AN INTEGRATED REAL-TIME DATA MANAGEMENT ARCHITECTURE FOR INDUSTRIAL CONTROL SYSTEMS

Jiandong Huang and John A. Stankovic

CMPSCI Technical Report 93-08
January 1993

Submitted to the IEEE Workshop on Parallel and Distributed Real-Time Systems 1993

An Integrated Real-Time Data Management Architecture for Industrial Control Systems*

Jiandong Huang
Honeywell, Inc.
Sensor & System Development Center
3660 Technology Drive
Minneapolis, MN 55418

John A. Stankovic
University of Massachusetts
Department of Computer Science
Amherst, MA 01003

Abstract

Next generation control systems will be more complex, dynamic, and yet flexible, in terms of functionality, distribution, and timeliness. In such systems, data management becomes more critical than ever before. It must not only provide the system with sophisticated data manipulation services, but also be able to complete the required services in a timely manner. In this paper, we present an integrated real-time data management architecture for managing plant-wide data under timing constraints. Based on the analysis of data properties, we take a functional partitioning approach to the design of the data management system. In particular, we define three types of database subsystems - hard real-time database, soft real-time database, and conventional database - with each providing support for management of different types of data. We specify the functionalities of each subsystem and further discuss the integration of these subsystems to form a plant-wide data management architecture. As far as we know, the proposed architecture is the first of its kind for industrial control systems. In this paper, we also discuss the design aspects and related research issues involved in developing such an integrated real-time data management system.

---

* This work was supported in part by the National Science Foundation under Grant IRI-9114197.

1 Introduction

Data processing and management has always been an important part in industrial control systems. For example, a control system needs to collect sensor data for on-line execution of control algorithms or provide history data for off-line analysis. The system also needs to manage a large set of parameters for describing and supporting system configuration, control process settings, etc. Today, as control applications become more sophisticated in terms of functionality, distribution, and timeliness, sophisticated data management must be supported. This is exemplified by the following requirements:

- *Time-constrained data management*—Timing constraints are often imposed on data management. These constraints may originate from either *real-time data* or *real-time processes*. Real-time data refer to data that need to be processed within a certain period of time. For example, input data from sensors must be stored in time, otherwise they will be lost. Real-time processes, on the other hand, are the ones that need to be completed by a certain time. The process for an alarm event is such an example, where the alarm status is required to be displayed on a control console, say in 2 seconds. Timing constraints are further categorized in two classes: *hard* and *soft*. The former requires guarantee on meeting the timing requirement, while the latter requires the best efforts in meeting the timing constraints. Examples for hard and soft timing constraints are execution of periodic control algorithms for discrete process control and display of certain trend data, respectively. Thus, a control system should be capable of providing data management services under both hard and soft timing constraints.

- *Distributed data management*—In a distributed control system, data are processed, stored, and used at various subsystems, such as process controllers, control stations, and plant management systems. In addition, data are accessed in a distributed fashion, from one subsystem to another across networks. For example, a control station which runs advanced control algorithms may access process data stored in a controller. Another example is the direct access of a plant information management system, such as a relational database, from plant floor devices for activities like on-line production scheduling. Clearly, the control system should provide support for distributed data management services.

The distribution and timing requirements have been challenging system designers in developing a unified, predictable data management system for next generation control systems.

Recently, substantial effort has been made in researching real-time database systems for time-constrained data management [JRTS92][Rama92]. Researchers have proposed and evaluated many real-time oriented data management schemes, especially for real-time transaction processing. The results are promising, indicating that real-time databases significantly improve performance, in terms of meeting data processing timing constraints, as compared to traditional databases. However, the research work is insufficient, by in large, to meet the application requirements discussed above. First, most of the work has focused on centralized real-time data management environments. Second, only soft real-time database systems were considered. Third, the previous research almost had no application contexts, making the results not directly usable in practice.

The objective of our work is to explore real-time data management in the context of industrial control systems. In particular, we consider a distributed environment where data reside in different components with different properties and different processing requirements. The goals of this paper are two-fold:

- We develop an architectural framework for supporting a unified, and yet flexible, data management system that meets different data access timing and consistency constraints.

- We study the design aspects and identify related research issues for developing such a data management system.

The paper is organized as follows. We first establish a control system model in Section 2. Then, in Section 3, we analyze and characterize various types of data in a control system with respect to their permanency, timing constraints, and consistency requirements. Based on the control system model and data characterization, we present a distributed, hierarchical real-time data management system in Section 4. In Section 5, the design and research issues regarding the proposed system architecture are discussed. Finally, we present some concluding remarks in Section 6.

