Where Do Time Constraints Come From? Where Do They Go?

Krithi Ramamritham
University of Massachusetts

While a lot of work has been done in real-time systems, in real-time database systems, and more recently, in real-time active databases on the topic of time constrained processing of tasks and transactions, very little work exists that deals with the origin of the time constraints associated with the data, the events, and the actions. In this paper we identify the sources and semantics of time constraints and show that it is important to minimize the number of “mandated” timing requirements and also weaken the implications of timing constraint violations. The Event-Condition-Action rules of active real-time databases provide a useful framework to specify the timing properties of interest as well as the actions to be taken when the properties are violated. That is, an active real-time database can be made to store the data pertaining to the controlled system as well as the meta-data about the controlling system.

What separates real-time systems from non real-time systems is the presence of data that becomes invalid with the passage of time, the presence of events that must occur in a timely fashion, and the presence of actions whose timely completion is as important as the results produced. Many of the timeliness requirements, however, are artifacts of the way a system is designed and the (usually ad-hoc) manner in which the time constraints are assigned. Of course, some time constraints are imposed by the external (physical) environment and these must be ensured by the system. However, quite often, since the origins of the time constraints are not known to a real-time system, it attempts to satisfy them all, leading to an overconstrained or overdesigned system.

Given specific time constraints, a lot of work has been done on the topic of time constrained processing of tasks and transactions (Ramamritham, 1993). But very little work exists that deals with the origins of the time constraints associated with the data, the events, and the actions. Knowing their origins it is possible to determine the semantics of the time constraints and also the best ways to satisfy timing requirements. This paper examines these issues and shows that the timing properties of interest and the actions to be taken when the properties are violated can be specified using the Event-Condition-Action (ECA) rules of active databases. Because of this, a real-time active database system (RTADB) can serve as a repository of not only the data about the environment of interest but also of the meta-data that can help in the adaptive handling of violation of time constraint—of the data, events and the actions.

Throughout this paper we will use the term action to refer to computations with time constraints. Actions could be complex, containing subactions. Actions encompass real-time tasks and real-time transactions.

Motivating Examples

In this section we introduce two examples of real-time applications that benefit from real-time active database technology. Aspects of these examples will be used throughout the paper to illustrate our ideas. (Further detailed examples can be found in Purimetla et al (1995)).

Consider recognizing and directing objects moving along a set of conveyer belts on a factory floor. An object’s features are captured by a camera to determine its type and to recognize whether it has any abnormalities. Depending on the observed features, the object is directed to the appropriate workcell. In addition, the system updates its database with information about the object.

The following aspects of this example are noteworthy. First of all, features of an object must be collected while the object is still in front of the camera. Then the object must be recognized by matching the features against models for different objects stored in a database. This matching has to be completed in time so that the command to direct the object to the appropriate destination can be given before the object reaches the point where it must be directed onto a different conveyer belt that will carry it to its next workcell. The
database update must also be completed in time so that the system’s attention can move to the next object to be recognized. If, for any reason, a time-constrained action is not completed within the time limits, alternatives may be possible. In this example, if feature extraction is not completed in time, the object could be discarded for now to be brought back in front of the camera at a later point in time.

As another example, consider the following air traffic control scenario at an airport. The air traffic control system makes decisions concerning incoming aircrafts’ flight path, the order in which they should land, and separation between landings based on a multitude of parameters including current wind velocity, position of the aircrafts, their speed, fuel position, and altitude and the type of aircraft. If an air traffic controller cannot accommodate all the incoming aircrafts, he or she can ask the aircraft(s) to assume a holding pattern until the situation improves.

Applications such as these introduce the need for real-time active database systems. These applications involve gathering data from the environment, processing gathered information in the context of information acquired in the past, and providing timely response. Another aspect of these examples is that they involve processing of both temporal data, which loses its validity after a certain interval, as well as archival data. Finally, the applications involve triggering of specific actions in specific situations.

