Timed-pNets: A formal communication behavior model for real-time CPS system *

Yanwen Chen[1,2], Yixiang Chen[1], and Eric Madelaine[2]

[1] MoE Engineering Research Center for Software/Hardware Co-design Technology and Application, East China Normal University, Shanghai, China
[2] INRIA, CNRS, I3S, University of Nice Sophia-Antipolis, Sophia Antipolis, France

Abstract. We propose a semantic model named timed-pNets to define hierarchical structures for CPSs as well as its communication behaviors semantic with time constraints. Logical clocks relations are introduced to describe the partial order of event occurring. After setting (time)-boundaries and designing properties, we use the TimeSquare tool to simulate the system and check its properties.

1 Introduction

Nowadays, cyber physical systems have received much attention since the next generation of computing revolution is integration of computation, control and communication [7]. In CPSs, heterogenous embedded devices are connected by wire or wireless networks to communicate among each other via sets of sensors and actuators. One typical application domain is intelligent transportation systems (ITS), in which cars and infrastructures are equipped with sensors and actuators. They communicate with each other to update physical information and accomplish remote controlling. Currently, Research and Innovative Technology Administration (RITA)[10] in U.S. Department of Transportation has started research work on it to achieve a vision of national transportation by feature a connected transportation environment among vehicles, infrastructures and passengers' portable devices. They raised the importance of real-time communication among these distributed nodes since the data out of date would make big mistakes even sometime could lead to car accidence. For example, the delay of sending global traffic information to cars may result to a wrong guiding for car to choose its best way. Also the delay of information exchanging among cars may cause a car accidence especially when they cannot see each other at cross.

Our aim is to build a low-level semantic model for the system and then analyse its properties. Since each distributed device has its own clock(s) and the communication delay between devices is uncertain, synchronous and asynchronous communication behavior will be considered in our model. We propose timed-pNets, which is an extension of pNets [2], to describe the communication behavior by adding logic clock relation. Time-pNets absorbs many advantages of pNets

* This work was partialy funded by the Associated Team DAESD between INRIA and ECNU, Shanghai; by ECNU for overside visiting funding, China 973 project(No.2011CB302802) and NSFC(No.610211004).

such as well hierarchical structure, flexible communication models, compact format, expressiveness, etc. By introducing clock relations which is defined in CCSL [8], the asynchronous communication behavior can be well defined which leads to well checking for real-time properties. Finaly, we use the TimeSquare tool to simulate the clock relation of models and then check the correction of time logic as well as properties.

The main contributions of our work in the paper include: 1) Introduce logical clocks into pNets, 2) Propose a model language timed-pNets , 3) Use timed-pNets to model ITS, 4) Use TimeSquare to simulate clock relations and check its properties.

In section 2 we introduce the related work. In section 3 we give the definition of timed-pNets. In section 4, we propose a simple use case and describe how we use timed-pNets to model it. In section 5, we use time square to simulate the clock relations of the system and check its properties. In the last section, we give an conclusion for our current work.

2 Related Work

Prior research works related to building model for ITS applications include:

- Timed-automata[1] can be used to model the behavior of real-time systems. They provide a simple, and yet powerful, way to annotate state-transition graphs with timing constraints using finitely many real-valued clocks. Closure properties, decision problems as well as automatic verification of real-time requirements of finite-state systems are considered in timed-automata, and are supported by a number of tools, e.g. in UPPAAL [4].
- Globally Asynchronous Locally Synchronous (GALS) systems [9] combine the benefits of synchronous and asynchronous systems, in which each embedded node is modeled as a FSM (Finite State Machine) and the communication between them as buffers. This architecture provides a methodology for combining concurrent embedded systems within loosely coupled systems.
- The BIP framework [3], which can be used to model heterogeneous real-time components. BIP provides a powerful mechanism for structuring interaction for layered components of which synchronous and time systems are particular classes. The BIP framework produces a very fine grained formal computational model of the system functional level and executes the model semantics at runtine. In parallel, recent evolutions of the BIP framework allow the use of timed models and provide a real-time BIP Engine.
- Spatio-temporal consistence language(STeC) [5], which provides a location-triggered specification as well as operational semantics for describing distributed system with time and location constraints.

