Data Services in Distributed Real-Time Embedded Systems

Woochul Kang and Sang H. Son

University of Virginia, Charlottesville VA 22904, USA

Abstract. The computing systems are becoming deeply embedded into ordinary life and interact with physical processes and events. They monitor the physical world with sensors and provide appropriate reaction and control over it. This cyber-physical interaction, which occurs through ubiquitous embedded systems, has the potential to transform how humans interact with and control the physical world. Applications of such systems include infrastructure management and environmental monitoring. For these applications, the demand for real-time data services is increasing since they are inherently data-intensive. However, providing real-time data services in such large-scale and geographically distributed environment is a challenging task. In particular, both unpredictable communicational delays and computational workloads of large-scale distributed systems can lead to large number of deadline misses. In this paper, we propose a real-time data service architecture called DRACON (Decentralized data Replication And CONtrol), which is designed to support large-scale distributed real-time applications. DRACON uses cluster-based replica-sharing and a decentralized control structure to address communication and computational unpredictability.

1 Introduction

Recent years have seen the emergence of large-scale distributed real-time embedded (DRE) systems. They monitor the physical world with sensors and provide appropriate reaction and control over it. The scale of such cyber-physical interactions is very wide; embedded systems of such interactions range from resource-constrained stand-alone devices to global-scale networked embedded systems. This cyber-physical interaction, which occurs through ubiquitous embedded systems, has the potential to transform how humans interact with and control the physical world. Applications of such systems include advanced traffic control, global environment control, irrigation network control, and nation-wide electric power grid control. For many of these systems, providing real-time data services is essential since they need to handle large amounts of data in real-time to satisfy the timing constraints from physical processes and events. The issues involved in providing predictable real-time data services in centralized or small-scale distributed database systems have been studied and the results are promising [1][2][3].

U. Brinkschulte, T. Givargis, and S. Russo (Eds.): SEUS 2008, LNCS 5287, pp. 162–173, 2008.

In large-scale distributed environments, it is challenging to provide data services with QoS guarantees while still meeting temporal requirements of transactions. One main difficulty lies in long and highly variable remote data access delays. Unlike small-scale systems, which utilize highly deterministic local-area networks, large-scale DRE systems in wide geographical areas have to use a network that is shared by many participants for cost-effectiveness. A second challenge involves the complex interactions among a large number of nodes, which can incur unpredictable workloads for each node. For instance, a local node may experience a dramatic load increase during cascading disturbance in power grids. A third challenge is the data-dependent nature of transactions or tasks. End-to-end QoS can be achieved only when both timely access to remote data and timely computation are guaranteed. For example, QoS management schemes that do not consider the timely access to remote data [4][5] can not provide the eventual QoS guarantees in DRE systems, in which large number of nodes have complex remote data access patterns.

In this paper, we propose a distributed real-time database (DRTDB) architecture called *DRACON (Decentralized data Replication And CONtrol)*, which supports QoS in a scalable manner. In particular, DRACON features a scalable replica-sharing mechanism that enables not only bounded-delay remote data access, but also a decentralized, thus scalable, QoS control structure. DRACON is designed to support scalable real-time data services, by considering both communicational and computational unpredictability of large-scale DRE systems. Previous approaches either ignore the data-dependent nature of tasks/transactions in providing QoS guarantees [4][5], or are not scalable [3].

Data replication can help database systems meet the stringent temporal requirements of real-time applications [3][6]. A node can access local replicas, which are updated periodically for freshness, without long communication delays. However, naïve replication approaches such as full replication, which is commonly found in small-scale distributed database systems, can incur high computational and communicational overhead as the system size scales up, leading to a large number of deadline misses. In DRACON, nodes are partitioned into *clusters* for high scalability, in which replicas are shared by member nodes of the cluster, instead of having a local replica at each node. Each node of a cluster is responsible for maintaining a fair share of replicas. Further, the replica-sharing clusters are constructed such that the intra-cluster communication delay to access the shared replicas is bounded with high statistical guarantees. This clustering algorithm is implemented and tested on *PlanetLab* [7], a world-wide distributed Internet testbed. The result demonstrates that, despite the variability of wide-area networks, delay bounds can be guaranteed with a high probability.