2 Control System Model

In this section we introduce a control system model to establish a framework for this study. Figure 1 shows a system model based on the Honeywell TDC 3000 control system. The system consists of four levels:

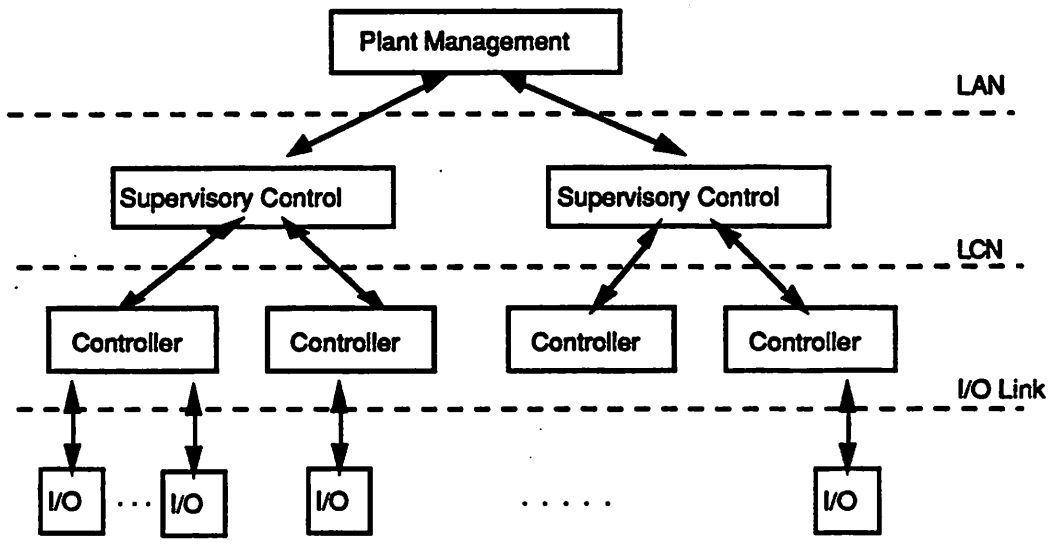

Figure 1. A Control System Model

- *Physical I/O level*—Sensors and actuators are the main processing components at this level. They provide real-time, raw data for monitoring and controlling.

- *Controller level*—Distributed controllers provide real-time, on-line direction for each automated production function. Controllers issue commands based on sensor input and fixed, configured or programmed logic. They also aggregate input data for decision making in the upper levels.

- *Supervisory control level*—Supervisory control centers communicate downward with distributed controllers to help department and work cell supervisors better manage their operation. These supervisory centers also communicate upward, linking the process or the factory floor to the plant-wide information system.

- *Plant management level*—Plant management systems are linked to the process and/or factory control system in real-time. Information to plant managers is always current, for sound decision making.

There are three types of communication networks which connect the multi-level system. The local area network (LAN) carries out communication between different sites of plant management and between the plant management and its underlying supervisory control systems. Similarly, the local control network (LCN) links the supervisory control and its underlying control systems. Further, the I/O link network (I/O Link) connects various controllers and physical I/O devices.

The model represents typical distributed control systems, and thus will be used throughout this study.

3 Data Characterization

As can be seen above, data reside at all levels in a distributed control system. These data need to be analyzed before appropriate data management schemes are developed. In this section we analyze some representative data through examples and characterize them from the view point of real-time data management.

We analyze data using the properties: permanency, timing, and consistency. *Permanency* refers to the duration of data validity. The *timing* property specifies the timing constraints imposed on data management. The *consistency* property specifies the requirement on data integrity. Figure 2 illustrates the residency of some data typically found in a control environment. Let us examine these data based on the three properties.

