IEEE Region 10 Conference on Computer and Communication Systems, September 1990, Hong Kong

Specification and Verification of Timing Properties of Distributed Real-Time Systems

R. Mall
Dept. of Computer Science and Automation
Indian Institute of Science
Bangalore, 560 012, INDIA.

L.M. Patnaik
Microprocessor Applications Laboratory,
Supercomputer Education and Research Centre, and
Dept. of Computer Science and Automation
Indian Institute of Science
Bangalore, 560 012, INDIA.

Abstract: There are no proficient formal tools available till now for specifying and analyzing the timing properties of distributed real-time systems. In this paper, we present a formalism for specification and analysis of timing properties of distributed real-time systems. This formalism is based on the Real-Time Logic of Jahanian and Mok, and it incorporates multiple occurrence functions to take care of the multiple unsynchronized clocks in a distributed system. The fundamental constraints that a distributed real-time system need to satisfy for the correctness of any formal reasoning about its timing properties have been investigated.

1. Introduction

It is imperative for real-time systems to satisfy total safety requirements due to the critical nature of their applications. Also, most of the real-time computer applications are inherently distributed in nature, e.g., automated manufacturing plants, nuclear power plant control systems, embedded spacecraft guidance and control systems, and many process control applications. These systems are usually required to satisfy a number of timing constraints, and any failure to meet a constraint may lead to disastrous consequences. However, the various timing errors that can creep into the design a real-time system, are often insidious, and in general, are hard to detect using experimental validation techniques – since the domain of the test parameters can be very large. Hence, formal techniques are vital to the validation of the design of a real-time system.

But, unfortunately, till now there are no proficient formal tools available for specifying and analyzing the timing properties of distributed real-time systems [7]. Of the handful of formal methods that are available, each one has some serious pitfall or the other. For example, the clock advancement problem of the temporal logic formalism (e.g., [2], [3], [6]) is a subtle one, and makes it difficult to use temporal logic to reason about real-time systems. Also, any formalism based on the assumption of the existence of a global clock cannot conceptually be used to reason about the timing properties of distributed systems having multiple unsynchronized clocks. Other related work includes a set of specification languages, e.g., LUSTRE [9], ESTEREL [10], and SIGNAL [11], that are designed for synchronous systems, and hence unsuitable for many practical applications.

In this paper, we present a formalism for specification and verification of the timing properties of distributed real-time systems. This formalism is based on the Real-Time Logic of Jahanian and Mok [1], and it incorporates multiple occurrence functions to take care of multiple unsynchronized clocks in a system. Some of the basic concepts of the RTL formalism have been generalized for extending its application to distributed real-time systems. The constraints that must be imposed on a distributed real-time system for the correctness of any formal reasoning about its timing properties have been investigated.

This paper is organized as follows. Section 2 discusses the specification of distributed real-time systems in the event-action model. Section 3 presents the Distributed Real-Time Logic (DRTL). Section 4 discusses the inference mechanism, and Section 5 concludes this paper.

2. Specification in Event-Action Model

In the event-action model, a distributed real-time computing system is specified by enumerating the events and actions that the system participates, the temporal ordering of these events and actions, and the timing constraints on them. The basic entities of this model, in terms of which a distributed real-time system can be described are as follows.

NODE: A node is a collection of events and actions associated with a common time base. It is the basic unit of composition in the design of a distributed system.

ACTION: Actions are the basic units of computation. Each action has a distinct starting and ending time marked by the corresponding start and stop events. Composite actions are composed from primitive actions, and are obtained by using any of the following constructs: X;Y is used to denote the sequential execution of X followed by execution of Y, X||Y denotes parallel execution of actions X and Y, and X!Y means that the action X must complete execution before the action Y can start execution (used for synchronization of actions).

Actions can be either periodic or aperiodic. A periodic action is executed at specific intervals as long as some state predicate evaluates to true. The timing constraint as-

CH2866-2/90/0000-274 $1.00 © 1990 IEEE

sociated with a periodic action can be represented by the following construct:

While < state predicate > **execute** < action >
with period = < *time* >, **deadline** = < *time* >, **delay** = < *time* >

Here, the delay clause is the constraint on the starting time of the action, and the deadline clause is the constraint on the completion time of the action.

EVENTS: The occurrence of an event marks a point of time that is of importance in describing a system's behaviour. The various events can be partitioned into the following six classes:

i) *External Events*: An external event occurs as a result of some action of the environment on the embedded system.

ii) *Start Events*: A start event marks the initiation of an action.

iii) *Stop Events*: A stop event marks the completion of an action.

iv) *Transition Events*: A transition event marks the change in the truth values of certain state predicate(s) of the system.

v) *Notifier Events*: A notifier event marks the time when a node wants another node to become aware of the occurrence of some event(s).

vi) *Notification Events*: A notification event marks a point in time, when a node becomes aware of the occurrence of the corresponding notifier event in another node.