Even though the *replica sharing* technique in DRACON should decrease the replication overhead significantly, replication still incurs non-negligible overhead, making the system sensitive to workload unpredictability. To deal with this problem, DRACON provides a decentralized and hierarchical control technique that guarantees tight deadline miss ratio under unpredictable workload. In particular, the workload control structure of DRACON is decentralized into replica-sharing

clusters. Since all remote data access requests from a node are handled within the cluster, clusters have less interactions with each other and are decoupled. This decoupling enables highly scalable decentralized control structure in DRACON.

The rest of the paper is organized as follows. The design of DRACON is described in detail in Section 2. Related work is briefly discussed in Section 3. Section 4 concludes the paper and discusses further research issues.

2 Approach

In this section, we present the design of DRACON that provides data services with QoS guarantees for large-scale DRE systems.

2.1 DRACON Architecture

Figure 1 shows the architecture of one node of DRACON. The architecture has 3 layers, *remote data access layer*, *QoS enforcement layer*, and *real-time DBMS layer*.

The *remote data access layer* enables transparent access to remote data within a bounded communication time. Remote temporal data are replicated locally to provide timely access to them. However, to avoid the high cost of full replication in large-scale distributed systems, the system is partitioned into clusters, and member nodes of each cluster share replicas of the cluster, instead of having respective local replicas. A local replica of remote data is made only if a replica is not found in the cluster that the node belongs to. Each node of a cluster is responsible for maintaining a fair share amount of replicas of remote data. The fair share amount of replicas for each node is controlled by the QoS enforcement layer to guarantee the desired QoS.

In *QoS enforcement layer*, QoS is guaranteed by feedback control loops. The primary metrics of QoS are transaction deadline miss ratio and utilization. A key intuition that affects the architecture of the feedback control loops is that the dynamics of DRACON manifest two different time-scales. At each node, fast

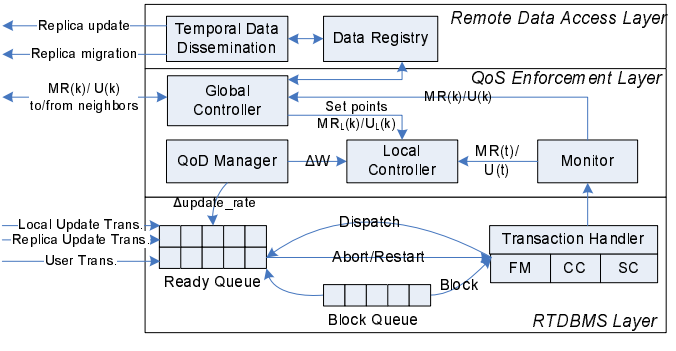


Fig. 1. The architecture of one DRACON node

dynamics are observed. These dynamics arise from changing data access patterns. At the global system level, slower dynamics are observed. They arise from changing global load distribution. Therefore, DRACON's feedback control architecture has two sets of control loops, local and global ones. In particular, since the cluster-based replica-sharing decouples clusters and decreases the interaction between clusters, DRACON's global control structure is decentralized into each cluster, making DRACON highly scalable. The global control information is exchanged only among member nodes of each cluster.

The *real-time database (RTDBMS) layer* does typical real-time transaction handling; the incoming transactions are dispatched and processed by the transaction handler. The transaction handler consists of a concurrency controller (CC), a freshness manager (FM), and a scheduler (SC). In the SC, update transactions are scheduled in the high priority queue while user transactions are scheduled in the low priority queue. Update transactions are either updates from local sensors to local data objects or updates from primary nodes to replicated temporal data objects. Within each queue, transactions are scheduled using Earliest Deadline First (EDF). The FM checks the freshness before accessing a data object using the corresponding *absolute validity interval* (avi). A sensor data object O_i is considered valid, or fresh, as long as $(current\ time - timestamp(O_i)) < avi(O_i)$ If the data object is not fresh, user transactions accessing the data object are blocked until the data object is updated.

