A Generic Temporal Consistency Model for Distributed Control Systems

Pedro Martins
pmartins@di.fc.ul.pt
Univ. of Lisboa*

António Casimiro
casim@di.fc.ul.pt
Univ. of Lisboa

Paulo Veríssimo
pjv@di.fc.ul.pt
Univ. of Lisboa

Abstract

The construction of applications that interact with their environment through sensors and actuators, is inherently a real-time problem if we consider the temporal constraints imposed by the evolution of the environment. The traditional vision is that actuation must be performed sufficiently fast to be consistent with the observation.

We believe that a more generic approach to the construction of distributed control systems is one that uses temporal consistency as the fundamental correctness criteria. In this paper we discuss preliminary ideas that support our conviction and we describe a temporal consistency model that is adequate for the construction of a wider range of distributed control systems.

1 Introduction and Related Work

The construction of applications that interact with their environment through the reading of sensor data from input sensors and by writing commands on output actuators in response to those readings, is inherently a real-time problem if we consider the temporal constraints imposed be the evolution of the environment. For instance, a control application designed to keep the temperature of an oven within a predefined distance from a set-point, must react within a bounded amount of time when the temperature reading sensors indicate a certain deviation from that set-point, by sending the appropriate actuation commands to the heating device. Failure in meeting the actuation deadline may lead to a violation of important safety properties.

This example also serves to highlight a typical approach taken in control systems, which is based on the definition of a sufficiently fast response time so that actuation is performed early enough to be consistent with the observation. Such a design is in fact based on the definition of a temporal validity or *temporal accuracy interval*, as defined in [1]. A similar approach is also taken in the context of real-time databases, where this interval has been called *absolute validity interval* [2]. The bottom line of these approaches is the use of temporal validity as the main correctness criteria. Correctness depends on the definition of a time interval, counted from an observation event, during which it is known that the information conveyed by the event is somehow consistent with the observed reality.

We believe that a more generic approach to the construction of distributed control systems is one that uses *temporal consistency* as the fundamental correctness criteria.

In this work we discuss this possibility, namely by studying some cases in which an application (or some state within the application) can be temporally consistent with the environment, even after the deadline corresponding to some temporal validity interval. Moreover, and in contrast with some works in the area of database systems [6, 3], we propose a model of temporal consistency where we take into account the impact of the underlying (communication and computation) infrastructure on the interaction delays.

This paper is organized as follows. In the next section we introduce the system model over which our temporal consistency model is defined. Then, in Section 3 we motivate the relevance and we describe the temporal consistency model. Finally, Section 4 presents some conclusions and future work.

2 System Model

The system model we consider is suited to distributed real-time systems and relies on the separation

*Faculdade de Ciências da Universidade de Lisboa. Bloco C6, Campo Grande, 1749-016 Lisboa, Portugal. Navigators Home Page: http://www.navigators.di.fc.ul.pt. This work was partially supported by the FCT, through the Large-Scale Informatic Systems Laboratory (LaSIGE).

between input-output, communication and computation. This model was formerly presented in [5] and here we summarize the main ideas.

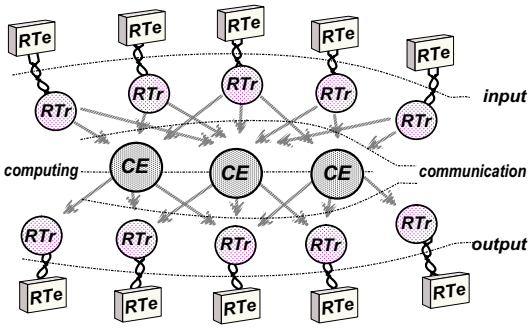

Figure 1. A generic real-time system model.

The model is depicted in Figure 1. A *real-time entity* (RTe) is an element of the environment, such as a fluid valve or the temperature of an oven, with a behavior which may be time-dependent and a state the system is supposed to acquire or modify. Computing is performed by *computational elements* (CE). Such elements are computational entities which process observation of RT entities (RTe), modify the internal state of the system and eventually trigger actuations on other RTe's, to modify the sate of the environment. The *representative* (RTr) of a real-time entity is an element of the computational system through which the latter observes (sensor) or acts (actuator) on the state of the real-time entity in the environment. *Input/Output* is performed between the RTe's and their representatives (RTr). Communication ensures timely flow between the computing elements and the input output representatives.