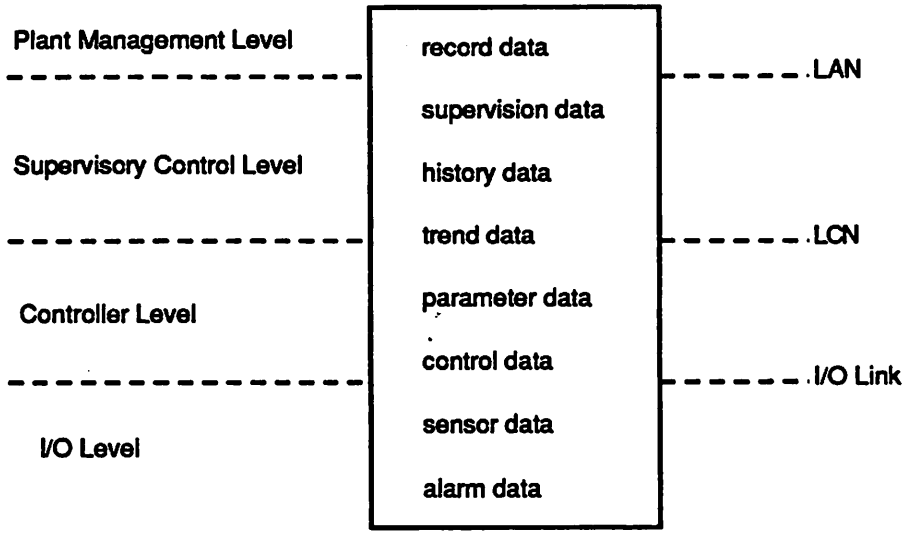

Figure 2. Data Residency

- *Record data*—can be found in plant management level, where product information is stored, as records, in a conventional database. Usually, these data are not frequently updated. In other words, they have the nature of "life-time" permanency, and thus are not real-time data. Also, the processes that access the data are not real-time oriented. Therefore, there is no need to manage these data under timing constraints. However, since the data are commonly shared among multiple users, maintaining strict consistency is required.

- *Supervision data*—reside at the supervisory control level. The data include load images for controllers, control algorithm data, system configuration files, and source codes. The data permanency last as long as the characteristics of the control algorithm or system settings. However, processes that access the data may have soft timing constraints. For example, an image loading process needs to be "quickly" done so that a controller can be restored in time. Due to data sharing, the data consistency is required.

- *History data*—are the collection of control process information over a specified time period. They are typically used for supervisory control or for plant management. The data permanency is "life-time" long, if the data need to be kept for permanent records. The collection operation may be invoked "statically" with a fixed time interval or "dynamically" at any time instant. Since the collection operation deals with the past data, the timing constraints are likely to be *soft*. For the same reason, the data consistency constraints may be relaxed.

- *Trend data*—are similar to history data, except that the data permanency is relatively short, since the data are collected for temporary use, such as trend display.

- *Parameter data*—are the values and set points used by control functions embedded in control units. The permanency of such data can be short due to on-line process control activities. These data may be frequently polled by a control console or other control units. The access operation may become critical to discrete control applications or in case of emergency. Thus, the timing constraints imposed on the control data can be *hard*. Data consistency may or may not be required, depending on the degree of data sharing among the controllers and supervisory control.

- *Control data*—are directly related to control operations and are subject to frequent access and change. Examples of such data are On/Off and duration counters of certain control devices. Access to these data can be time-critical. On the other hand, data consistency is not strictly required.

- *Sensor/actuator data*—are directly received from or sent to physical control devices, thus having "short" permanency. These data are usually periodic and critical in time, especially for discrete control applications. In addition, since sensor/actuator data are processed independently by dedicated tasks, there is no data sharing. Therefore, there is no consistency requirements on the sensor/actuator data.

- *Alarm data*—describes the status of a controlled object or the status of a controller itself. As soon as alarms are detected, the corresponding data must be reported and "journaled" (recorded) immediately. Of course, processing of alarm data needs to be guaranteed in time. Consistency is not required for this type of instant data.

To summarize, Table 1 lists the types of data and their corresponding properties found in today's control systems. Clearly, the characteristics of those data vary in terms of permanency, timing constraints and consistency requirements. Generally, data which reside at the top level of a hierarchical control system have the property of long permanency. Management of these data is less urgent, but requires a great care of data consistency. At the low level, on the other hand, data has short lifetime. Data processing has less consistency requirements, but are subject to stringent timing constraints.

Table 1 Data Characterization

Data Type	Permanency	Timing	Consistency
Record	long	none	absolute
Supervision	long	none/soft	absolute
History	long	none/soft	absolute/relax
Trend	short	soft	absolute/relax
Parameter	short	soft	absolute/relax
Control	short	soft/hard	relax
Sensor/Actuator	short/instant	hard	relax
Alarm	instant	hard	none

4 An Integrated System Architecture

Having established a control system model for industrial process control and examined various types of data, we now consider data management from the system architecture stand point in this section.

As described in the previous section, process data reside at different levels of a control system and they may have different characteristics. Particularly, they are different in the data management requirements on timeliness and consistency. We envisage that different types of data need different kinds of management systems. This can be supplied in various ways including: a separate subsystem for each type with clean interfaces between the subsystems, or a single unified system which supports a wide variety of properties. This paper focuses on the former approach. The need for different types of data management support is due to the following considerations:

- *Real-time scheduling*—Data access processes need to be scheduled based on their timing constraints. At one extreme, access to real-time data, such as sensor/actuator data, needs the support of real-time operating systems which shall be able to provide predictable access time. On the other extreme, access to non real-time data, such as inventory records, can be carried out under conventional operating systems which emphasizes scheduling fairness among concurrent users and balancing CPU and I/O utilizations. Thus, different scheduling algorithms are needed for processing different types of data.

