

DRACON: QoS Management for Large-Scale Distributed Real-Time Databases *

Woochul Kang, Sang H. Son, and John A. Stankovic
Department of Computer Science
University of Virginia
{wk5f,son,stankovic}@cs.virginia.edu

Abstract

The demand for real-time data services is increasing in many large-scale distributed real-time applications including advanced traffic control, global environment control, and the nation-wide electric power grid control. However, providing quality-of-service (QoS) for 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. We have designed a distributed real-time database architecture called DRACON (Decentralized data Replication And CONTROL), which enables QoS guarantees for large-scale distributed real-time applications. DRACON couples cluster-based replica-sharing and a decentralized control structure to address communication and computational unpredictability, simultaneously. The cluster-based replica-sharing mechanism not only enables scalable and bounded-delay access to remote data with high probability, but also decouples clusters to have less interaction, allowing a decentralized, thus scalable, QoS control structure. The simulation study demonstrates that DRACON's decentralized QoS control structure combined with a decentralized replica-sharing structure provides robust and predictable QoS guarantees in a highly scalable manner.

1 Introduction

Recent years have seen the emergence of large-scale distributed real-time embedded (DRE) systems, which includes advanced traffic control, global environment control, irrigation network control, and the 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 [8][19]. However, we are not aware of research results for providing data services with *Quality-of-Service* (QoS) guarantees in large-scale distributed real-time database environments.

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 major 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 [18][15] 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 guarantees QoS in a highly 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. The contributions of this paper are three-fold:

1. a replica-sharing mechanism that guarantees bounded-delay access to remote data with high probability,
2. a decentralized feedback control architecture to control unpredictable workloads both locally and globally, and
3. an extensive evaluation of the proposed approach through simulation in the context of wide-area power grid control.

To the best of our knowledge, this is the first paper on scalable QoS management in DRTDBs, which considers 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 [18][15], or are not scalable [19].

Data replication can help database systems meet the stringent temporal requirements of real-time applications [19]. 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

*This research work was funded in part by NSF CNS-0614886.

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* [1], 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 decreases the replication overhead significantly as will be shown in the Evaluation section, 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 evaluation results demonstrate that QoS can be provided only if both timely access to remote data and timely processing of data even with transient overloads are guaranteed. The study shows that DRACON's decentralized replication and control scheme gives a robust and controlled behavior of DRTDBs in a highly scalable manner.

The rest of the paper is organized as follows. Section 2 presents the power grid monitoring and control problem to illustrate the challenges posed by large-scale DRE systems. In Section 3, the system model is presented. The design of DRACON is described in detail in Section 4. Section 5 shows the details of the simulation settings and presents the evaluation results. Related work is discussed in Section 6 and Section 7 concludes the paper and discusses future work.

2 Large-Scale Power Grid Control Problem

In this section, we consider the large-scale monitoring and control problem of a power grid as a concrete example to illustrate the challenges of large-scale DRE systems.

The power grid is a complex large-scale distributed system; in North America, the power grid involves about 3,500 utility organizations [6]. Regional power control centers owned by power companies maintain and control the flow of electricity over the grid, supplying electricity to meet the demand. Since power utilities are physically inter-connected to other power utilities, an overload or disconnection at one power system can result in a cascading effect as shown in 2003 blackout [6]. This close coupling between distributed power systems requires coordination between regional control centers in wide-area to provide a better understanding of the grid-wide situation.

However, current wide-area control schemes are rudimentary. Some wide-area control schemes such as *remedial action schemes (RAS)* have been developed, but they depend on dedicated point-to-point communication, which can not be a general scheme because of high cost and inflexibility. Therefore, it is anticipated that an Internet-like network will emerge in the coming decade for wide-area power grid control [2]. One of the key challenges in using such Internet-like wide-area networks for critical infrastructures is long end-to-end communication delay and high variability. Several wide-area monitoring/control architectures have been proposed to address the communication delay problem in such Internet-like networks [17][2].

Furthermore, coping with grid-wide situation can not be achieved by 'fast' communications only. It requires real-time data processing and analysis of large volume of sensor data to make a timely control decision. Traditionally, real-time databases (RTDBs) have been a key component that enables real-time data processing and analysis in SCADA/EMS (Supervisory Control and Data Acquisition/Energy Management System) for power grids [4][14]. It is not unusual to have between 50,000 and 100,000 telemetry/measurement points in a control center database. Measurements are often taken or derived values computed at intervals on the order of 2 seconds apart. This leads to a relatively large database with dynamic data modifications and entry [14]. Grid-wide situation monitoring and control require these local RTDBs to cooperate by exchanging monitoring information in wide-area, forming a large-scale distributed real-time database (DRTDB). However, one major challenge in such a large-scale DRTDB is that complex interaction between local RTDBs can incur unpredictable workloads, which is another source of deadline misses.