Compared to these previous works, our appraoch uses logical clocks instead of physical clocks, so that our approach is flexible enough to describe a communication delay by clock relation rather than by concrete time units. This gives us a very flexible way to specify interaction of systems with different clocks.

3 Timed-pNets

In this section, we define Timed-pNets, which are an extension of pNets (parameterized networks of synchronized automata), a very expressive and flexible semantic model developped by the Oasis team at INRIA for the modeling and verification of (untimed) distributed systems [2].

Definition 1 (Timed-Actions). *Let $\mathcal{T}$ be a set of discrete time variables taken from non-negative natural numbers $\mathbb{N}$. $\mathcal{B}_{\mathcal{T}}$ is the set of boolean expressions (guards) over time variables and $\mathcal{L}_{\mathcal{A},\mathcal{T}}$ is an action set built over $\mathcal{T}$, in which each action has a free time variable $t \in \mathcal{T}$. We call $a^{t|\mathcal{B}} \in \mathcal{L}_{\mathcal{A},\mathcal{T}}$, with $\mathcal{B} \in \mathcal{B}_{\mathcal{T}}$ a timed action, in which t describes a* time delay *before the action can be executed.*

We set $a^0 = a$ that means the action a is always ready. As an example, $a^{t|1 \leq t \leq 3}$ means the action a cannot be executed until t units times are passed.

Logical Clocks of timed-Actions Logical clocks [8] represent a relaxed form of time where any events can be taken as a reference for counting. It can be used for specifying classical and multiform real-time requirements as well as formally specifying constraints on the behavior of a model.

Definition 2 (Logic Clocks). *A clock C_a consists of an infinite set of discrete ticks of timed-action a^t. We write $C_a = \{(a^t)\_1, (a^t)\_2, ..., (a^t)\_k, ...\}(k \in \mathbb{N}, a^t \in \mathcal{L}_{\mathcal{A},\mathcal{T}})$, in which $(a^t)\_k$ denotes the k^{th} instance of clock C_a.*

Clock Constraints Clock constraints are predicates built from binary relations between clock expressions. We take the syntax and semantics of clock relations from [8], which is a language to express multi-clock time specifications by defining clock relations of time models for real-time systems. The clock relations include: $\subseteq$ (subclock), $\sharp$ (exclusion), $=$ (coincidence), $\prec$ (strict precedence),$\preceq$(precedence) and defined as:

- $a_1 \subseteq a_2$ (a_1 is a subclock of a_2), which means each instance of a_1 must be coincident with an instant of a_2.
- $a_1 \sharp a_2$ (a_1 excludes a_2), which means none of their instances coincide.
- $a_1 = a_2$ (a_1 coincides with a_2), which means the action a_1 ticks if and only if the action a_2 ticks.
- $a_1 \prec a_2$ (a_1 strictly precedes a_2), which means for every instant k ($k \in N$), the k^{th} instant of a_1 strictly precedes the k^{th} instant of a_2.
- $a_1 \preceq a_2$ (a_1 precedes a_2), which similar to the previous one. The only different is the action a_1 can tick as late as when a_2 ticks.
- New relations or expressions can be define by combining these basic relations with arithmetic and boolean. For example: $a_1 - a_2 \prec a_3$

The next definition extends the classical pNets definition from [2] with timed-actions and clock constraints. From pNets, we retain the hierarchical structure that is essential in structuring our heterogeous systems, but also the parameterization of subnets: holes in a pNet can be instanciated by a variable number

of subnets (as e.g. a number of cars in the forthcoming case-study). Then synchronisation vectors allow very flexible and expressive multi-way synchronisation mechanisms, that naturally we extend here with clock constraints.