The state of a real-time entity is not accurately reflected in its representative at all times during system evolution. Apart from possible hardware failures (e.g. sensor failures), which should be masked using appropriated fault tolerant mechanisms (e.g. replication), the discrepancy between the state of RTe's and their RTr's results from observation errors caused by errors steaming from the observation apparatus (i.e. sensor module) and also because of the intrinsic nature of RTe's, which are *time-value entities*. Note that a time-value entity is essentially an entity such that there are actions on it whose time-domain and value-domain correctness are inter-dependent.

So, in order to be able to use the value of RTr's, it is necessary to handle the above-mentioned observation errors. But this is not enough. Secondly, we must ensure that the information available at RTr's is sufficiently fresh to be useful inside the system. For example, when we say that "the temperature is X at T", it is important that the interval between the time when it was measured and the time when it is used inside the system, is known and short enough to be useful, so that the temperature hasn't drifted too much in the meantime. This must be ensured by the infrastructure, and a practical way is to define a fixed parameter, known at design time, based on estimates of the variable's dynamics. This prefigures the *temporal validity interval*, typically used in control.

Enforcing time-related consistency requirements such as temporal validity is done in classical control theory under two generic approaches: *time-triggered* and *event-triggered*. In the time-trigger approach control systems execute cyclicly, fast enough in order to secure predefined set-points. On the other hand, time-triggered systems react to significant events that occur in the environment directly and immediately. Both approaches have advantages but also disadvantages, and both consider control in a closed-loop fashion.

We believe nevertheless these classical control system models are not suitable to address all the problems posed by the systems we would like to consider, namely due to their closed-loop nature. Therefore, we propose a temporal consistency model for distributed control systems, which is describe in the next section.

3 Temporal Consistency Model

As described in the previous section, the observation of time-value entities must be subjected to consistency constraints so that they can be handled correctly by the system. Time-value entities may be concerned with real-time entities or with internal entities. In the latter case, time-value entities are produced by internal system entities instead of being variables of the external environment. However, note that the state of internal entities may result from previous observation and processing of sensorial information (from real-time entities).

The state of a time-value entity E is mapped in a system state variable S_E. Since time-value entities cannot be accurately represented inside the system, we must establish correctness criteria for such representations. In general, this may be achieved by defining a maximum value error that may be accumulated by a state variable.

Assume bound $\mathcal{V}_s$ for the maximum acceptable error accumulated by a state variable S_E of a time-value entity E, $S_E(t_a)$ as computed with the sensorial information available at t_a, and $E(t_a)$ the state of E at t_a.

- Given t_a and a known $\mathcal{V}_s$, we say that a state variable S_E is **temporally consistent at** t_a if and only if $|S_E(t_a) - E(t_a)| \leq \mathcal{V}_s$, we call to $\mathcal{V}_s$ the *user tolerance*.

We say that a state variable S_E is a *real-time image* of E, iff S_E is temporally consistent during all system operation, according to a specified user tolerance $\mathcal{V}_s$.

Temporal consistency is more generic than temporal validity. All systems designed according to temporal validity requirements are also temporally consistent. However, it is possible to fulfill temporal consistency requirements even when temporal validity requirements expire or simply do not exist.

Intuitively, we can sustain the above statement by giving two simple examples. For instance, consider a temporal validity requirement Δ for an observation of a temperature X at a given instant T. This means that the observation of the temperature X is only valid during Δ units after T, and possibly inconsistent afterwards. Consider also that the observation is temporally consistent at T. Now, suppose that the temperature X does not change after the observation done a T. In this scenario, the temporal validity requirement expires at $T + \Delta$. However, since the observation was temporally consistent at T and the temperature did not change afterwards, it is obvious that the observation remains temporally consistent even after $T + \Delta$.

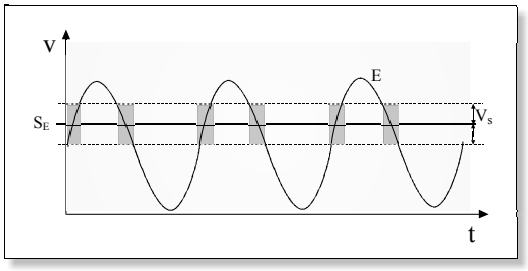

Figure 2. Temporal consistency example.

Now consider the scenario shown in Figure 2. The state of the time-value entity E is described by a sinusoidal wave with a certain frequency and amplitude. In this example, there are no temporal validity requirements. Observe however that periodically the state variable S_E is temporally consistent (dashed areas in the figure).

In our model, real-time images (RTi's) may assume a composite form, being possible to have RTi's of RTi's. For example, consider a real-time entity E (e.g. a temperature) and its RTi S_E, with an user tolerance $\mathcal{V}_s$; the RTi of E is stored inside a sensor device. Now consider an internal system entity with a RTi of the sensor data (S_{S_E}) and an user tolerance $\mathcal{V}_a$. This means the internal system entity has a RTi of the RTi kept by the sensor. Applying a transitive relation, we can say that S_{S_E} is a real-time image of E with an user tolerance $\mathcal{V}_s + \mathcal{V}_a$.