STATE PREDICATE: A state predicate is an assertion about the physical state of the system. The value of a state predicate can change with time either due to the execution of an action, or the occurrence of an event.

TIMING CONSTRAINT: A timing constraint is an assertion about the absolute time of occurrence of certain events. The most general timing constraint that can be imposed on an event is a combination of a delay and a deadline. There can be three special types of timing constraints:

i) *Minimum*: The event should not occur before a certain minimum time.

ii) *Maximum*: The event should occur before a certain maximum time.

iii) *Bounded Interval*: The event should occur within a certain upper and lower time bounds.

The general construct that is used for expressing the timing constraint on an asynchronous action is as follows:

When < event > **execute** < action >
with separation = < time >, **delay** = < time >, **deadline** = < time >.

The delay and the deadline clauses are the restrictions on the start and the stop time of the action respectively. The separation clause is a constraint on the environment and limits the maximum rate at which it can apply stimuli to the embedded system. For representing the special classes of constraints, either the delay or the deadline clause of the above construct can be omitted, if necessary.

We now give an example of the event-action specification for a simple real-time distributed system – that of an abstract autonomous robot in an automatic manufacturing plant.

Example 1: An object (partially complete product) enters the work space of an autonomous robot (Fig. 1). The arrival times of any two consecutive incoming objects have a separation of at least 2000 msec. The sensor of the robot, on sensing an incoming object, signals the perception node. The perception node analyzes the signals received from the sensor, and identifies the type of object that has arrived, and communicates this information to the planner node. The planner computer decides the kind of work that has to be performed on the object. This information is passed on to the actuator node. The timing restriction on the system is that the system must initiate and complete the necessary action on the object within 500 msec and 1000 msec respectively from the object being sensed by the sensor.

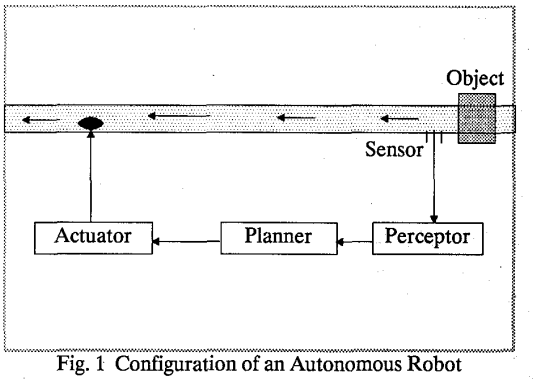

Fig. 1 Configuration of an Autonomous Robot

Event-Action Specification:

Perceptor Node

External Event:

ARRIVAL; :Arrival of an object.

Primitive Action:

IDENT; :Identify the the object.

Notifier Event:

NCTRLR :Notify Planner node of the arrival of an object.

Planner Node

Notification Event:

OBJARR; :Notification of the type of arrived object

Primitive Action:

DWORK; :Decide the type of work to be performed on the arriving object.

UPDATE; :Keep track of objects that have arrived and the type of work performed on them.

Notifier Event:

NOTACT :Notify the actuation node about the kind of work to be performed on the arriving object.

Actuator Node

Notification Event:

NTACT; :Notification of kind of work to be performed

Primitive Action:

ACTUATE; :Perform work on the object by driving the actuators.

Composite Action:

IDACT : IDENT; DWORK; (UPDATE || ACTUATE).

:Identify the type of object, decide the kind of work to be performed, record it as well as perform the required work.

Timing Constraint:

When OBJARR **do** IDACT

with separation = 2000ms, **delay** = 500ms, **deadline** = 1000ms.

3. Distributed Real-Time Logic

DRTL is a first-order logic. A system specification in DRTL consists of a set of formulas that are the safety assertions for the system. The DRTL formulas are composed of constants, functions, predicates, universal and existential quantifiers, and first-order logic connectives as in [1]. In this model, an action is characterized by a start action event (denoted by $\uparrow A$) and a stop action event ($\downarrow A$). The ith occurrence of an event e is denoted by (e,i). An occurrence function denoted by $@_i$ is associated with each node k. The occurrence function associates a unique time to each event occurrence. Thus, it can be viewed as a mapping $@_k:(E,N)\rightarrow N$, where E is the set of events associated with a node k, and N is the set of natural numbers. Each node in the system has a unique occurrence function to take care of the various unsynchronized clocks in the system.