Definition 3 (Timed-pNets). *A Timed-pNet is a tuple* $< P, A_G, R_G, J, C, \widetilde{O}_J, \widetilde{R}_J, \overrightarrow{V} >$, *where:*

- $P = \{p_i / p_i \in Dom_i\}$ *is a finite set of parameters*
- $A_G \subseteq \mathcal{L}_{\mathcal{A},\mathcal{T}}$ *is a set of global actions*
- C *is a set of clocks for all actions*
- R_G *is a set of relations between actions taken from each subnet*
- J *is a countable set of argument indexes: each index* $j \in J$ *is called a hole and is associated with a sort* $O_j \subseteq \mathcal{L}_{\mathcal{A},\mathcal{T}}$ *and a set of clock constraints* $\widetilde{R}_J$
- $\overrightarrow{V} = \{\overrightarrow{v}\}$ *is a set of synchronous vectors of the form:*
  - *(binary communication)* $\overrightarrow{v} =< ..., !a^{t1}_{[k_{i1}]}, ..., ?a^{t2}_{[k_{i2}]}, ... >\rightarrow (a^{t_g}_g)$, *in which* $a^{t_g}_g \in A_G, k_{i1} \in Dom_1, k_{i2} \in Dom_2, !a^{t1} \in O_{i1}, ?a^{t2} \in O_{i2}, t_g = max\{t1, t2\}$
  - *or (visibility)* $\overrightarrow{v} =< ..., a^{t1}_{[k1]}, ... >\rightarrow (a^{t_g}_g)$, *in which* $a^{t_g}_g \in A_G, k1 \in Dom_1, a^{t1} \in O_{i1}, t_g = t1$

4 Case Study

In this part we illustrate our approach with a simple use case called speed controlling system taken from [10]. Here cars' speed are monitored by infrastructure that collects information from cars and sends brake signal back to cars under some global decision procedure. The communication protocol is described as following:

- Cars send heartbeat signals "I'm here" with parameters "(location, speed)".
- Infrastructure collects heartbeat signals from cars.
- Infrastructure sends "brake" signal to the car if it is over the speed limitation.
- A car reduces its speed when it gets the "brake" signal.

Fig. 1 presents its architecture in which cars and infrastructures are distributed nodes. A cars consists of three sub components: a sensor, a controller and a brake system. The car sensor is used to detect its current location and speed and to receive control signals received from the infrastructure. The Car controller gets signals from the sensor and then call the relevant systems to execute brake operations. The local communications between the sub components of cars are synchronized, which means that the sending event and receiving event coincide. These sub components' LTS are shown in Fig. 1. The car sensor is modeled by two LTSs: one defining the periodical sending of heartbeat signals to report its location and speed, the other describing its reactions to control signals.

Now, we describe how to use timed-pNets to model the communication behavior and how to build clock relations in and between components. By lack of space we only explain the top-level synchronisation elements, and the behaviour of the car's subprocesses.

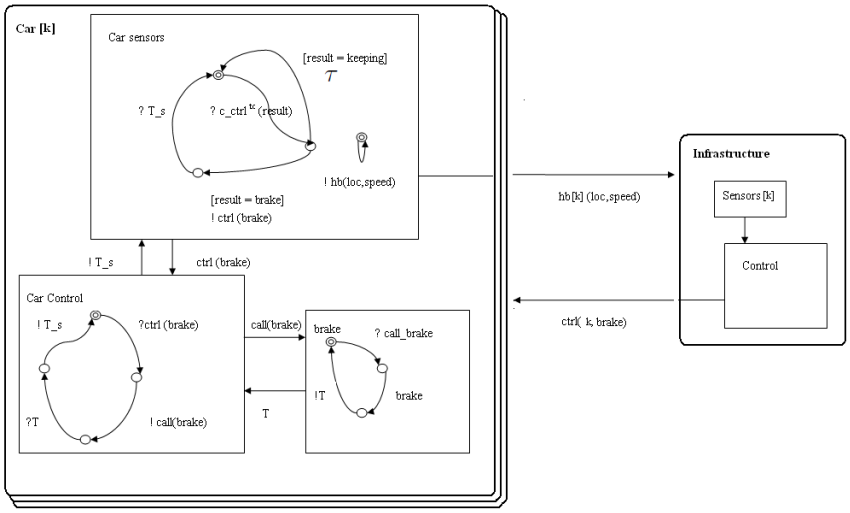

Fig. 1. Timed-pNets architecture with details of the car's subprocesses

4.1 Formalisation of timed-pNets Architecture

The building of timed-pNets is hierarchical. For our exemple, we generate separately a pNet structure for the toplevel assembly, and for each of the Car and Infrastructure components. The top-level pNet has 2 holes, the first one receiving an arbitrary number of Cars, the second one a single Infrastructure. Within each "second layer" pNet local communications will be synchronous, while at toplevel communications are asynchronous. This shows in the following pNet, where clock relations on the "hb" and "ctrl" links are defined as precedence.