We use this large-scale wide-area power grid control problem throughout the paper.

Finally, note that DRACON is not intended to replace current local data exchange and control mechanism such as SCADA. DRACON complements them when a large number of control centers are networked in a geographically wide-area.

3 Distributed Real-Time Database Model

In this section, an overview of our DRTDB model is discussed.

3.1 Data and Transaction Model

In our data model, data objects can be classified into two classes, *temporal data* and *non-temporal data*. Temporal data are sensor data from physical world and updated periodically by *update transactions*. In contrast, non-temporal data do not change dynamically with time. For instance, in SCADA/EMS update transactions are invoked to update temporal data when sensor values are read from *remote terminal units (RTUs)*; the temporal data from RTUs includes data such as voltage, line frequency, phase angle of the phaser, status of equipment, and

so forth. Temporal data objects have validity intervals. Validity intervals are used to maintain the temporal consistency between the real-world state and sensor data in the database. A sensor data object O_i is considered valid, or fresh, as long as $(\text{current time} - \text{timestamp}(O_i)) < \text{avi}(O_i)$, where $\text{avi}(O_i)$ is the *absolute validity interval* of O_i .

User transactions are periodic or aperiodic queries issued by applications to analyze the system status, which involves both temporal data from sensors and non-temporal data. For example, applications of SCADA/EMS perform operational analysis such as power flow analysis, transient stability analysis, and on-line safety analysis by issuing queries to underlying RTDBs [4][14]. Unlike traditional e-commerce applications, data freshness and timeliness of queries are more important than data integrity in DRE systems. To this end, ACID properties of transactions are often relaxed [20].

3.2 Distributed Database Model

In this paper, we consider a DRTDB which consists of a group of local databases connected by a wide-area network. Each local database is called a *node* and it is the centerpiece that enables the collection and analysis of a large amount of critical sensor data in a local control and automation system.

Each node maintains all or most data in main-memory for fast response time and high predictability.¹ Each node hosts a set of temporal data objects and non-temporal data objects. To overcome long data access delays to remote data objects, data objects can be replicated in a local node. Once a replica is made for a remote data object, it is updated periodically by the *primary node*, which maintains the original copy of the data object. In this paper, we assume only temporal sensor data are replicated and updated periodically since effective temporal data dissemination is the key issue. The replication model follows *single primary, multiple replica* strategy; write operations can occur only on the primary of the data object.

This DRTDB approach provides several key advantages over *ad-hoc* distributed data management schemes. The temporal data dissemination through replication enables each control center to make a globally consistent control decision, which is hardly achievable without transactional support from DRTDBs. Moreover, the standardized data access interfaces such as SQL can facilitate the interoperability between heterogeneous nodes.

3.3 Major QoS Metric

In our DRTDB model, the main QoS metric is *deadline miss ratio*. The miss ratio is defined as:

$$MR = 100 \times \frac{\#tardy}{\#tardy + \#timely} (\%), \quad (1)$$

where $\#tardy$, and $\#timely$ represent the number of transactions that have missed and met their deadlines, respectively. Since database workloads and access patterns of transactions

¹This model can be extended to include I/O with added complexity as in our previous work [9].

vary dynamically, it is reasonable to assume that some deadline misses are inevitable.

4 Approach

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

4.1 DRACON Architecture

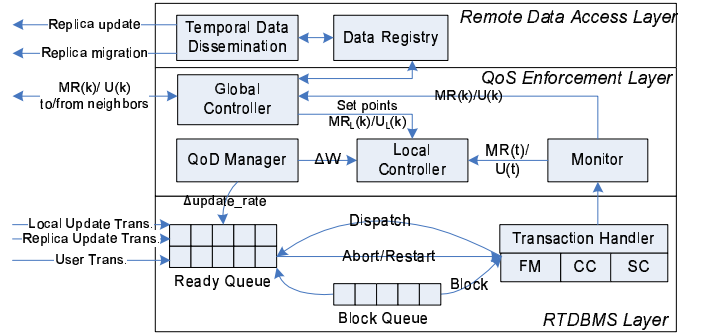


Figure 1. The architecture of one DRACON node.

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.

To guarantee the desired deadline miss ratio even in the presence of unpredictable workloads, *QoS enforcement layer* exploits two feedback control loops. 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 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 *avi*. If the data object is not fresh, user transactions accessing the data object are blocked until the data object is updated.

4.2 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 other control stations in a bounded time to prevent cascading disturbances. However, deterministic delay bound guarantees are 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, $T_D(n_i, n_j)$, of temporal data from a source node n_i to a destination node n_j in a different cluster is the sum of inter- and intra-cluster communication delay as shown in Equation 2.

$$T_D(n_i, n_j) = Comm_{inter} + Comm_{intra}(n_j), \quad (2)$$

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.