- *Consistency management*—Different types of data are subject to different kinds of management schemes. For example, record data at the plant management level usually needs full support of transactions. However, at the controller level, not all of the transaction ACID properties[1] are necessary. For instance, isolation and durability may not be required for data with short permanency. Therefore, the consistency management can be better optimized, in terms of functionality and

[1] ACID stands for Atomicity, Consistency, Isolation, and Durability.

efficiency, for different types of data. In addition, with different management schemes, one can accordingly explore real-time scheduling mechanisms to increase predictability of data management services. For example, a relaxed consistency requirement (e.g., non-searializable read as opposed to serializable read and write [Bern87]) may lead to a non-blocking concurrency control protocol. The non-blocking feature is particularly important in achieving predictability of real-time scheduling.

- *Data model*—Data can be organized and presented to applications in various ways depending on application needs. An object-oriented data model may be suitable for describing control applications [Huang91a], while a relational model is desirable for record data. Hence, different models may co-exist in a plant-wide data management system. This is especially true when existing enterprise databases are integrated with the control system. On the other hand, it is a research issue whether a single data model (be it the relational, object-oriented or active database model) can sufficiently support all types of data requirements.

- *Data size and storage*—Different types of data may have different sizes and require different storages. For example, the amount of data managed at a control is relatively small, say a few hundred bytes. In addition, to speed up data access and provide better access predictability, a piece of main memory is usually required for data storage. The situation is different at the supervisory level, where data volume can be large, say millions of bytes, and a number of disks is needed for storage.

In order to apply different management schemes to different types of data and to accommodate the data residency, we take a functional partitioning approach [Huang91a]. That is, rather than managing all the data in one database system, we design different subsystems with each providing support for management of different types of data.

Figure 3 illustrates a multi-level data management architecture that we propose based on the functional partitioning approach. This system consists of three types of database subsystems: conventional database (DBMS), soft real-time database (SRT-DBMS), and hard real-time database (HRT-DBMS), residing at the plant management level, the supervisory level, and the controller level, respectively. The interface at each subsystem enables interconnection between subsystems. It provides access transparency in terms of system timing behavior as well as data residency and data model mapping. We will further discuss the related design and research issues in Section 5.3.

Figure 4 shows a system-wide data management organization in a multi-level hierarchy. At the top, a DBMS can be connected with multiple SRT-DBMS'. It may collect data from the SRT-DBMS' or send information to them. Similarly, at the middle level, a SRT-DBMS may be connected with multiple HRT-DBMS'. At the bottom, a HRT-DBMS may carry out data management for multiple controllers and upload/download various process data to/from SRT-DBMS. DBMS may also directly access HRT-DBMS through SRT-DBMS interface. Hence, the hierarchy is a distributed, heterogeneous database system. On the other hand, horizontal access between the same type of subsystems at each level is possible. Thus, horizontally, each level of the data management hierarchy is a distributed, homogeneous database system.