Observation of time-value entities: observations may take place both at the periphery (sensors) of the system and at its center (internal entities), and to be useful they must be consistent both in time and value domains.

For example, consider the temperature of an oven observed at an arbitrary instant T. Firstly, that we know that "the temperature is X at T", might seem to provide a precise indication. However, what Tportrays is the time at which the system observed the temperature. When observing the value of a continuous variable, it is relevant to define the error. For an observation $\langle r(E_i)(t_i), T_i\rangle$ of the value of an RTe E_i at t_i receiving timestamp T_i, the *observation error* in the *value domain* is given by $\nu_i = |E_i(T_i) - E_i(t_i)|$: we expect the value of E_i at T_i, but we get an approximation of the value measured approximately (t_i) at T_i. If we consider also the error caused by the observation apparatus (sensor module), then we get $\nu_i = |(E_i(T_i) - E_i(t_i)) + (E_i(t_i) - r(E_i)(t_i))| = |E_i(T_i) - r(E_i)(t_i)|$. Alternatively, when observing the time at which a given discrete value E_i occurs (e.g., opening of a door), we must define the observation error (jitter) in the *time domain*, $\zeta_i = |T_i - t_i|$: E_i assumed a given value at t_i, but the system logs it as having happened at T_i.

So, we must establish a bound for the errors, in order for our measurement to be useful:

- Given a known $\mathcal{V}_o$, we say that an observation is *consistent* in the *value* domain, iff $\nu_i \leq \mathcal{V}_o$
- Given a known $\mathcal{T}_o$, we say that an observation is *consistent* in the *time* domain, iff $\zeta_i \leq \mathcal{T}_o$

Our definition of temporal consistency enforces a bounded error ($\mathcal{V}_s$) in the value domain. However, since we are dealing with time-value entities, we must establish also constraints in the time-domain to secure the required (value) bound, described in [5] as the time-value duality. Therefore, we have to ensure consistent observations in the time domain ($|T_i - t_i| \leq \mathcal{T}_o$) in order to secure $|E_i(T_i) - r(E_i)(t_i)| \leq \mathcal{V}_s$.

Modelling the environment: since real-time entities are part of the external environment we need to describe their behavior so that temporal consistency requirements can be secured by the system. Real-time entities can be described in several ways. For example,

a RTe whose state follows a sinusoidal wave can be described by its frequency and amplitude. A fundamental specification of a time-value entity is the variation of its value over the time. We denote the *drift of a time-value entity* E in a time interval T as $E_\delta(T)$.

Handling discrete time-value entities: we have been addressing temporal consistency criteria for continuous time-value entities. However, computer systems are discrete, acquiring the state of continuous variables at discrete time instants. In that sense, continuous time-value entities can be viewed as discrete entities from a system point of view. For instance, consider a perfect observation (one that acquires the exact value of a time-value entity E) done at an arbitrary instant t_i. Since the computer system is discrete, the next observation will take place at an instant no sooner than $t_i + \Delta_t$. This means that what the system can observe is only the variation of the entity E in the interval $[t_i, t_i + \Delta_t]$. Therefore, even for perfect observations, it is not possible to ensure temporal consistency requirements for user tolerances lower than $\Delta_v = E_\delta(\Delta_t)$.

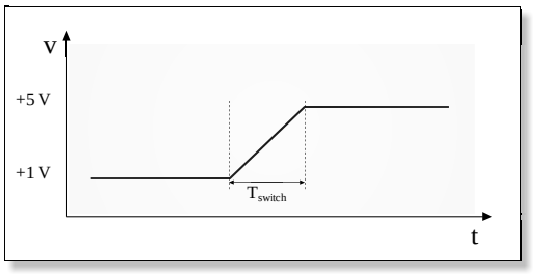

Figure 3. A discrete time-value entity.

On the other hand, we can argue that discrete time-value entities do not exist in the environment. For instance consider a real-time entity, an electrical circuit voltage which may be in a discrete state of $+1V$ or $+5V$ at a given instant. Is this really a discrete time-value entity? The answer must be negative, since it would be necessary an infinite power to switch instantaneously from a voltage level of $+1V$ to $+5V$, which is physically impossible. So this time-value entity has in fact a behavior as depicted in figure 3. The voltage does not change immediately from $+1V$ to $+5V$, rather there is a switching time interval T_{switch} during which the voltage changes continuously from one state towards the other.