$< P = \{k : \mathbb{N}\}, A_G, R_G, J, \widetilde{C}_J, \widetilde{O}_J, \widetilde{R}_J, \overrightarrow{V} >$
$A_G = \{CI\_hb^{t_h}(k, loc, speed), CI\_ctrl^{t_{ct}}(k, brake)\}$
$J = \{car[k], infrastructure\}$
$O_{Car} = \{!c\_hb^{t_{hb\_c}}(loc, speed), ?c\_ctrl^{t_{ct\_c}}(brake), !call(brake), T_s, ...\}$
$O_{Infrastructure} = \{?I\_hb^{t_{hb\_I}}(k, loc, speed), !I\_ctrl^{t_{ct\_I}}(k, brake), ...\}$
$R_G = \{!c\_hb^{t_{hb\_c}}[k](loc, speed) \prec ?I\_hb^{t_{hb\_I}}[k](loc, speed);$
$\quad !I\_ctrl^{t_{ct\_I}}(k, brake) \prec ?c\_ctrl^{t_{ct\_c}}[k](brake);\}$
$\overrightarrow{V} :< O_{car}[k], O_{infrastrcuture} > \rightarrow A_{Car\_infrastrcuture}$
$\quad =< !c\_hb^{t_{hb\_c}}[k](loc, speed), ?I\_hb^{t_{hb\_I}}[k](loc, speed) > \rightarrow CI\_hb^{t_h}(k, loc, speed);$
$\quad\quad < ?c\_ctrl^{t_{ct\_c}}[k](brake), !I\_ctrl^{t_{ct\_I}}(k, brake) > \rightarrow CI\_ctrl^{t_{ct}}(k, brake).\}$

An interesting point is that the Infrastructure receives independent heartbeats for the Cars, that are subsequently interleaved within the Infrastructure structure. This is expressed by a clock relation on the link between Infrastructure sensors and control: $?I\_hb^{t_{hb\_I}}[k](loc, speed) \subseteq !hb_I^{t_I}(k, loc, speed)$, telling the each Car heartbeat clock transmitted by the Sensor is a subset of the (single) heartbeat clock received by the Control.

Finally, we give an example of the set of Clock constraints that we generate from a pLTS, here for the Car Sensor sub component:

$R_{CarSensor} = \{hb(loc, speed) \triangleq idealClockdiscretizedByrate$ (1);
$(\tau \sharp !ctrl(brake))$ (2);
$?c\_ctrl(result) \prec (\tau \wedge !ctrl(brake))$ (3);
$!ctrl(brake) \prec ?T\_s$ (4);
$(?T\_s[i] \vee \tau[i]) \prec ?c\_ctrl(result)[i+1]$ (5);$\}$

where (1) describes that the heartbeat signal is sent periodically; (2) indicates that events τ and $!ctrl(brake)$, from different paths of the same LTS, are exclusive; (3) denotes that the event $?c\_ctrl(result)$ always precedes the event τ and $!ctrl(brake)$; (4) tells us that event $!ctrl(brake)$ precedes the event $?T\_s$; (5) explains that the events in the i^{th} cycle precedes those in the $(i+1)^{th}$ cycle.

5 Simulation

We use TimeSquare [6] to simulate the clock relations and check its logic correction. The input of TimeSquare is a CCSL file including clock relations, bound requirements and properties. The tool proceeds with a symbolic simulation, and generates a trace model (one partial order satisfying the specification). Output files (text and graph) are generated to display the traces and eventually show the property violations.