A system specification given in the event-action model can be mechanically transformed into a set of DRTL formulas. For example, let us consider the general event-action construct for expressing an asynchronous timing constraint that is given by:

When e **execute** A

with separation = s, **delay** = d_1, **deadline** = d_2.

where A is the action and e is the corresponding triggering event associated with a node k. This construct can be transformed into the following DRTL formulas:

$\forall i\ \exists j(@_k(e,i)+d_1)\leq @_k(\uparrow A,j))\wedge(@_k(\downarrow A,j)\leq(@_k(e,i)+d_2))$

$\forall i\ (@_k(e,i)\ +\ s)\leq\ @_k(e,i+1))$

The transformation of other event-action constructs into DRTL formulas can be done in a way similar to that given in [1], with care being taken to apply the appropriate occurrence function to an event.

4. Inference Mechanism

In a distributed system, there are two fundamental uncertainties which hinder any rigorous formal analysis of the system's timing properties. One is associated with the message transmission delays, and the other with the presence of unsynchronized clocks at the various nodes of a system.

For simplicity in analyzing the effect of the message transmission delays, let us assume a reliable medium, i.. every transmitted message ultimately reaches its destination after a finite delay. This delay can be assumed to be bounded between a certain maximum and a minimum value, without assuming anything about how a transmission medium is physically implemented, e.g., shared bus, point-to-point line, etc. Let $d_{min}^{i,j}$ and $d_{max}^{i,j}$ be respectively the minimum delay and the maximum delay associated with a message transmission from a node i to another node j. Let us define **MIN** and **MAX** as functions finding respectively the minimum and the maximum of a set of values. Let d_{min} = **MIN**$\{d_{min}^{i,j}\}$ and d_{max} = **MAX**$\{d_{max}^{i,j}\}$. Then, the period over which a message is expected to reach its destination is spread over an interval (d_{max} - d_{min}), let us call this interval as the *message transmission skew.*

Many papers have appeared on the clock synchronization problem [5,12,13,14], containing different algorithms and analysis of the worst case accuracy of the clocks under stated conditions. It has been shown that the synchronization accuracy depends on the network delay, the quality of crystal time base, the frequency of resynchronization, the properties of the synchronization algorithm and the fault assumption. Hence, in general, the clock skew between different pairs of clocks in a distributed system may vary widely. However, for analyzing the effect of unsynchronized clocks on the deadline restrictions associated with the execution of an action, we will assume a bounded drift (clock skew) between any two clocks in the neighboring nodes of the system. Let $y^{i,j}$ represent the skew between the clocks of any two connected nodes i and j, and let y_{max} = **MAX**$\{|y^{i,j}|\}$.

Notification/notifier event axiom:

$\forall k(@_i(Y,k)<(@_j((X)+y^{i,j}),k)))$, if the notification event X occurs in a node j, and Y is the corresponding notifier event in the node i.

Definition 1: The specification of a hard real-time system is called **feasible** for a given set of clock skew and message transmission skew values, if it is possible to design the system without violating any of the constraints of the specification, assuming the presence of the clock skews and message transmission skews among the various nodes.

Lemma 1: The maximum uncertainty U in the execution of an action a of a distributed system is given by: U = $k*(d_{max} - d_{min} + y_{max})$, where k is the maximum number of message transmissions necessary for execution of an action **a**.

Proof: Each time there is a message transmission from one node to another, there is an introduction of a maximum uncertainty of (d_{max} - d_{min} + y_{max}) in the execution of the corresponding action. Since, there are at most k message transmissions required for execution of the action **a**, the

maximum uncertainty U in execution of this action is $k*(d_{max} - d_{min} + y_{max})$. ●

Theorem: A hard real-time system is feasible only if $d > (U + t_a)$, where d is the shortest deadline requirement on the execution of any action **a**, and t_a is the maximum execution time of the action in the absence of clock skew and message transmission skew.

Proof: The proof follows from definition 1 and lemma 1. ●

The various $d_{min}^{i,j}$, $d_{max}^{i,j}$, and $y^{i,j}$ values are parameters of the network and of the hardware of a design implementation. Thus, in a practical application, we must lay down the various $d_{min}^{i,j}$, $d_{max}^{i,j}$, and $y^{i,j}$ values that the system has to satisfy. In a simple case, the various $d_{min}^{i,j}$, $d_{max}^{i,j}$, and $y^{i,j}$ values may be the same, i.e., equal to d_{min}, d_{max}, and y_{max} respectively. However, they may vary widely if the distances between different nodes differ considerably – since, the transmission delay is, in general, a function of the distance between any two nodes.