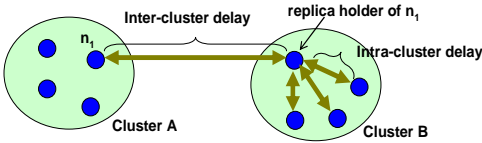


Figure 2. Clusters and inter/intra delays.

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 $Comm_{intra}(N)$ of an arbitrary cluster N is determined by the member nodes of the cluster. $Comm_{intra}(N)$ is bounded by d with probability p :

$$p \leq Pr \{Comm_{intra}(N) \leq d\}, \quad (3)$$

where

$$d = \max_{n_i, n_j \in N} (p \text{ quantile of measured delays btw. } n_i \text{ and } n_j). \quad (4)$$

Therefore, clusters should be constructed to guarantee that the partitioned clusters satisfy the requirement of an application on its data propagation delay bound. In general, 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 [16]. Therefore, the geographical distance should be considered in constructing clusters. This requirement can be stated as follows:

$$T_D(n_i, n_j) + \alpha \leq T_P(n_i, n_j), \quad (5)$$

$$= \frac{distance(n_i, n_j)}{disturbance \text{ propagation speed}} \quad (6)$$

where $T_D(n_i, n_j)$ is the data propagation delay between the two nodes, $T_P(n_i, n_j)$ is the traveling delay of physical disturbance between the two nodes, 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.

Algorithm 1: GeoSpeedPartitioning(cluster N)

Input: distances between arbitrary two nodes

Input: measured delays between arbitrary two nodes

1 **if** all pairs (n_i, n_j) in N satisfy Equation 5 **then**

2 | continue;

3 **else**

4 | select n_i and n_j with the greatest $\frac{T_D(n_i, n_j)}{T_P(n_i, n_j)}$;

5 | $N_1 = \{n_i\}; N_2 = \{n_j\}$;

6 | $N = N - \{n_i, n_j\}$;

7 **foreach** node n in N **do**

8 | **if** $\frac{T_D(n, n_i)}{T_P(n, n_i)} \leq \frac{T_D(n, n_j)}{T_P(n, n_j)}$ **then**

9 | | $N_1 = N_1 \cup \{n\}$;

10 | **else**

11 | | $N_2 = N_2 \cup \{n\}$;

12 | **end**

13 **end**

14 GeoSpeedPartitioning(N_1);

15 GeoSpeedPartitioning(N_2);

16 **end**

Given this requirement, the system is partitioned into a set of 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 line 4 – 12, two nodes with the greatest ratio of communication delay to disturbance traveling delay, or $\frac{T_D(n_i, n_j)}{T_P(n_i, n_j)}$, are chosen, and the remaining nodes are attached to the closer one

based on the same criteria. In the resulting replica-sharing structure, the total number of clusters should be much smaller than the total number of nodes. Otherwise, the replica-sharing via clustering has no gain; in such situation, each pair of nodes should have a dedicated communication link. However, this is unlikely even with the current Internet as will be shown shortly. The worst-case running time of Algorithm-1 is $O(n^3)$ when N_1 and N_2 are highly unbalanced on every recursion. However, the average case running time is $O(n^2 \log n)$, and we do not further optimize the algorithm since it runs once when the system is first deployed. We assume that 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.

4.2.1 Delay Bounds in Wide-Area Networks

We demonstrate the communication delay bounds that can be achieved in the current Internet with the proposed cluster partitioning. This also shed light on the feasibility of the proposed approach in future Internet-like networks. Algorithm-1 is implemented and tested on *PlanetLab* [1] with 64 nodes from 26 institutions in eastern United States.

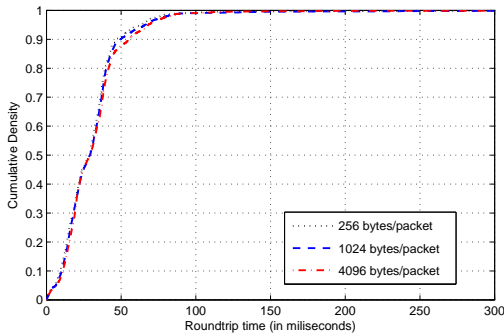


Figure 3. Communication latency between arbitrary two nodes

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

the 8 clusters is 181ms with 99.99% probability. 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 implies 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.

4.3 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.

4.3.1 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.

