kinds of invalid behaviour (untimely and inopportune) which enable the error detection power to be improved are identified on the basis of our model. We then introduced the notion of Timed Checkers. The role of the latter is to check that a behaviour satisfies properties including physical time constraints. A Timed Checker is characterized from TPN oriented specifications. It has the power of

Objective : to test that the IUT will send a PDU CR at a time comprised between t1 and t2, upon reception of an ASP TCONreq	
BEHAVIOUR DESCRIPTION	VERDICT
U!TCONreq START T,Tmax L?CR READTIMER T,Curtime [Curtime < t1 OR Curtime >=t2] +subtree of the end of the sequence	FAIL (P)
?OTHERWISE	FAIL
?TIMEOUT T	FAIL

Figure 4: Test case derived from the specification of figure 1

detecting the untimely and inopportune outcomes defined below. Such outcomes can also be detected by test sequences which automatic derivation method is proposed.

A main direction of our current research is the applicability of our methods to check the interoperability of systems[9].

References

- [1] B.Baumgarten, A.Giessler, R.Platten. Test Derivation from Nets Models. 2nd International Workshop on Protocol Test Systems. Berlin. October 1989.
- [2] G.v.Bochmann. Protocol Specification for OSI. In Computer Networks and ISDN Systems 18 (p. 167-184). North Holland 1989-1990
- [3] R.Castanet, R.Sijelmassi. Methods and Semi-automatic Tools for Preparing Distributed Testing. In 6th IFIP International Workshop on Protocol Specification Testing and Verification. Montreal, Gray rock. North Holland. June 1986.
- [4] Samuel C. Chamberlain. Paul D. Amer. Formal Specification of Real-Time Constraints in Estelle. in Réseaux et Informatique Répartie 2(pp. 113-134). Hermes 1992.
- [5] R.Dssouli. Etudes des méthodes de test pour les implantations de protocole de communication basées sur les spécifications formelles. Ph D Thesis. Univ. Montreal September 1986.
- [6] C.Jard, O.Drissi. Deriving trace checkers for Distributed Systems. Publication interne IRISA no 347. Rennes, Février 1987.
- [7] ISO9074. Information Processing Systems. Open Systems Interconnection-Estelle (Formal Description Technique based on Extended State Transition Model. 1989.
- [8] P.Merlin, D.J.Farber. Recoverability of Communications Protocols. IEEE Transactions on Communications, vol. COM-24, no 9, Sept 1976.
- [9] O.Koné. Méthodes de Test de Protocoles. Proc. 1st African Conference on Research in Computer Science. Yaoundé. October 1992.
- [10] C. Ramchandani. Analysis of Asynchronous Concurrent Systems by Timeds Petri Nets. Project MAC, TR 120, Massachussets Institute of Technology, Feb. 1974.
- [11] M.Raynal. La communication et le temps dans les Réseaux et les Systèmes Répartis. Tome 1. Editions Eyrolles, Janvier 1991.
- [12] Information Processing Systems - OSI Conformance Testing Methodology and Framework. ISO/IEC JTC 1/ SC 21. DIS 9646, Parts 1-5, Mar.1989.
- [13] K.Sabnani. A.Dahbura. A protocole Test Generation Procedure. Computer Networks and ISDN Systems 15. 1988. pp 285-297.
- [14] D.Sidhu. T.Leung. Formal Methods for Protocol Testing : a Detailed Study, IEEE Transactions on Software Engineering. vol 15, n.4. April 1989.
- [15] Special Issue on Open Systems Interworking. Proc. IEEE. December 1983.
- [16] B.Walter. Timed Petri Nets for Modelling and Analyzing Protocols with Real-Time Characteristics. Proc. 3rd IFIP Workshop on Protocol Specification Verification and Testing. Edited by H.Rudin and C.H.West. Switzerland 1983.