2.2 Bounded-Delay Communication

In a distributed real-time system like the power grid, which interacts with physical processes and events, the latency of data propagation from one node to the other should be predictable and bounded in time to make a timely control decision. For example, the power grid monitoring and control in wide-area requires that status information from a control station should be delivered to the other control stations in a bounded time to prevent cascading disturbances. However, deterministic delay bound guarantees is virtually impossible to achieve in Internet-like networks. Instead, we try to achieve delay bounds with a high probability.

In DRACON, the temporal data is delivered indirectly from a source node to a destination node via a node that has a replica of the original data. Therefore, the total propagation delay of temporal data from a source node to a destination node in a different cluster is sum of inter- and intra-cluster communication delay as shown in Equation 1.

$$PropagationDelay(n_i, n_j) = Comm_{inter} + Comm_{intra}(n_j), \tag{1}$$

where $Comm_{inter}$ is the inter-cluster communication delay and $Comm_{intra}$ (n_j) is the intra-cluster communication delay of the cluster that node n_j belongs to. Figure 2 shows inter- and intra communication delays with 2 replica-sharing clusters.

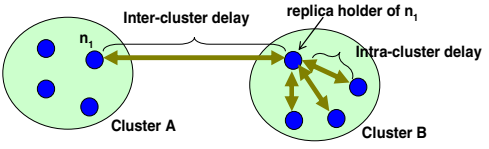


Fig. 2. Clusters and inter/intra delays

Since a temporal data of one node can be replicated by any node in the system, $Comm_{inter}$ is the communication delay between any arbitrary nodes of the system, and it is the global property of a given communication network; the bound on $Comm_{inter}$ is not affected by a cluster construction mechanism. However, the bound on $Comm_{intra}(\mathcal{N})$ of an arbitrary cluster $\mathcal{N}$ is determined by the member nodes of the cluster. $Comm_{intra}(\mathcal{N})$ is bounded by d with p probability:

$$p \leq Pr\left\{Comm_{intra}(\mathcal{N}) \leq d\right\}, \tag{2}$$

where

$$d = \max_{n_i, n_j \in \mathcal{N},} (p \text{ quantile of measured delays btw. } n_i \text{ and } n_j). \tag{3}$$

Therefore, clusters should be constructed to guarantee that the partitioned clusters satisfy the requirement of an application on its data propagation delay bound. Generally speaking, the bound on $Comm_{intra}$ of a cluster is inversely proportional to the size of the cluster. However, the computational and communicational overhead increases proportionally to the number of clusters as will be shown in the Evaluation section.

In power grids and other wide-area DRE systems, the requirement on the data propagation delay is highly related to the geographical distance between two nodes since the travel speed of physical disturbances is linearly proportional to the geographical distance. For example, disturbances travel at the speed of $500km/sec$ in power grids [8]. Therefore, the geographical distance should be considered in constructing clusters. The requirement can be stated as follows:

$$PropagationDelay(n_i, n_j) + \alpha \leq \frac{Dist(n_i, n_j)}{PSpeed}, \tag{4}$$

where $Dist(n_i, n_j)$ is the geographical distance between the two nodes, $PSpeed$ is the travel speed of the disturbance, and α is the additional overhead to process the data including actuation latency. Intuitively, this requirement tells that status data should be delivered and processed faster than the propagation of a physical disturbance.

Given this requirement, the system is partitioned into *clusters* using Algorithm-1 when the system is deployed. In Algorithm-1, clusters are recursively partitioned into smaller clusters until each cluster satisfies application's requirement. In the resulting replica-sharing structure, each cluster $\mathcal{N}$ has $Comm_{intra}(\mathcal{N})$ as its remote data access delay bounds. The temporal data

Algorithm 1. GeographicalPartitioning(cluster $\mathcal{N}$)

Input: Distance between nodes
Input: End-to-end delay bounds between nodes
foreach *node* n_i *in neighbor clusters* **do**
  foreach *node* n_j *in cluster* $\mathcal{N}$ **do**
    if $PropagationDelay(n_i, n_j) + \alpha \leq \frac{Dist(n_i,n_j)}{PSpeed}$ **then**
      continue;
    else
      partition $\mathcal{N}$ geographically into $\mathcal{N}_1$ and $\mathcal{N}_2$;
      GeographicalPartitioning($\mathcal{N}_1$);
      GeographicalPartitioning($\mathcal{N}_2$);
    end
  end
end

propagation delay bound from an arbitrary node to a node in the cluster $\mathcal{N}$ is given as $Comm_{inter} + Comm_{intra}(\mathcal{N})$. In wide-area networks, the delay bounds are statistical.