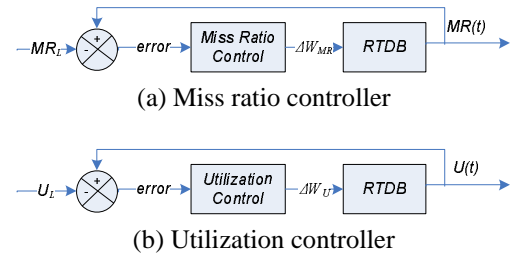


Figure 4. Local controllers.

At each sampling instant, the local miss ratio controller takes the current miss ratio, compares them with the desired miss ratio, and computes 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 $\Delta W = \text{Minimum}(\Delta W_{MR}, \Delta W_U)$ 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.

$$Comm_{inter} + Comm_{intra}(n_j) + \beta < avi(O_i). \quad (7)$$

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 O_i is applied as follows to select target data objects:

$$AUR[i] = \frac{Access\ Frequency[i]}{Update\ Frequency[i]}. \quad (8)$$

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 [9].

4.3.2 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. These close interactions in a cluster incur 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:

$$MR_L(k) = MR_L(k-1) + K_M(MR_A - MR(k-1)). \quad (9)$$

$$U_L(k) = U_L(k-1) + K_U(U_A - U(k-1)). \quad (10)$$

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.

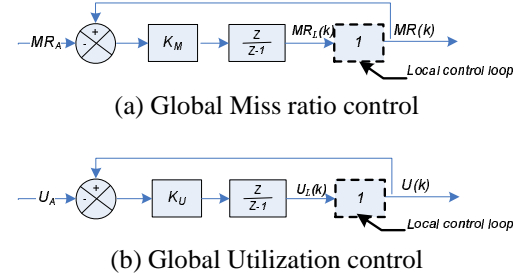


Figure 5. Block diagram for the global system

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.² When a system has multiple dynamics, the fast mode of the system can be discarded for model simplification [12][10]; 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 [7]. Therefore, the closed loops for the global system in Figure 5 are 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 7 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, $AW_i(k)$:

$$\Delta TW_i(k) = \Delta W_i(k) - AW_i(k). \quad (11)$$

If $\Delta TW_i(k)$ is positive, the node is overloaded and it can not be fully controlled by its local controller. Therefore, some

²In our evaluation, the sampling intervals of local and global control loops are 1 second and 10 seconds, respectively.

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.

5 Evaluation

In this section, we describe the simulation settings and present the results of the performance evaluation. The objective for our evaluation is to study the performance impact of coupling the replica-sharing scheme with the decentralized QoS control algorithms under unpredictable communication delays and workloads. In the experiments, we consider a QoS specification for each node, which limits the average deadline miss ratio below MR_L that is set by the global controller at each node.

5.1 Simulation Settings

Parameter	Value
# nodes	64
# clusters	8
Inter-cluster distance	400 Km - 1265 Km
End-to-end network delay	Pareto Dist.
# temporal data	10,000/node
Temp. data update freq.	2 seconds
Temp. data AVI	4 seconds
Update Trans exec time.	2 ms

Table 1. System parameter settings.

Parameter	Value
Exec. time	Uniform(0.5 - 2)ms
Exec. time Est Error	Normal(20%,10%)
Temporal data #/Trans.	Uniform(1,8)
Deadline	\propto distance

Table 2. User transactions settings.

For the simulation, we have chosen system parameter values that are, in general, representative of wide-area power grid monitoring and control [14]. The general system parameter settings are given in Table-1.

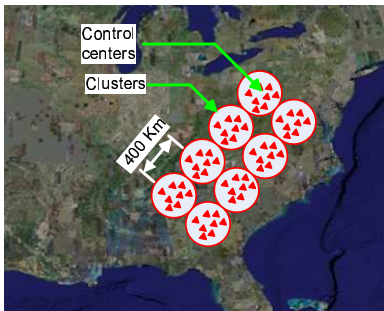


Figure 6. The modeled power grid.

Figure 6 shows a modeled power grid that has 64 regional control centers, or nodes. The nodes are partitioned into 8 clus-

ters. The clusters are geographically located in a 4×2 equal-spaced grid, and the distances between neighbor clusters are 400km. This geographic setting is similar to *Eastern Interconnect* of US power grid, and it is used to derive approximate end-to-end communication delay parameters and proper transaction deadlines. Each node has 10,000 monitoring points and 500 *remote terminal units* (RTUs); each RTU manages 200 monitoring points. Data is collected by scanning each RTU every 2 seconds. One polling of a RTU updates 200 temporal data, which correspond to monitoring points. Because of the confidentiality and the large volume of data, these raw data from sensors can not be delivered directly to other control centers, which might be run by competing utility companies. Instead, the temporal data are analyzed by periodic aggregate queries, which derive additional 50 summarized status data every 2 seconds [2]. Only the derived status data are replicated between nodes for wide-area grid control.