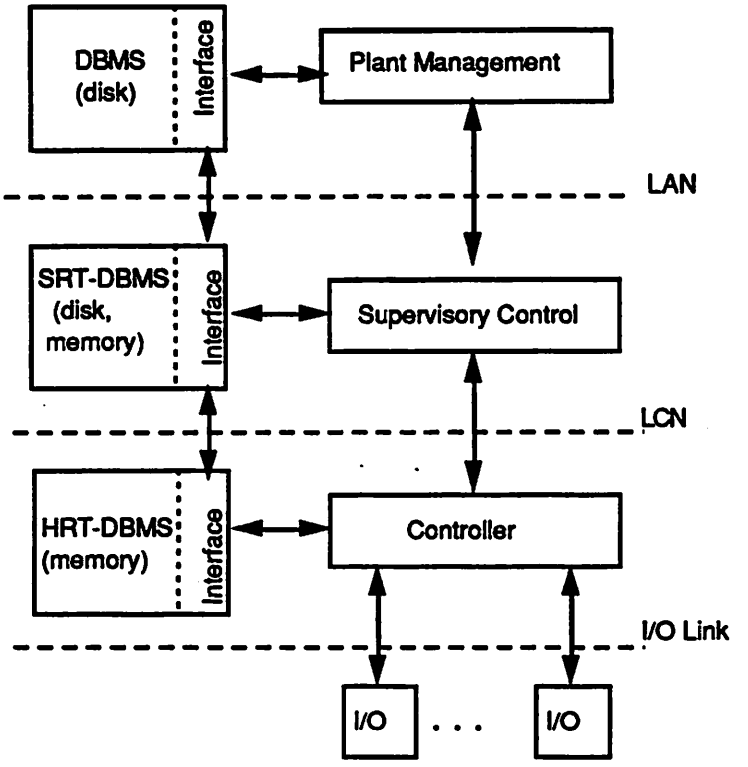

Figure 3. Real-Time Data Management Architecture

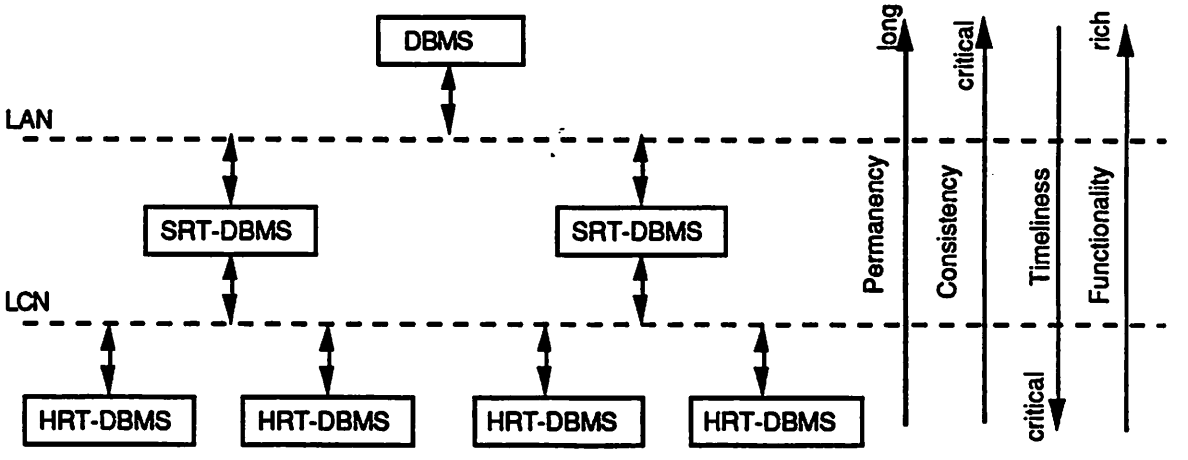

Figure 4. A Hierarchical System Organization

The three types of databases are different in their functionality and real-time processing capability.

- *HRT-DBMS*—is a database system that is capable of providing data management services under *hard* timing constraints. In particular, it supports structured data storage for parameter data, control data, trend data, and alarm data, and provides data access services for control process running at the local controller(s) or at the supervisory control station(s). The services are carried out by three types of transactions:

  - *Command transaction*—executes service requests issued from supervisory control. For example, a supervisory control process may ask the HRT-DBMS to query a group of controllers about their current states.

  - *Periodic transaction*—collects data from controllers in a specified period of time and insert them in the HRT-DBMS. The data is used for trend display at the supervisory control level, history data collected by SRT-DBMS, or advanced control algorithms running at the supervisory station.

  - *Event-triggered transaction*—executes service requests from controllers. An alarm event, for example, may enforce a controller to send its current status to the HRT-DBMS. The transaction must respond to the event "quickly" and intelligently. For instance, it may gather related data and prepare a report for the supervisory control.

  HRT-DBMS has the following characteristics:

  - It is hard real-time in the sense that transaction execution is scheduled based on the associated timing constraints and the execution time is predictable.

  - It is memory-resident in order to facilitate the real-time guarantee effort.

  - It supports loose consistency in the sense that the notion of serializability [Bern87] is relaxed (see discussion in Section 5).

- *SRT-DBMS*—is a database system that supports data management under soft timing constraints. The system provides structured data storage for supervision data, trend data, and history data, and access services for supervisory control and plant management. It may support three types of transactions:

  - *Command transaction*—executes service requests issued from supervisory control or plant management. For example, a supervisory control process may update system configuration parameters stored in the SRT-DBMS. Or, a plant management process may query the SRT-DBMS for some history data.

  - *Periodic transaction*—collects data from the underlying HRT-DBMS' in a specified period of time and, for instance, insert them in the SRT-DBMS for history display.

  - *Distributed transaction*—carries out distributed data management services. For example, a distributed transaction may collect data from multiple HRT-DBMS'. There are two types of distributed transactions: one is *coordinating transaction* which accepts the initial service request and distributes the request to other SRT-DBMS sites or the underlying HRT_DBMS'; the other is *sub-transaction* which carries out the request from a coordinating transaction at another SRT-

DBMS or DBMS site.

SRT-DBMS has the following characteristics:

- It makes "best-effort" to schedule real-time transactions, but does not provide absolute guarantee in terms of bounded execution time.
- It supports data consistency based on the notion of serializability, with relaxed ACID properties (e.g., Isolation may not be required.).
- It is mainly disk-resident. Memory storage is also needed to support data buffering, especially for staging hard real-time data from HRT-DBMS.