Given the various $d_{max}^{i,j}$, $d_{min}^{i,j}$, and $y^{i,j}$ values, the DRTL formulas can be analyzed in a way similar to [1], for detecting any possible timing inconsistencies. However, whenever a comparison of expressions involving different occurrence functions @i and @j is involved, we can add an appropriate $x\ (0 \le x \le y^{i,j})$ to either the left hand or the right hand side expression, whichever helps to prove a timing inconsistency [1], while applying the Bledsoe and Hines' procedure [8]. Similarly, whenever two expressions involving occurrence of a notification event in node i and the occurrence of the corresponding notifier event in node j need to be compared, we can add a suitable $x\ (0 \le x \le (d_{max}^{i,j} - d_{min}^{i,j} + y^{i,j}))$ to either the left hand or the right hand side of the expression, whichever helps to prove a timing inconsistency. It can be observed that this results in worst case analysis of the specification, which is an important criterion for validation of hard real-time system designs.

5. Conclusion and Discussions

In this paper, we have proposed a formalism for analysis of the timing properties of distributed real-time systems. This formalism is based on the Real-Time Logic of Jahanian and Mok. This is a three level approach to formal specification and verification of timing properties of a system. In the first level of specification, the system is described in an event-action model. In the second level, this specification is mechanically transformed into a set of DRTL formulas for carrying out the safety analysis in the next level.

A node can be added modularly to a distributed system, by describing its event-action model along with its interaction with the other nodes in the system. The interactions among various nodes in the system are described by means of a set of notifier and notification events. DRTL formulas can be analyzed in a way similar to the analysis RTL formulas by imposing a few restrictions on the system. The proposed formalism does not consider functional specification and verification. Currently, we are investigating such an integrated framework for the verification of the correctness of distributed real-time systems.

6. References

[1] **F. Jahanian and A. Mok,** *"Safety analysis of timing properties of real-time systems,"* IEEE Transactions on Software Engineering, Vol. SE-12, pp. 890-904, September 1986.

[2] **R. Koymans,** *"Specifying message passing and real-time systems with real-time temporal logic,"* Technical Report 86/01, Eindhoven University of Technology, The Netherlands, 1987. [3] **R. L. Schwartz, P. M. Melliar_smith, and F.H Vogt,** *"An interval logic for higher order temporal reasoning,"* In Proceedings of Second ACM Symposium on Principles of Distributed Computing, 1983, pp. 173-185.

[4] **G. MacEwen and D. Skilicorn,** *"Using higher order logic for modular specification of real-time distributed systems,"* In Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems: Lecture Notes in Computer Science, M. Joseph Ed., Vol. 331, Springer-Verlag, 1988, pp. 36-66.

[5] **L. Lamport,** *"Time, clocks, and the ordering of events in a distributed system,"* Communications of the ACM, Vol. 21, No. 7, pp. 558-565, July 1978.

[6] **R. Koymans, J. Vytopil, and W.P. DeRoever,** *"Real-time programming and asynchronous message passing,"* Technical Report, RUU-CS-83-9, Universiteit Utrecht, The Netherlands, 1984.

[7] **M. Joseph and A. Goswami,** *"Formal description of real-time systems: a review,"* Technical Report RR129, Department of Computer Science, University of Warwick, U.K., August 1988.

[8] **W.W. Bledsoe and L.M. Hines,** *"Variable elimination and chaining in a resolution based prover of inequalities,"* In 5th Conference on Automated Deduction, Lecture Notes in Computer Science, W. Bibel and R. Kowalski Eds., New-York, Springer-Verlag, 1980, pp. 70-87.

[9] **P. Capsi, D. Pilaud, N. Halbwachs, and J. A. Plaice,** *"LUSTRE: a declarative language for programming synchronous systems,"* In Proceedings of the ACM Symposium on Principles of Programming Languages, January 1987, pp. 178-188.

[10] **G. Berry, S. Moisan, and J. Rigault,** *"ESTEREL: Towards a synchronous and semantically sound high level language for real- time applications,"* In IEEE Real-Time Systems Symposium, December 1983, pp. 30-37.

[11] **P. Le Guernic and A. Benveniste,** *"Real-time, synchronous, data-flow programming: the language SIGNAL and its mathematical semantics,"* Technical Report 620, INRIA, Rennes, France, 1986.

[12] **J. Lundelius and N. Lynch,** *"A new fault tolerant algorithm for clock synchronization,"* In Proc. of the Third ACM SIGACT-SIGOPS Symposium on the Principles of Distributed Computing, Vancouver, Canada, August 1984, pp. 75-88.

[13] **L. Lamport and P.M. Melliar Smith**, "*Synchronizing clocks in the presence of faults*," Journal of the ACM, Vol. 32, No.1, January 1985, pp. 52-78.

[14] **S.R. Mahaney and F.B. Schneider**, "*Inexact agreement: Accuracy, precision and graceful degradation*," In Proceedings of the Fourth ACM SIGACT-SIGOPS Symposium on the Principles of Distributed Computing, Minaki, Ontario, Canada, August 1985, pp. 237-249.