Algorithm-1 runs only when the system is first deployed since we assume that the network characteristic of future Internet-like networks for critical infrastructures will be less dynamic than the current Internet once they are deployed. In a network with highly time-varying characteristics, Algorithm-1 should be extended to include post-adjustment capability with dynamic network probing. We leave this as our future work.

Delay Bounds in Wide-Area Networks. We demonstrate the communication delay bounds that can be achieved in the current Internet with the proposed replica-sharing mechanism. This also shed light on the feasibility of the proposed approach in future Internet-like networks. Algorithm-1 is implemented and tested on *PlanetLab*, a world-wide distributed Internet testbed, with 64 nodes in eastern United States.

Before running Algorithm-1, the round-trip communication latencies were probed for 24 hours at every 30 second. Figure 3 shows the probability distribution of round-trip communication latencies between arbitrary two nodes. This graph shows that 99.999% of round-trip times between any arbitrary two nodes are less than $250ms$. The size of data packet has little impact on communication latency. The result indicates that the tight delay bounds for inter-cluster communication, $Comm_{inter}$, is $250ms$ with 99.999% statistical guarantees.

The measured delay bounds between arbitrary two nodes were provided as inputs to Algorithm-1. Instead of setting a specific requirement on the propagation delay, a cluster with the longest intra-cluster delay bound was partitioned recursively until we had 8 clusters.

The resulting replica-sharing clusters has $300km$ inter-cluster distance at a minimum. In power grid, this implies that it takes at least $600ms$ for the electric disturbance to propagate to neighbor clusters. The average intra-cluster delay bounds, $Comm_{intra}$, of the 8 clusters is $181ms$ with 99.99% probability.

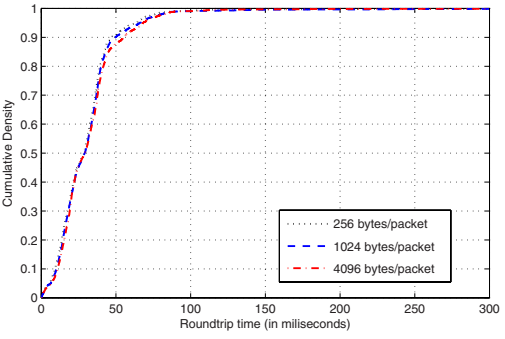


Fig. 3. Round trip time between arbitrary two nodes

Therefore, the propagation delay bounds of data from a node to the other in a different cluster is $431ms (= 250ms + 181ms)$ with 99.99% probability.

This result shows that the indirect access to temporal data through replica-sharing does not violate the requirement on the data propagation delay as long as an application requires data propagation delay no less than $431ms$. For example, it is feasible to take control action to avoid cascading electric disturbance between clusters as long as overhead for data processing and actuation takes less than about $169ms$ since it takes $600ms$ on average for a disturbance to propagate to a neighbor cluster in the stated geographical setting[1].

In practice, the requirement on the data propagation delay of most wide-area DRE systems, which were mentioned in Introduction, is much longer since the speed of physical process is much slow, e.g., the speed of road traffic, and tsunami.

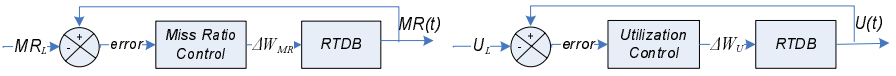
2.3 Decentralized QoS Control

In this section, we design feedback control loops for DRACON. The goal of the feedback control loops is to maintain the desired deadline miss ratio and utilization both locally and globally.

Local QoS Control. At each node, there are a local miss ratio controller and a local utilization controller as shown in Figure 4. The local feedback controllers are responsible for tracking the QoS set points set by global controllers, MR_L and U_L, and ensuring that transactions have a minimum miss ratio and the node remains fully utilized.