However, it is not possible to handle all time-value entities as continuous variables, being sometimes necessary to discretize some of them. For example considerer again the example shown in figure 3. For very fast switching periods (e.g. $< \Delta_t$) the system might not perceive the intermediate states, being thus necessary to discretize the state. Handling "discrete" time-value entities is not a simple problem which we intend to address in the context of our temporal consistency model as future work.

Semantic issues: when observing time-value entities through real-time images there are some semantic issues that emerge, and programmers should be aware of them when conceiving control systems under our temporal consistency model.

Consider a real-time image S_E of a time-value entity E with an user tolerance of $\mathcal{V}_s$, $A = S_E(t_i)$ as the value of the state variable S_E at an instant t_i, and $B = S_E(t_f)$ as the value at an instant $t_f > t_i$, for any considered t_i and t_f. Based on the values read from the real-time image, it is only possible to conclude the following aspects regarding the state of the time-value entity E:

- the variable increased between t_i and t_f iff $B > A + \mathcal{V}_s$;
- the variable decreased between t_i and t_f iff $A > B + \mathcal{V}_s$;
- nothing can be concluded concerning the evolution of the time-value entity if $|A - B| <= \mathcal{V}_s$.

The above discussion leads to an interesting conclusion. For any user tolerance $\mathcal{V}_s > 0$ on a temporal consistent criteria for a time-value entity E, it is not possible to conclude whether or not the state of E is the same at two arbitrary instants. Since $\mathcal{V}_s = 0$ is theoretically impossible, the above conclusion holds for any considered system and time-value entities.

When values from two or more real-time entities are used at a given instant t_a, there are also some semantic issues that must be considered. Consider a real-time image S_E of a time-value entity E with an user tolerance of $\mathcal{V}_s$, and a real-time image S_U of a time-value entity U with an user tolerance of $\mathcal{V}_u$. Consider $A = S_E(t_a)$ as the value of the state variable S_E at the instant t_a, and $B = S_U(t_a)$ the value of the state variable S_U also at the instant t_a. Taking into account the values read from the two real-time images, it is only possible to conclude the following:

- $A > B$ iff $A > B + \mathcal{V}_s + \mathcal{V}_u$;
- $A < B$ iff $A < B - \mathcal{V}_s - \mathcal{V}_u$;
- nothing can be concluded about the values if $|A - B| <= \mathcal{V}_s + \mathcal{V}_u$.

4 Conclusions and Future Work

Temporal consistency is not a new concept. However, in our opinion, existing works in this area are not generic enough, since they make simplifying assumptions about the interactions taking place, focusing essentially on the data itself. On the contrary, system timing properties, often neglected in other works on temporal consistency, play an essential role in our work. Another important aspect of our work is that we look at temporal consistency as the main correctness criteria, whereas traditional approaches try to secure temporal validity instead. We argue that temporal validity is just one practical mean to achieve temporal consistency, but other means may be envisaged where temporal validity is not applicable. With our work we aim at exploring this idea to provide a better understanding and new tools for the construction of distributed control systems.

In particular, we intend to address some issues regarding the architectural support for developing applications in accordance to our temporal consistency model. The Generic Events ARchitecture (GEAR) [4] provides some initial background for this purpose, which we will extend. One of the objectives is to develop a runtime support that reacts and adapts to the underlying infrastructure timing conditions in order to secure the temporal consistency requirements of user applications. In addition, the study of applications that can dependably change their behavior to secure their temporal consistency requirements will be also subject of our future work.

References

[1] H. Kopetz. *Real-Time Systems*. Kluwer Academic Publishers, 1997.

[2] K. Ramamritham. The origin of tcs. In *In Proceedings of the First ACM International Workshop on Active and Real-Time Database Systems*, pages 50–62, Skovde, Sweden, 1995.

[3] R. Srinivasan, C. Liang, and K. Ramamritham. Maintaining temporal coherency of virtual data warehouses. In *RTSS*, pages 60–, 1998.

[4] P. Veríssimo and A. Casimiro. Event-driven support of real-time sentient objects. In *Proceedings of the Eighth IEEE International Workshop on Object-oriented Real-time Dependable Systems (WORDS 2003)*, Jan. 2003.

[5] P. Verssimo and L. Rodrigues. *Distributed Systems for System Architects*. Kluwer Academic Publishers, 2001.

[6] M. Xiong, J. A. Stankovic, K. Ramamritham, D. F. Towsley, and R. M. Sivasankaran. Maintaining temporal consistency: Issues and algorithms. In *RTDB*, pages 1–6, 1996.