The settings for the user transactions workload are given in Table 2. User transactions analyze the system status, which includes power flow status, transient stability, and safety [4][14]. The execution time for one operation is between 0.5 ms to 2 ms. The deadline of a user transaction t at node n is:

$$deadline_t = \min_{n_i \in N} (T_P(n, n_i)), \quad (12)$$

where N is a set of primary nodes of the replicated remote data objects that the transaction t is accessing; The further the distance from n to the primary node of the data object, the longer deadline is given for the transaction. This reflects the disturbance propagation delay (T_P) in the power grid. At each node, the user transactions arrive according to a Poisson distribution and the average arrival rate is shown.

The parameter values of end-to-end communication delays between nodes are estimated using the PlanetLab experiments in Section 4.2. *Pareto* distributions are chosen to model the end-to-end communication delays since the distribution has been shown effective to model the high variability of wide-area networks [21]. Different pairs of PlanetLab nodes with different geographical distances are chosen to reflect the geographical impact on end-to-end communication delays. The optimization-based fitting tool of Matlab was used to fit the end-to-end delay observations to Pareto distributions. The network delay is assumed to be not affected by the data size since our PlanetLab experiments show no significant delay difference between different data sizes. In the simulation, the overhead of running controllers is not modeled since the controllers are simple enough to ignore the computation cost. The sampling periods are set to 1 second and 10 seconds for local controllers and global controllers, respectively.

All simulation results are based on at least 10 runs and the 95% confidence intervals are less than 10% of the mean values.

5.2 Baselines

To evaluate our approach (**DRACON**), we compare the performance of DRACON with three baseline algorithms.

Full replication (Full): All temporal data objects are fully replicated to each node. No replica sharing is used.

On-demand (OnDemand): Temporal data objects are not replicated, but accessed on-demand.

DRACON without feedback control (DRACON-NoFC): Temporal data are replicated using the cluster-based replica-sharing scheme. However, no feedback control scheme is applied for QoS management.

5.3 Performance Evaluation Results

5.3.1 Varying Loads

In this set of simulations, we apply workloads from 80 transactions/sec to 240 transactions/sec per node. The miss ratios and utilization of DRACON and baselines have been observed.

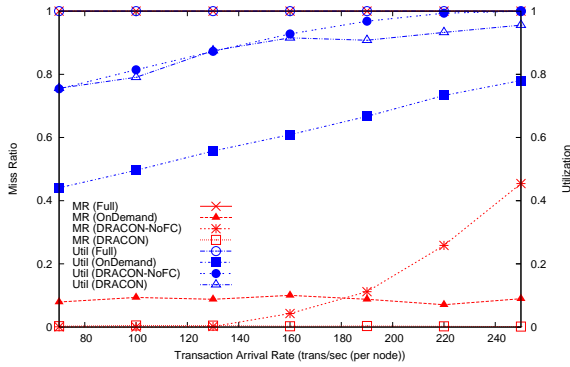


Figure 7. Varying loads.

As shown in Figure 7, the full replication shows the worst performance. Even for the least workload, it has 100% utilization and miss ratio; the measured load was more than 300%. In large scale distributed applications, full replication can not be considered for this high overhead. In contrast, OnDemand has the lowest overhead. Even for the highest workload, its utilization is lower than 80%. However, OnDemand has almost constant 10% deadline miss ratio regardless of the workload. These deadline misses are caused by communication delays, not by system overloads. DRACON achieves the best performance in terms of both the miss ratio and throughput. The cluster-based replica sharing mechanism of DRACON-NoFC and DRACON incurs about 30% increased load than OnDemand. When the workload is less than 130 trans/sec, the increased load does not overload the system. However, in DRACON-NoFC, as the workload increases to more than 130 trans/sec, the increased load from the replica sharing begins to overload the system, hence incurring deadline misses. In DRACON, the increased loads from the replica sharing are effectively controlled by the feedback control loop. The feedback control loop of DRACON keeps both the miss ratio and utilization as specified via dynamic update-mode switching and cluster-level load balancing.

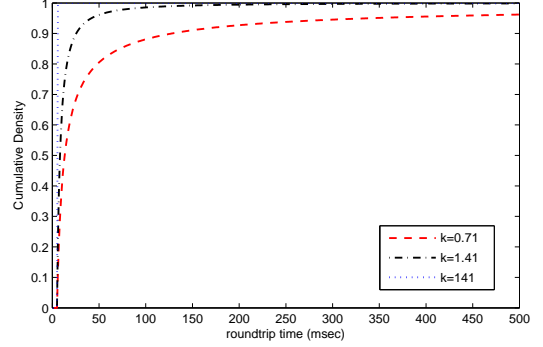


Figure 8. Pareto distributions ($X_m = 5msec$).