- *DBMS*—is a conventional (distributed) database system for record-type information management such as production scheduling, inventory control, material requirements planning, labor reporting. The system enforces data consistency based on the notion of serializability, but does not provide real-time-oriented data management services.

In summary, the data management system has a distributed, hierarchical architecture. The system consists of three types of databases which are adapted to different service needs in terms of permanency, timing constraints, consistency requirements, residency, and functionality. Overall, towards the top level of the hierarchy, the system provides support for stringent consistency requirements and richer data management functionality; towards the bottom, on the other hand, the system provides support for stringent timing requirements with relaxed consistency requirements and less functionality. The architecture offers three advantageous features:

- Modularity—Each subsystem is a self-contained functional unit with a well-defined interface. A plant-wide data management system can then be configured using these units. Depending on application needs, different types or number of subsystems may be used. The modularity enables flexibility, scalability, and cost-effective productization.
- Transparency—Even though data management is carried out in different types of subsystems, it functions as a uniform database. Applications at each level of the system hierarchy see a consistent view of the data management system in terms of timing, consistency, data model, and data location. The transparency isolates applications from different data types and their corresponding management schemes.
- Evolution—The functional partitioning approach can more easily support system evolution compared to the unifying approach which employs a single database. A current database supporting an Industrial process control application can simply become the conventional database part of our model. This provides an evolutionary path from today's systems which only support the highest level of the hierarchy to the full model proposed here. The unifying approach may prove more efficient in the long run, but it cannot easily be integrated with today's systems.

5 Design Aspects and Research Issues

We have proposed a strawman architecture for system-wide real-time data management. In this section we discuss its key design aspects. We briefly examine the previous research work applicable to the design of the proposed system. In addition, we point out related research issues

that need to be further addressed for development of such a data management system.

5.1 Transaction Model

A transaction is a process that carries out data access operations. Transactions in the integrated real-time data management system are different from those in traditional databases in several ways and need to be well defined before transaction processing algorithms/protocols are designed. We characterize the transactions as follows

- *Timing constraints*—are associated with transactions. Like real-time tasks, transactions may have *hard* or *soft* deadlines, the former requiring guaranteed data access within the specified deadline while the later requiring the best effort in meeting the timing requirement. Similarly, transactions can be periodic or aperiodic. Consider an example where a transaction updates a set of actuator data in a controller every 200 ms. It is often required that the updating transaction must be completed by the end of each period to activate each control operation in time. (This is especially true for discrete control processes.) Then, such a transaction is a hard, periodic real-time transaction. Previous research work (e.g., [JRTS92]) mainly focused on aperiodic transactions with soft deadlines. Periodic transactions were only considered in [Song92], where some data may have a time associated with its validity.

- *Criticality*—is a value reflecting semantic importance of control applications. For instance, an alarm reporting transaction can be more critical than a history-collecting transaction, even though they may happen to have the same deadline. Thus, timing constraints and criticality are considered as two orthogonal parameters in defining the transaction model. Given timing constraints and criticality, a performance goal is to maximize the number of transactions with higher criticality in meeting their timing constraints.

- *ACID properties*—are used to characterize the correctness of traditional transactions. In real-time database systems (HRT-DBMS and SRT-DBMS), however, these properties may not be appropriate. As discussed earlier, transactions dealing with short data permanency do not need isolation and durability properties. Previous work in real-time databases relied on all ACID properties, including using serializability for defining and enforcing consistency. Now the relaxation of these properties and its effect need to be addressed.

- *Event triggering*—is a mechanism to support *event-based* real-time transactions. Different from the triggering mechanism often found in conventional databases (e.g., see CODASYL, System R, Sybase or SQL3 Standard) which monitors data update inside a database and enforcing data integrity constraints, the triggering mechanism that we consider relates to both internal and external events such as alarm signals and corresponding control actions. Thus, a model is needed to define such types of transactions. An event-condition-action model was developed under HiPAC project [HIPAC90]. The issue was also considered in [Korth90]. These models will be considered for defining the event-based transactions employed in our integrated real-time data management system.

- *Distribution*—refers to execution of a transaction in multiple subsystems. We consider two types of distributed transactions: *horizontal* and *vertical*, the former is similar to traditional distributed transactions running in the same type of subsystems, while the later is similar to traditional multi-level transactions running

in the different type of subsystems across the hierarchical architecture. Here the major difference is that the distributed transactions that we consider have timing constraints such as deadlines. Little attention has been paid to the distributed real-time transaction model in previous studies.

Clearly, more research effort is needed in defining the transaction model for the integrated real-time data management system, especially for relaxing ACID properties and specifying event triggering mechanisms.