Real-time systems consist of a controlling system and a controlled system. For example, in an automated factory, the controlled system is the factory floor with its robots, assembling stations, and the assembled parts; the controlling system is the computer and human interfaces that manage and coordinate the activities on the factory floor. The controlled system can be viewed as the environment with which the computer interacts.

Factors That Determine the Time Constraints

In this section, we examine the factors that determine these time constraints. The externally-imposed temporal properties depend on many factors including:

- Characteristics of the physical systems being controlled: e.g., the speed of the aircraft (to calculate time at which it will touch ground); the speed of the conveyer belt (to calculate available time to decide the path of the object).
- Stability characteristics of a system as governed by its control laws: e.g., servo control loops of robot hands, fly-by-wire avionics.
- Quality of service requirements (service delays tolerable by the controlled system or humans): e.g., sampling rates for audio and video; responsiveness of air traffic controller to an aircraft pilot’s request for current weather conditions at destination.
- Human (re)action times, human sensory perception: e.g., time between warning (e.g. low fuel level) and action based on the warning (e.g. immediate start of landing following emergency procedures).

These factors determine the time constraints that are inherited by the controlling system from the external environment. Understanding these factors is important because some of these are more important to satisfy (e.g., physical environment related factors) than others (e.g., quality-of-service factors related to human interactions). Using this information, importance levels can be set for actions.

In some sense, all time constraints, be they externally imposed or resulting from design decisions (see below), are artifacts. For instance, in relation to the items listed above, length of a runway or speed of an aircraft are determined by cost and technology considerations; quality of service requirements, for instance, in telephone networks, are quite often decided by regulatory authorities; response times guaranteed by service providers are determined by cost and competitive-ness factors. Unfortunately, many of these decisions are not under the control of a computer system designer. Hence, the system designer’s decisions must be made within the boundaries laid out by others. This is what we referred to above as being externally-imposed. What is important to note is that the number of factors “given” to the designer must be kept to a minimum leaving more options to satisfy the requirements.

Subsequent decisions of the designer introduce additional constraints. The type of computing platform used (e.g., centralized vs. distributed), the type of software design methodology used (e.g., data-centric vs action-centric), the (pre-existing) subsystems used in composing the system, the nature of the actions (e.g., monolithic action vs. graph-structured action), etc., further curtail the decisions.

At some level in the design of a real-time application, decisions concerning whether an action is periodic, sporadic, or aperiodic have to be made, the values for the periods, deadlines, and offsets within periods must be chosen, and importance or criticality values must be assigned.

Thus, the decisions made at one level affect those at the other level(s). What has to be recognized is that while no decision at any level is likely to be sacrosanct, i.e., beyond modification, cost and time considerations will prevail in any such overhaul of prior decisions. Decisions at one level may percolate to many subsystems below and a previous decision may not be changeable just to accommodate the needs of a single subsystem.

While it will be beneficial to determine all related time constraints in an optimal fashion, for non-trivial systems the problem is likely to be intractable. This is the reason a divide-and-conquer approach is adopted wherein a system is designed from subsystems each of which must satisfy a set of given time constraints. Such time constraints reflect the specific design strategy and the subsystems chosen as much as
Related Content

A Machine Learning Approach to Data Cleaning in Databases and Data Warehouses
www.igi-global.com/chapter/machine-learning-approach-data-cleaning/20376?camid=4v1a

Fuzzy Database Modeling: An Overview and New Definitions
www.igi-global.com/chapter/fuzzy-database-modeling/44379?camid=4v1a

Automated Insertion of Exception Handling for Key and Referential Constraints
www.igi-global.com/article/automated-insertion-of-exception-handling-for-key-and-referential-constraints/84066?camid=4v1a

Framework for a Geographic Districting DSS using an Intelligent Object-oriented Model
www.igi-global.com/article/framework-geographic-districting-dss-using/51180?camid=4v1a