5.3.2 Varying Communication Latency and Variability

We repeat the same experiment of Section 5.3.1 with different shape parameters, k of Pareto distributions. The original location parameters, X_m , of the Pareto distributions are used without change. Figure 8 shows the CDFs of the Pareto distribution with different shape parameters when X_m is 5.³ In Figure 8, the CDF with $k = 1.41$ represents the current Internet observed from PlanetLab. Figure 9-(a) and 9-(b) are the results when the shape parameter, k , is changed to $\frac{1}{2}$ times and 100 times of the original shape parameters, respectively. The settings represent two extreme cases. When $k = original\ k \times \frac{1}{2}$, the network suffers from long end-to-end communication latency with high probability. In contrast, when $k = original\ k \times 100$, the network has almost constant end-to-end communication latency.

As shown in Figure 9, DRACON still achieves the best performance compared to the baselines approaches; it shows the lowest miss ratios in both cases. When $k = original\ k \times \frac{1}{2}$, DRACON has about a 20% miss ratio regardless of the workloads. These deadline misses from communication delays are inevitable in networks with such long communication delays. If the communication network shows such high variability and end-to-end latency, it can not be considered for wide-area distributed monitoring and control applications.

When $k = original\ k \times 100$, the miss ratios of both OnDemand and DRACON are almost zero. This implies that OnDemand is the best choice since it incurs the least communication/computation overhead without causing deadline misses. However, the end-to-end delay characteristic of the hypothesized network is hardly achievable in Internet-like wide-area networks.

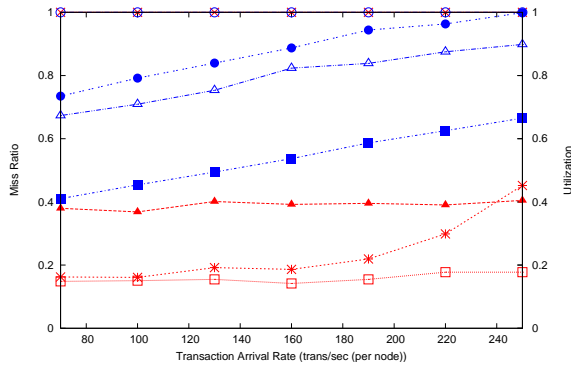
5.3.3 Load Balancing

In this experiment, we evaluate the load balancing function of DRACON. The performance of DRACON is compared with the following baseline algorithm, in addition to DRACON-NoFC (discussed previously).

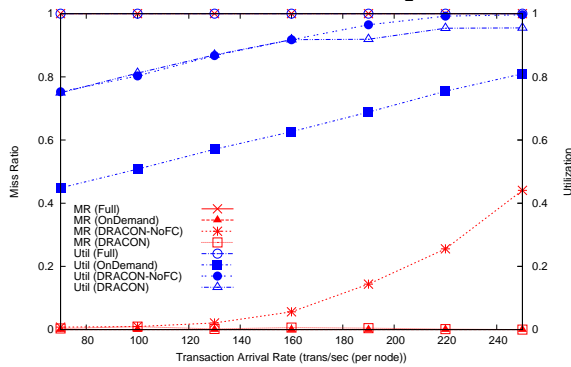
DRACON with local feedback control (DRACON-LFC):

Nodes employ only local feedback controllers in addition to the cluster-based replica sharing scheme.

³Since the variance of a Pareto distribution is infinite when $k \leq 2$, the variability of a distribution can not be described with its variance [21].

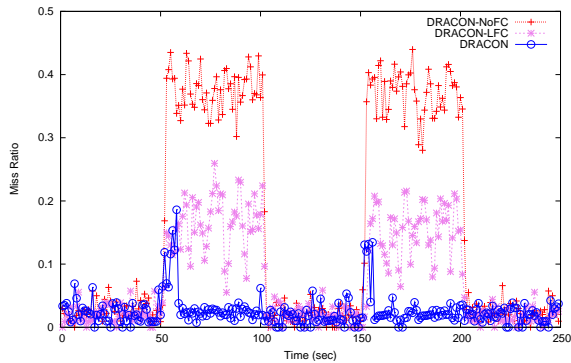


(a) $k = \text{original } k \times \frac{1}{2}$

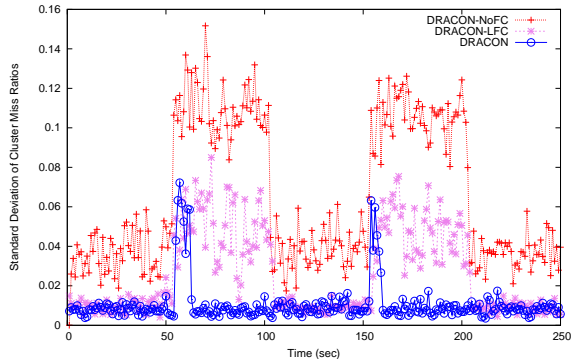


(b) $k = \text{original } k \times 100$

Figure 9. Varying Communication Latency and Variability.



