Science and Technology, 8 (2003) 33-46
© 2003 Sultan Qaboos University

Performance Evaluation of Database Systems Using Colored Petri Nets

A.B. Mnaouer, K. Day, M. Al-Towaiq and F.A. Masoud

Department of Computer Science, College of Science, Sultan Qaboos University, P.O.Box 36,Al Khod 123, Muscat, Sultanate of Oman, Email: fawaz@squ.edu.om

تقويم الأداء لنظم قواعد البيانات باستخدام شبكات بتري الملونة

عادل بن منور، خالد داي، محمد الطويق، و فوّاز أحمد مسعود

خلاصة: في هذه الورقة قام الباحثون بتصميم نماذج لنظم قواعد البيانات المكررة والمركزية وذلك باستخدام شبكات بتري الملونة. وقد تم إجراء دراسة مستفيضة للمقارنة بين أداء هذه النماذج. إنّ النماذج التي تم تصميمها قد حققت الغرض المطلوب من الدراسة من حيث تغطية المتغيرات الديناميكية لنظم قواعد البيانات التي خضعت للدراسة. و قد تم تدقيق صحة التصاميم المقترحة بإجراء عدة تجارب محاكاة عملية والتي تمّ إخضاعها لضوابط متغيرة منها سرعة المعالج وسرعة شبكة ألحواسيب والنسبة بين عمليتي القراءة والكتابة للأنظمة. وقد أعطت نتائج المحاكاة مدلولا يشير إلى أنّ نظم قواعد البيانات المكررة تتمتع بكفاءة أداء أعلى منها في حال نظم قواعد البيانات المركزية وذلك نسبة إلي زمن الرد على الاستعلامات المرسلة وأيضا في حالة التشبع التي لا يستطيع فيها النظام الاستجابة لأية زيادة في المدخلات. ولكن تم رصد تحسن نسبي في أداء نظم قواعد البيانات المركزية عندما تكون سرعة المعالج والشبكات كبيرة جدا بالمقارنة مع نظم قواعد البيانات المكررة.

ABSTRACT: We propose Colored Petri Net (CPN) models for replicated and centralized database systems and conduct a comparative study of their performance. The designed CPN models capture the dynamics of the studied database systems and estimate their expected performance with an appropriate level of abstraction. A number of simulation experiments were performed under various load conditions of varying parameters such as server speed, network speed, and read/write ratio. The simulation results show that under similar operation conditions, replicated systems exhibit higher performance than centralized systems in terms of query response time and system saturation levels. However, centralized systems become more competitive when their network and server speeds are much higher than those of the replicated systems.

KEYWORDS: Client-Server, Centralized Databases, Colored Petri Net, Performance Evaluation, Replicated Databases.

1. Introduction

The traditional setting of database systems is based on the centralized client-server architecture. In such systems, a central server handles requests coming from a number of remote clients. A more recent development consists of deploying multiple service points keeping a copy of the data at each service point (replica). Database queries are submitted to replicas independently. The service replicas cooperate in servicing the queries and maintaining data consistency. Database queries are either read queries processed locally or write queries propagated to all replicas.

The replicated setting can be a viable alternative for many reasons. Firstly, it can offer a distributed base for data that can be scaled up and down to meet varying needs and demands. Secondly, it increases data availability, system robustness and fault tolerance. Finally, it allows deployment of clusters of workstation that offer aggregate computing power and storage capacity.

On the negative side, replicated database systems may impose substantial communication overhead especially with a high ratio of write queries resulting in poor query response time. The aim of this study is to investigate the conditions under which replication is a more attractive solution than centralization in terms of performance.

The type of consistency control method used affects the performance of the replicated system. Two broad models of consistency control are known in the literature. The first one is the asynchronous model (also known as the *lazy update model*) where changes introduced by a transaction are propagated only after the transaction has been committed (Wiesmann *et al.,* 2000). The second one is the synchronous model (also known as the *eager model*) where a transaction should not commit until full synchronization with all replicas is completed (Wiesmann *et al.,* 2000).

It is argued that synchronous models are hardly feasible in practice due to problems related to synchronization delays, deadlock avoidance, and scalability (Gray *et al.,* 1996). However, it is asserted in Wiesmann *et al.,* (2000) that group communication primitives may be a solution for building good performance synchronous replication models (Kemme and Alonso, 2000; Kemme and Alonso, 1998; Pedone *et al.,* 1997; Pedone *et al.,* 1998). A number of consistency control methods have also been studied in the literature (Beeri *et al.,* 1989; Guerraoui and Schiper, 1997; Wiesmann *et al.,* 1999; Day *et al.,* 2001).

In this study we assume asynchronous replication with minimum communication overhead with only one update message broadcasted to all replicas for each write query. This assumes a fault-free system in order for the updates to be delivered to all replicas. Modeling fault-tolerant consistency control protocols is beyond the scope of this paper.