5.2 Real-Time Transaction Processing

Given a transaction model, a key design aspect is to deal with transaction processing. The goal is to meet the timing constraints of any type of transactions (periodic, aperiodic, or event-based) under their specified properties. The major issues include transaction scheduling, handling of event-based transactions, and handling of distributed transactions.

Real-time transaction scheduling deals with many system components through which a transaction is executed. These components include concurrency control, buffer management, disk I/O, CPU, etc. Substantial efforts have been made in developing various real-time oriented algorithms for individual components in soft real-time databases. For example, priority-based locking protocols were initially developed in [Abbott88] [Stankovic88]; real-time optimistic concurrency control schemes were studied in [Harisa90] [Huang91b]; real-time disk scheduling was investigated in [Carey89][Abbott90][Chen91]. To design a real-time database system, however, it is necessary to use an integrated approach which takes all the processing components into account. This is because even a single component in the system which ignores timing issues may undermine the best efforts of algorithms which do account for timing constraints. This integrated approach was investigated and reported in [Huang89] and [Carey89], respectively. The previous work has laid down the foundation for developing the SRT-DBMS subsystem. However, hard real-time transaction processing remains to be addressed for HRT-DBMS. Specifically, we need to answer

- How does HRT-DBMS interact with the real-time kernel and other system components?
- How to provide guarantee for hard real-time transactions?

Event-based transaction processing deals with event monitoring, condition checking, and action execution. It is particularly needed in HRT-DBMS which interacts with external events directly. As discussed above, active transaction models were explored in, e.g., [HIPAC90] [Korth90], and architecture issues were studied in [HIPAC90]. However, more effort is needed in addressing the real-time scheduling aspect for active transactions. The questions to be answered are

- How to incorporate real-time scheduling into the event-triggering paradigm?
- How to integrate event-based transaction processing with the rest of the real-time data management system?

Distributed transaction processing deals with multiple database access. Even though it is well known in the traditional database world, it has not been addressed in the context of real-time databases. Here we must develop an integrated suite of protocols for supporting both horizontally and vertically distributed real-time transactions. Interesting research issues include

- Are the solutions developed for centralized soft real-time database systems suitable for distributed systems? For example, should conflict resolution favor use of

blocking or aborts?

- How to schedule transactions across multiple subsystems? What is an appropriate distributed commit protocol for real-time transaction processing? What global information is required by the resource schedulers at each subsystem in order to obtain good performance?

5.3 System Integration

According to our design philosophy, a plant-wide real-time data management system is built through integration of the different types of data management subsystems. Here the key design aspects include

- Subsystem interfacing under a unifying framework, and
- Provision of a system specification capability that allows application designers to specify the characteristics of an entire plant-wide real-time data management system at the system engineering time.

The interfacing provides data access transparency among different types of subsystems. This needs to be achieved through

- A unifying framework—A common set of system definitions must be specified with respect to priority assignment, transaction model, triggering mechanism, rules, etc., by which all the subsystems understand and follow. For example, transaction criticality shall be defined and interpreted consistently among the three types of subsystems. Furthermore, a set of system interface protocols and associated parameters must be defined to facilitate the inter-subsystem communication. An example is the Remote Database Access protocol - a draft international standard developed to support inter-database operations based on a client-server model [ISO/IEC DIS 9579]. Here the key issues under the scope of integrated real-time data management are how to incorporate timing, criticality and other semantic information (such as transaction commit, abort, etc.) into protocols, how to schedule the inter-subsystem communication based on this information, and how to schedule real-time transactions across heterogeneous database systems. Overall, a unifying framework will provide essential support for system interoperability and data access transparency.

- Timing behavior mapping—Different types of subsystems have different timing properties. For example, HRT-DBMS provides guaranteed access for hard real-time transactions, while SRT-DBMS only makes "best effort" scheduling for soft real-time transactions. When the two subsystems are connected, their timing behavior shall be transparent to each other, i.e., access from HRT-DBMS to SRT-DBMS appears the same as access to another HRT-DBMS or vice versa. This kind of timing behavior mapping is new and needs substantial research effort.

- Transaction mapping—Besides the timing behavior aspect, different types of subsystems may employ different transaction mechanisms for data management. As discussed earlier, for example, a traditional database usually provides ACID properties while real-time databases may only support, if not at all, a subset of the properties. Another example is that a traditional database usually defines inter-data consistency using the notion of serializability while a hard real-time database may consider temporal consistency based on data validity [Song91]. Now consider data access from a traditional database to a hard real-time database. Transactions running

across different subsystems need to be mapped in terms of correctness criteria, mechanisms, and operations. Such an issue does not exist in traditional distributed database systems and now must be addressed in the system integration.