(a) Transient behavior of an overloaded node.



(b) Standard deviation of cluster miss ratios.

Figure 10. Load balancing

The bursts begin at the 50th second and the 150th second and each one lasts for 50 seconds. Figure 10 shows the transient behavior of the node and the standard deviation of miss ratios of the cluster that it belongs to, respectively. The standard deviation measures the performance differences between the cluster nodes. The lower the standard deviation value, the more balanced the system loads are. As shown in the figures, both DRACON-NoFC and DRACON-LFC remain unbalanced throughout the workload burst periods; with DRACON, the system becomes balanced within 10 seconds. DRACON-LFC decreases the miss ratio using dynamic update-mode switching. However, there is a limitation in this scheme since switching the update-mode is a plausible option only for data objects with long absolute validity intervals; oblivious update-mode switching for the data objects with short absolute validity intervals could incur more deadline misses in updating stale data on-demand. In DRACON, further load control is achieved by migrating replicas to underutilized nodes of the same cluster.

5.3.4 Scalability

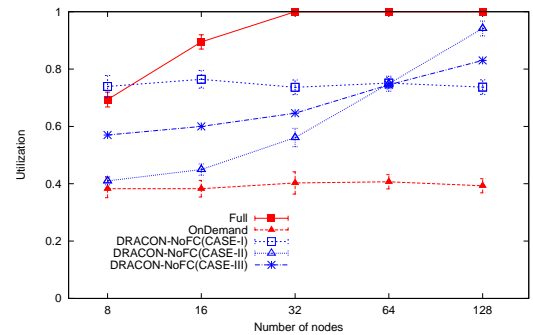


Figure 11. Per node overhead.

Our final set of simulations evaluate the scalability of DRACON. We increase the number of nodes in the system exponentially. Since the total number of nodes is the product of the number of clusters and the number of member nodes per cluster, the system size can be varied in two ways:

- **CASE-I:** We keep the number of clusters fixed at 8 and then increase the number of member nodes per cluster from 1 to 16.
- **CASE-II:** We keep the number of member nodes per cluster fixed at 8 and then increase the number of clusters from 1 to 16.

We apply 70 user transactions/second per node, and the feedback controllers are turned off to observe the uncontrolled load changes as the number of nodes increases. Figure 11 shows the CPU utilization as the number of nodes increases. We can see that the replication overhead in DRACON is not proportional to the number of nodes but to the number of clusters; the overhead remains almost constant when the number of clusters is fixed. **CASE-I** and **CASE-II** represent two extreme cases, the best case and the worst case, respectively, in terms of the scalability. In reality, the overhead of DRACON lies between the two as **CASE-III**, in which the overhead increases

rather slowly. As shown in Section 4.2, the size of each cluster determines its intra-cluster communication delay. Therefore, the number of clusters should be chosen to strike a good balance between the replication overhead and the data propagation delay.

However, the proportional increase of overhead to the number of clusters can be a problem in extremely large scale systems. When a larger number of clusters is required to guarantee a certain intra-cluster communication delay bound, we may need a hierarchical approach that groups a set of clusters, which have close interactions, into a higher level of cluster. We reserve this as our future work.

6 Related Works

Traditional replicated relational database systems focus on the problem of guaranteeing strong consistency to replicated data. Although strong consistency provides convenient programming model, these systems are limited in scalability and availability [5]. Moreover, they do not support timeliness of transactions [11][3].

Distributed real-time databases (DRTDBs) have drawn research attention in recent years [13][19]. Instead of providing strong logical consistency, DRTDBs focus on data freshness and timeliness of transactions. However, most previous DRTDB work target small-scale systems. DRACON is based on these previous DRTDB work. However, we extend it to large-scale systems in wide-area network environments.

Several distributed feedback control schemes [10][18][15] have been proposed for DRE systems to provide QoS in unpredictable operating environments. However, these approaches are not directly applicable to DRTDBs, because they do not consider DRTDB-specific issues such as the data freshness. Moreover, these approaches consider only the unpredictability of computation load, ignoring communication latencies. DRACON considers both of them for robust QoS guarantees.

7 Conclusions and Future Work

DRACON has been designed to provide a highly scalable data service with QoS guarantees in large-scale DRE 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. Using an extensive simulation study, DRACON is shown to achieve a significant performance improvement compared to several baseline algorithms in guaranteeing the desired deadline miss ratio in large-scale and wide-area network environments.