The targeted performance evaluation is conducted using Colored Petri Nets (CPN). CPNs represent powerful means for modeling and prototyping complex, distributed, and parallel systems (Jensen 1992, 1994). CPNs were used successfully for modeling various systems such as VLSI chip design (Shapiro 1991), communication protocols (Mnaouer *et al.,* 1999; Mnaouer *et al.,* 2002; Morera and Gonzalez, 1999), and algorithm analysis (Jorgensen and Kristensen, 1999).

We propose two CPN models for centralized and replicated database systems. The CPN models are used for simulating the two systems and comparing their query response times and system saturation levels as functions of the load under various conditions of network latency, query service time, and query read/write ratio.

The paper is organized as follows. In the next section, a description of the evaluated centralized and replicated system models is given. Section 3 gives an informal overview of Colored Petri Nets followed by a detailed description of the designed CPN models used in the simulation. The simulation results are presented and discussed in section 4. Concluding remarks on the work are provided in section 5.

2. Overview of the centralized and replicated database system models

In a typical centralized database system, a database server is connected to a number of clients via a switched network. Database queries sent from clients to the server are subject to two communication delay components: a client-switch (τ_{cs}) delay and a switch-server delay (τ_{ss}). In real situations, the switch-server link can be 1, 10, or even 100 times faster than the client-switch link.

In a fully replicated client-server database system, each site runs a replica of the database server, hence acting as a server as well as a client. That is the case when all the sites are included in the replication scheme. Read database queries are processed locally while write queries are processed both locally and remotely at all replicas. Propagation of write queries is performed via a switched network. Figure 1 shows the components of the centralized and replicated database systems.

In Figure 1.a the arrival of transactions to the server, in the centralized system, is assumed to form a Poisson process of average arrival rate $N\lambda$ transactions per second (where N represents the number of clients involved in the system). The service time for a transaction at the server is assumed to be a random variable corresponding to an exponential distribution of the service time with average service rate of μ_c transactions per second. Let τ_{cs} be the average communication delay between a client and the switch and τ_{ss} the average communication delay between the switch and the server. The message switching time at the switch is assumed negligible.

Figure 1.b shows the components of the replicated database system. We do not consider any particular replication control method; however we assume asynchronous replication with minimum inter-replica communication with only one update message broadcasted to all the replicas for every write transaction request. The assumption is realistic in the sense that many commercial database replication systems opt for the asynchronous replication model, and therefore, allow transient inconsistent states of the database system to exist (Kemme and Alonso, 2000). We wanted also to analyze the performance of replicated versus centralized database systems subject to the variation of server speed, network speed, and read/write ratio parameters while isolating the performance issues related to the application of more complex consistency control mechanisms.

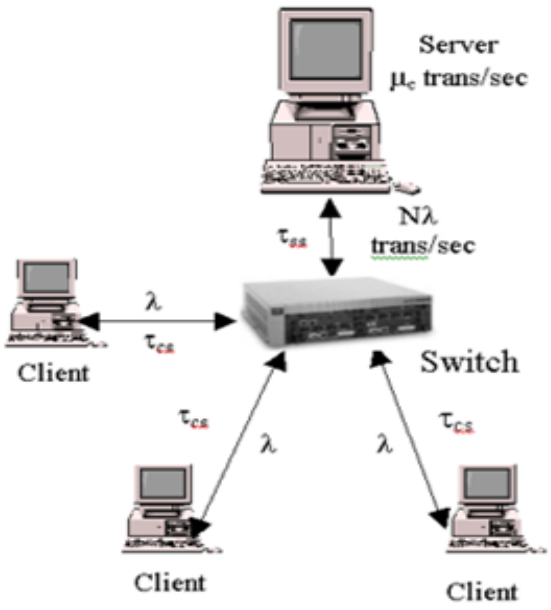


Figure 1a. The centralized model

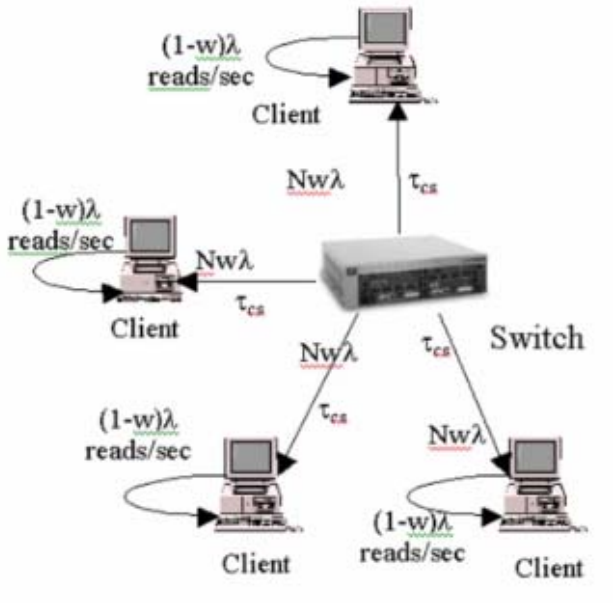


Figure 1b. The replicated model

In this system each of the N processors (replica) is holding a copy of the database and serves both as a client and as a server. An average of λ transactions per second are assumed to be generated at each replica. A portion $w\lambda$ of this rate corresponds to write transactions that have to be broadcasted to all processors. The remaining $(1\text{-}w)\lambda$ rate corresponds to read transactions that can be processed locally. Since each write transaction is propagated to all processors, the total arrival rate of write transactions to each processor is $Nw\lambda$. The overall arrival of transactions to each replica (read or write), is assumed to follow a Poisson process. The service time for a transaction at each processor is assumed exponentially distributed with average service rate of μ_r transactions per second. The average replica-to-switch communication delay is assumed the same as τ_{cs} of the centralized model.