In our use-case the clock relations generated from the pNets model. Then we representing the communication and computation delays by fixing delay bound in timed-action guards. We set heartbeat interval $hi = hb\_(i+1) - hb\_i$. The minimum and maximum communication delay between car and infrastructure is set as $(1/5)hi$ and $(3/5)hi$. For the computation delay, we assume that it takes at most $(2/5)hi$ for each action transition so that for instance we can set $(?c\_ctrl(result) - ?T\_s) \preceq (2/5)hi$. For deadline, assuming each heartbeat signal should be processed before sending next heartbeat, then for each action a_i, we set $deadline = hb\_(i+1)$. All these boundaries are merged into the TimeSquare CCSL input file. We expressed a "boundary liveness property" for this simulation to see if the system satisfies the real-time property under the hypothesis of boundaries. $(?T\_s\_(i) \prec hb\_(i+1)) \wedge (\tau\_i \prec hb\_(i+1))$ denotes each heartbeat signal finally will be processed before deadline.

After simulation, we got the result as Fig.2, which shows that the real-time property is not satisfied since the clock !ctrl(brake) is over the deadline. The boundary condition we set in previous part was too large. After we modify the maximum computation boundary, the simulation shows no more violations.

6 Conclusion

In this paper, we have added time constraints to the pNets behavior semantic model. In timed-pNets, logical time relations in lower-level (synchronous) components are derived from the corresponding label transition systems. Then

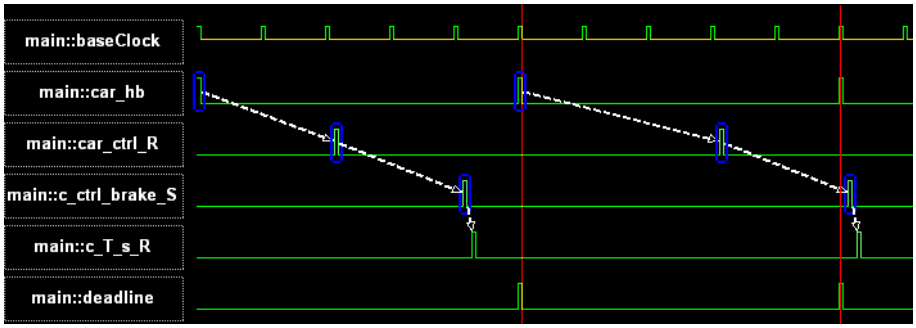

Fig. 2. property checking

communication delays, processing delays and required global properties are defined by the user. We illustrate our approach on a simple use case from Intelligent Transport Systems, show how our Timed-pNets models are constructed, and how timed properties are validated through simulations in the TimeSquare tool.

References

1. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183 – 235, 1994.
2. T. Barros, R. Ameur-Boulifa, A. Cansado, L. Henrio, and E. Madelaine. Behavioural models for distributed fractal components. *Annals of Télécommunications*, 64(1-2):25–43, 2009.
3. Ananda Basu, Marius Bozga, and Joseph Sifakis. Modeling heterogeneous real-time components in BIP. In *Fourth IEEE International Conference on Software Engineering and Formal Methods (SEFM 2006), 11-15 September 2006, Pune, India*, pages 3–12. IEEE Computer Society, 2006.
4. J. Bengtsson, K. G. Larsen, F. Larsson, P. Pettersson, and Wang Yi. Uppaal — a Tool Suite for Automatic Verification of Real–Time Systems. In *Proc. of Workshop on Verification and Control of Hybrid Systems III*, LNCS 1066, pages 232–243. Springer–Verlag, October 1995.
5. Yixiang Chen. Stec: A location-triggered specification language for real-time systems. In *ISORC Workshops*, pages 1–6. IEEE, 2012.
6. Julien Deantoni and Frédéric Mallet. TimeSquare: Treat your Models with Logical Time. In *TOOLS - 50th International Conference on Objects, Models, Components, Patterns - 2012*, LNCS 7304, pages 34–41, Prague, 2012. Springer.
7. E.A. Lee. Cyber physical systems: Design challenges. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pages 363 –369, may 2008.
8. Frédéric Mallet. CCSL: specifying clock constraints with UML/MARTE. *Innovations in Systems and Software Engineering*, 4(3):309–314, 2008.
9. J. Muttersbach, T. Villiger, and W. Fichtner. Practical design of globally-asynchronous locally-synchronous systems. In *Advanced Research in Asynchronous Circuits and Systems (ASYNC 2000)*, pages 52 –59, 2000.
10. Intelligent transportation systems. http://www.its.dot.gov/research.htm.