- Data model mapping—Different types of subsystems may employ different data models. For instance, a relational data model may be used in a DBMS subsystem while an object-oriented data model may be employed in SRT-DBMS. Yet, even if these subsystems use the same model, the degree of sophistication can be different. Thus, the interfacing needs to take into account the data model mapping. Similar issues and related design schemes can be found in heterogeneous information management systems (e.g., [HN91]).

The system specification capability provides a means to define the characteristics of the entire real-time data management facility. It includes descriptions of data and their timing and storage requirements, of transactions, of knowledge based rules that might be applied to the system, and various triggers as found in active database systems. These triggers can arise both internally based on specified constraints on the data or externally due to environmental dynamics. Here the research issues arise as to

- Definition of the specification space itself,
- Development of a systematic approach in specifying the system characteristics, and
- Translation of a system specification into the corresponding system realization.

6 Concluding Remarks

In this paper, we considered distributed, real-time data management for multi-level, hierarchical control systems. The uniqueness of the problem lies in the fact that data reside at all levels of the control hierarchy and exhibit a diversity of characteristics with respect to permanency, timing, and consistency. We first analyzed some representative data according to their characteristics. Then, based on the analysis and the consideration on real-time scheduling, consistency management, data modeling, and data size and storage, we took a functional partitioning approach to the design of the data management system. In particular, we defined three types of database subsystems, HRT-DBMS, SRT-DBMS and DBMS, to accommodate and manage different types of data. We developed a multi-level, hierarchical architecture by integrating the subsystems. We also discussed the key design aspects in developing such an integrated real-time data management system. We briefly examined the previous research work applicable to the design of the proposed system, especially for SRT-DBMS. Moreover, we identified a number of research issues relevant to distributed real-time transaction modeling and processing and system integration. These issues provide new and interesting research topics for distributed real-time systems studies.

Reference

[Abbott88] R. Abbott and H. Garcia-Molina, "Scheduling Real-Time Transactions," *ACM SIGMOD Record*, March 1988.

[Abbott90] Abbott, R. and H. Garcia-Molina, "Scheduling I/O Requests with Deadlines: A Performance Evaluation," *Proc. of the 11th IEEE Real-Time Systems Symposium*, December 1990.

[Carey89] M.J. Carey, R. Jauhari and M. Livny, "Priority in DBMS Resource Scheduling,"

Proc. of the 15th Conference on Very Large Data Bases, Aug. 1989.

[Chen91] S. Chen, J. Stankovic, J. Kurose, and D. Towsley, "Performance Evaluation of Two New Disk Scheduling Algorithms for Real-Time Systems," *Journal of Real-Time Systems*, Sep. 1991.

[Harisa90] Haritsa, J.R., M.J. Carey and M. Livny, "On Being Optimistic about Real-Time Constraints," PODS, 1990.

[HIPAC90] "HIPAC: A Research Project in Active Time-Constrained Database Management," RADC-TR-90-60 Final Technical Report, prepared by Xerox Advanced Information Technology, May, 1990.

[HN91] Honeywell SSDC and Northern States Power Company, "Database Access Integration Services - Design and Application, Version 1.0," *Technical Report EPRI RP 2949-5*, Electric Power Research Institute, December 1991.

[Huang89] J. Huang, J.A. Stankovic, D. Towsley, and K. Ramamritham, "Experimental Evaluation of Real-Time Transaction Processing," *Proc. of the 10th IEEE Real-Time Systems Symposium*, December 1989.

[Huang91a] J. Huang and A. Anderson, "A Study Toward Real-Time Data Management for Industrial Control Systems," *Honeywell Technical Report C910640*, Honeywell Sensor and System Development Center, Minneapolis, Minnesota, July 1991.

[Huang91b] J. Huang, J.A. Stankovic, D. Towsley, and K. Ramamritham, "Experimental Evaluation of Real-Time Optimistic Concurrency Control Schemes," *Proceedings of the 17th International Conference on Very Large Data Bases*, Spain, September, 1991.

[ISO/IEC DIS 9579] "Remote Database Access: Part 1: Generic Model, Service and Protocol," International Organization for Standardization, 1991.

[JRTS92] Journal of Real-Time Systems, A special issue on real-time database systems, Editor J.A. Stankovic, September 1992.

[Rama92] K. Ramamritham, "Real-Time Databases," to appear in *International Journal of Distributed and Parallel Databases*, Vol. 1, No. 2, 1993.

[Stankovic88] J.A. Stankovic and W. Zhao, "On Real-Time Transactions," *ACM SIGMOD Record*, March 1988.

[Song92] X. Song and J. W.S. Liu, "How Well Can Data Temporal Consistency Be Maintained?," *Proc. of the IEEE Symposium on Computer-Aided Control System Design*, 1992.