Each node has a desired deadline miss ratio, MR_L, and a desired utilization, U_L, as its specification from the node's global miss ratio and utilization controllers. At each sampling instant, the local miss ratio controller takes the current miss ratio, compares them with the desired miss ratio, and computes

[1] In practice, electric disturbances typically take minutes before they become serious enough to cause widespread disruption [9].



(a) Miss ratio controller (b) Utilization controller

Fig. 4. Local controllers

the local workload control signal, ΔW_{MR}, which is used to adjust the utilization at the next sampling period. The local utilization control loop takes the similar control action as the local miss ratio controller. Employing a utilization control loop is to avoid a trivial solution, in which all the miss ratio requirements are trivially satisfied by under-utilizing the system. At each sampling instant, we set the current control signal $DeltaW = Minimum(\Delta W_{MR}, \Delta W_U)$ to support a smooth transition from one system state to another.

The target utilization from the local controller is achieved by switching between the on-demand update scheme and the immediate update scheme for selected temporal data objects. The candidate data objects of this *dynamic update-mode switch* are selected based on the communication delays between a primary node n_i and its replica holder node n_j of data object O_i. The $avi(O_i)$ should be large enough for O_i to be still fresh even when the data object is updated on-demand as shown in Equation 5.

$$Comm_{inter} + Comm_{intra}(n_j) + \beta < avi(O_i). \tag{5}$$

In the equation, $Comm_{inter} + Comm_{intra}(n_j)$ is the communication delay for on-demand update, and β is the expected processing time to retrieve O_i at the primary node. Since the communication delay bounds of any two arbitrary nodes are known with high statistical guarantees from the system partitioning procedure, we can get the set of candidate data objects, O_{cand}, which satisfy the above condition. When the estimated load adaptation from the update-mode switch of data object O_i is $U_c(O_i)$, the maximum adjustable load is $\sum_{O_i \in O_{cand}} U_c(O_i)$. After the candidate data objects for the update-mode switch are selected, the notion of *Access Update Ratio (AUR)* for a data object is applied O_i as follows to select target data objects:

$$AUR[i] = \frac{Access\ Frequency[i]}{Update\ Frequency[i]}. \tag{6}$$

AUR models the ratio of the benefit (*Access Frequency*) to the cost (*Update Frequency*) of O_i. It is clear that data objects with high AUR should be updated aggressively; if they are out-of-date when accessed, potentially multiple transactions may miss their deadlines waiting for the updates. Therefore, data objects are considered in the order of smaller AUR for update mode switch.

In the design of controllers, each local RTDB is modeled as an *first-order time-invariant linear model*, and *proportional integral* (PI) control law is used for controllers. The details of our local controller design procedure can be found in [2].

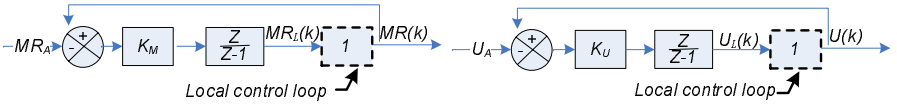
Global QoS Control and Load Balancing. In DRACON, replicas are shared by cluster member nodes; hence, the nodes in the same cluster have closer interactions. This close interaction in a cluster incurs a changing load distribution. Global controllers at each node balance this load distribution.

At each global sampling period, global controllers exchange utilization/miss ratio information with other member nodes in the same cluster to calculate the average miss ratio, MR_A, and utilization, U_A. The global control outputs of each node are determined from the following difference equations:

$$MR_L(k) = MR_L(k-1) + K_M(MR_A - MR(k-1)). \tag{7}$$

$$U_L(k) = U_L(k-1) + K_U(U_A - U(k-1)). \tag{8}$$

The global control outputs, $MR_L(k)$ and $U_L(k)$, are the set points for the local miss ratio controller and the local utilization controller, respectively. The controller gains, K_M and K_U, determine the characteristics of the controllers. Note that global control information is exchanged only among cluster members since cluster-based replica sharing decouples each cluster from the others. Furthermore, the control information delivery time is highly predictable since the communication delay in an arbitrary cluster c is bounded by $Comm_{intra}(c)$ with high statistical guarantees.