We plan to extend this work in several ways. One direction is to extend the cluster formation algorithm of DRACON so that clusters can be post-adjusted while the system is running as opposed to the current static approach. Another direction is to support more hierarchies for higher scalability. The interaction of controllers at multiple hierarchies poses an interesting research question that we plan to investigate.

References

- [1] T. Anderson, L. Peterson, S. Shenker, and J. Turner. Overcoming the internet impasse through virtualization. *IEEE Computers*, 38(4):34–41, 2005.
- [2] K. P. Birman, J. Chen, E. M. Hopkinson, R. J. Thomas, J. S. Thorp, R. V. Renesse, and W. Vogels. Overcoming communications challenges in software for monitoring and controlling power systems. In *Proceedings of the IEEE*, 2005.
- [3] P. F. et al. Perdis: Design, implementation, and use of a persistent distributed store. In *Advances in Distributed Systems, Advanced Distributed Computing: From Algorithms to Systems*. Springer-Verlag, 1999.
- [4] T. Fusheng and L. Jie. Online security analysis based on database method for electrical power system. In *TENCON'93*, 1993.
- [5] J. Gray, P. Helland, P. O'Neil, and D. Shasha. The dangers of replication and a solution. In *SIGMOD '96*, 1996.
- [6] C. H. Hauser, D. E. Bakken, and A. Bose. A failure to communicate: next generation communication requirements, technologies, and architecture for the electric power grid. *Power and Energy Magazine, IEEE*, 3(2):47–55, 2005.
- [7] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury. *Feedback Control of Computing Systems*. Wiley IEEE press, 2004.
- [8] K.-D. Kang, S. H. Son, and J. A. Stankovic. Managing deadline miss ratio and sensor data freshness in real-time databases. *IEEE Transactions on Knowledge and Data Engineering*, 16(10):1200–1216, October 2004.
- [9] W. Kang, S. H. Son, J. A. Stankovic, and M. Amirjoo. I/O-aware deadline miss ratio management in real-time embedded databases. In *The 28th IEEE Real-Time Systems Symposium (RTSS)*, Dec, 2007.
- [10] S. Lin and G. Manimaran. Double-loop feedback-based scheduling approach for distributed real-time systems. In *Conference on High Performance Computing (HiPC)*, 2003.
- [11] B. Liskov, M. Castro, L. Shrira, and A. Adya. Providing persistent objects in distributed systems. In *ECOOP '99: Proceedings of the 13th European Conference on Object-Oriented Programming*, 1999.
- [12] J. Nils R. Sandell, P. Varaiya, M. Athans, and M. G. Safonov. Survey of decentralized control methods for large scale systems. *IEEE Transactions on Automatic Control*, 23(2):108–128, 1978.
- [13] P. Peddi and L. C. DiPippo. A replication strategy for distributed real-time object-oriented databases. In *Symposium on Object-Oriented Real-Time Distributed Computing*, pages 129–136, 2002.
- [14] W. Rutz. Critical Issues Affecting Power System Control Center Databases. *IEEE Transactions on Power Systems*, 11(2):923–928, 1996.
- [15] J. A. Stankovic, T. He, T. Abdelzaher, M. Marley, G. Tao, S. Son, and C. Lu. Feedback control scheduling in distributed real-time systems. In *RTSS '01: Proceedings of the 22nd IEEE Real-Time Systems Symposium (RTSS'01)*, page 59, Washington, DC, USA, 2001. IEEE Computer Society.
- [16] J. S. Thorp, C. E. Seyler, and A. G. Phadke. Electromechanical wave propagation in large electric power systems. *IEEE Transactions on Circuits and Systems*, 45(6):614–622, 1998.
- [17] K. Tomsovic, D. E. Bakken, V. Venkatasubramanian, and A. Bose. Designing the next generation of real-time control, communication, and computations for large power systems. *Proceedings of the IEEE (Special Issue on Energy Infrastructure Systems)*, 93(5), 2005.

- [18] X. Wang, D. Jia, C. Lu, and X. Koutsoukos. DEU-CON:Decentralized End-to-End Utilization Control for Distributed Real-Time Systems. *IEEE Transactions on Parallel and Distributed Systems*, 18(7):996–1009, 2007.
- [19] Y. Wei, S. H. Son, J. A. Stankovic, and K. D. Kang. Qos management in replicated real time databases. In *RTSS '03: Proceedings of the 24th IEEE International Real-Time Systems Symposium*, 2003.
- [20] J. Wu, Y. Cheng, and S. N. N. Overview of real-time database management system design for power system scada system. In *Proceedings of the IEEE SoutheastCon*, 2006.
- [21] W. Zhang and J. He. Modeling End-to-End Delay Using Pareto Distribution. In *Second International Conference on Internet Monitoring and Protection (ICIMP'07)*, 2007.