(a) Global Miss ratio control (b) Global Utilization control

Fig. 5. Block diagram for the global system

The interaction between local and global controllers is modeled by simplifying a local feedback control loop into an identity transfer function. This simplification of the local feedback control loop is possible since a local feedback control loop has several orders of magnitude faster dynamics than a global control loop. When a system has multiple dynamics, the fast mode of the system can be discarded for model simplification [10][11]; this enables modeling of a complex system. With this technique, the global system can be modeled as shown in Figure 5. In the figure, the blocks with an identity transfer function are local feedback control loops. Intuitively, modeling a local feedback control loop into an identity transfer function means that a QoS set point (MR_L or U_L) from a global controller is achieved instantaneously by the local controller and the state is maintained until the next global sampling period.

In the above block diagrams, the poles of closed loops are $\frac{1}{K_m+1}$ and $\frac{1}{K_U+1}$, respectively. A discrete system is stable if and only if the poles of the closed loop are inside a unit circle [12]. Therefore, the closed loop for the global system

in Figure 5 is stable if positive values for the controller gains, K_M and K_U, are selected. The final controller parameters are determined in consideration of other desired characteristics of the closed loop system such as a settling time and an overshoot.

Once target miss ratio and utilization are set for local controllers, they are tracked by local controllers at each node. However, the maximum achievable load adjustment from a local control loop can be limited by data freshness requirements; the update mode of a data object can be switched to on-demand update only if Equation 5 is satisfied. The remaining workload adaptation is achieved by migrating replicas between cluster member nodes. At each global sampling period k, the amount of load that a node i needs to transfer (or to receive), $\Delta TW_i(k)$, is the difference between the required load adaptation to achieve the new set points, $\Delta W_i(k)$, and the local controller's maximally achievable load adaption, $AW_i(k)$:

$$\Delta TW_i(k) = \Delta W_i(k) - AW_i(k). \tag{9}$$

If $\Delta TW_i(k)$ is positive, the node is overloaded and it can not be fully controlled by its local controller. Therefore, some replicas are migrated to neighbor nodes, which have a negative $\Delta TW(k)$, until $\Delta TW_i(k)$ becomes less than or equal to zero. Since local target set points, MR_L and U_L, are determined to track the average miss ratio and the utilization of cluster nodes, $\sum_i \Delta TW_i(k)$ approaches zero, making each cluster balanced.

3 Related Works

Distributed real-time databases (DRTDBs) have drawn research attention in recent years [13][14]. Wei et. al. proposed replication techniques for DRTDBs [3][6]. However, their approaches do not consider communication delay bounds in wide-area networks, and their workload control mechanism is not scalable in large-scale systems.

Feedback control has been applied to QoS management in real-time systems due to its robustness against unpredictable operating environments [15][1]. In particular, several distributed feedback control schemes have been proposed for distributed real-time systems [5][11][4]. However, those approaches target small-scale distributed systems and do not consider communication delays in wide-area networks.

4 Conclusions

DRACON has been designed to provide a highly scalable data service with QoS guarantees in large-scale distributed real-time systems. DRACON features a replica sharing mechanism that enables bounded-delay access to remote data

in a highly scalable manner. Furthermore, the replica sharing resolves the complex interactions between nodes by decoupling clusters, allowing a decentralized, hence scalable, QoS control structure.

Currently we are evaluating the performance of DRACON in large-scale applications. The preliminary results show that DRACON is capable of achieving a significant performance improvement compared to baselines in providing the desired miss ratios while maintaining high utilization in wide-area networking environments. While these results are promising, there are several technical challenges that need further research. We plan to address the following questions:

1. *Variability of Network:* How does the variability of the network affect the replica-sharing scheme? Our preliminary work is based on the assumption that the communication network is stable in its delay characteristics. However, if a communication network's end-to-end delay characteristics change significantly in different time periods, e.g., days and nights, the clusters need to be post-adjusted at runtime. A dynamic scheme for network status monitoring and modeling is required to achieve that.
2. *Inter-Cluster Load Balancing:* How can the load-imbalance between clusters be resolved? In DRACON, each cluster is decoupled by cluster-level replica-sharing. However, it is still possible that there exist some interactions between clusters. In such situations, we may need higher level controllers/load balancers to resolve the inter-cluster load imbalance. What would be the constraints in designing such inter-cluster level controllers?
3. *Dependability:* DRACON considers real-time data replication to reduce data access time. However, replication is also effective in improving the dependability of the system. By having multiple replicas of sensor data, the system can guarantee that the critical information is accessible on time even when some nodes are unavailable, although it requires additional cost. We will investigate how to improve the dependability with minimal cost, using the idea of adaptive virtual replication we have developed for embedded sensor networks [16].
4. *Interaction with physical processes:* How does the timing constraints from physical processes affect the QoS management architecture? We will use real-world examples to investigate appropriate QoS models for networked embedded systems.

References

1. Kang, K.D., Son, S.H., Stankovic, J.A.: Managing deadline miss ratio and sensor data freshness in real-time databases. IEEE Transacctions on Knowledge and Data Engineering 16(10), 1200–1216 (2004)
2. Kang, W., Son, S.H., Stankovic, J.A., Amirijoo, M.: I/O-aware deadline miss ratio management in real-time embedded databases. In: The 28th IEEE Real-Time Systems Symposium (RTSS) (December 2007)
3. Wei, Y., Son, S.H., Stankovic, J.A., Kang, K.D.: Qos management in replicated real time databases. In: RTSS 2003: Proceedings of the 24th IEEE International Real-Time Systems Symposium (2003)

4. Wang, X., Jia, D., Lu, C., Koutsoukos, X.: DEUCON:Decentralized End-to-End Utilization Control for Distributed Real-Time Systems. IEEE Transactions on Parallel and Distributed Systems 18(7), 996–1009 (2007)
5. Stankovic, J.A., He, T., Abdelzaher, T., Marley, M., Tao, G., Son, S., Lu, C.: Feedback control scheduling in distributed real-time systems. In: RTSS 2001: Proceedings of the 22nd IEEE Real-Time Systems Symposium (RTSS 2001) (2001)
6. Wei, Y., Shlinger, A.A., Son, S.H., Stankovic, J.A.: ORDER: A Dynamic Replication Algorithm for Periodic Transactions in Distributed Real-Time Databases. In: RTCSA 2004, Gothenburg, Sweden (August 2004)
7. Anderson, T., Peterson, L., Shenker, S., Turner, J.: Overcoming the internet impasse through virtualization. IEEE Computers 38(4), 34–41 (2005)
8. Thorp, J.S., Seyler, C.E., Phadke, A.G.: Electromechanical wave propagation in large electric power systems. IEEE Transactions on Circuits and Systems 45(6), 614–622 (1998)
9. Birman, K.P., Chen, J., Hopkinson, E.M., Thomas, R.J., Thorp, J.S., Renesse, R.V., Vogels, W.: Overcoming communications challenges in software for monitoring and controlling power systems. In: Proceedings of the IEEE (2005)
10. Nils, R., Sandell, J., Varaiya, P., Athans, M., Safonov, M.G.: Survey of decentralized control methods for large scale systems. IEEE Transactions on Automatic Control 23(2), 108–128 (1978)
11. Lin, S., Manimaran, G.: Double-loop feedback-based scheduling approach for distributed real-time systems. In: Conference on High Performance Computing (HiPC) (2003)
12. Hellerstein, J.L., Diao, Y., Parekh, S., Tilbury, D.M.: Feedback Control of Computing Systems. Wiley, IEEE press (2004)
13. Peddi, P., DiPippo, L.C.: A replication strategy for distributed real-time object-oriented databases. In: Symposium on Object-Oriented Real-Time Distributed Computing, pp. 129–136 (2002)
14. Wei, Y., Prasad, V., Son, S.H., Stankovic, J.A.: Prediction-based qos management for real-time data streams. In: RTSS 2006, pp. 344–358 (2006)
15. Lu, C., Stankovic, J.A., Son, S.H., Tao, G.: Feedback control real-time scheduling: Framework, modeling, and algorithms. Real-Time Syst. 23(1-2), 85–126 (2002)
16. Mathiason, G., Andler, S., Son, S.H.: Virtual full replication for saclable and adaptive real-time communication in wireless sensor networks. In: Second International Conference on Sensor Technologies and Applications, Cap Esterel, France (August 2008)