3. Colored petri nets

We start by presenting an informal overview of CPNs before describing the details of the proposed CPN models for the centralized and replicated database systems.

3.1 Overview of colored petri nets

A Petri net is a network of interconnected locations and activities, with rules that determine when an activity can occur, and specify how its occurrence changes the states of the associated locations. Petri Nets can be used to model systems of any type. They are particularly useful in facilitating the design and analysis of complex distributed systems that handle discrete flows of objects and information (Jensen 1992, 1994).

CPNs represent an extension of Petri Nets. They are graphical models that use the concept of colored tokens to represent data structures and state conditions. The presence of data or state conditions is marked by *colored tokens* in locations. Each token has an associated *token color* that specifies the type of data it is representing (usually representing arbitrary complex data values).

The locations are represented graphically by ellipses and called *places*. Each place is associated with a *color set* that specifies the type of tokens (i.e., data) that may reside in the location. Activities are represented by rectangles called *transitions*, which govern the occurrence of events in the system. *Places* can be either input or output *places* for a *transition*. *Places* and *transitions* are linked through directed *arcs* modeling the flow of data. Each arc has an associated *arc expression* that controls the *transition's occurrence*. This expression specifies the number of tokens consumed by the *transition*, or the number of tokens that is produced after its occurrence.

When the number of tokens in each input place of a transition satisfies the corresponding arc expression, then the transition is said to be *enabled*. An enabled transition can fire (i.e., occur) at any time. When it *fires*, it consumes as many tokens from its input places, as specified by their corresponding arc expressions, and produces as many tokens in its output places as specified by their corresponding arc expressions. Additional conditions for the enabling of the transition can be specified through the *guard* of the transition. All the Boolean conditions specified in the guard must evaluate to true for the transition to be enabled.

CPN modeling and simulation is supported by various simulation packages such as the Design/CPN tool (Jensen *et al.,* 1996) used in this study. In this tool, the different parts of a CPN model are constructed in different CPN pages. This helps making use of the CPN *hierarchy* constructs that enable the designer to breakdown the complexity of the modeled system into different layers with different abstraction levels. In the Design/CPN there is a provision for using a *declaration node* that can be used to record the color sets, constants, variables, and function definitions. Figure 2 depicts a small CPN diagram used for processing shipping orders.

The transition *Process Orders* has one input place, *Order In*, and one output place *Order Out*. The token color *Order* is associated with the place *Order In*, and the equivalent token color *ProductShipped* is associated with the place *Order Out*. The color set *Order* is declared to hold *Big* and *Small* as data values (see declaration node). A variable *Ordent* is declared in the declaration node.

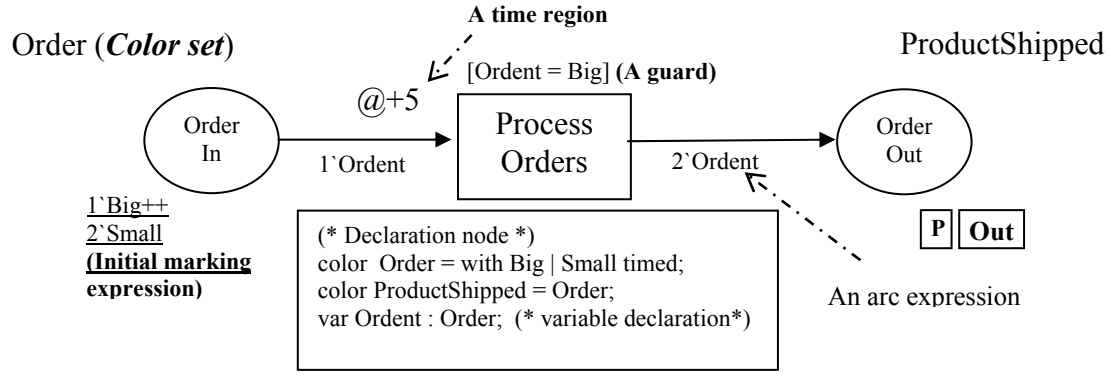


Figure 2. Components of a CPN[1]

The guard of the transition specifies that *Ordent* should be bound only to tokens having the value *Big*. The binding means that the variable *Ordent* of the arc expression from the place *Order In* is substituted by any token that is part of the current marking of the place subject to the constraint of the transition guard. An arc expression is evaluated against the current marking of (i.e., current distribution of tokens on) the input places of the transition. In the displayed (initial marking) state the transition may be *enabled* if the CPN is simulated, since at the start of the

[1] The figure is taken and edited from a tutorial of Design/CPN by Meta Software Corporation

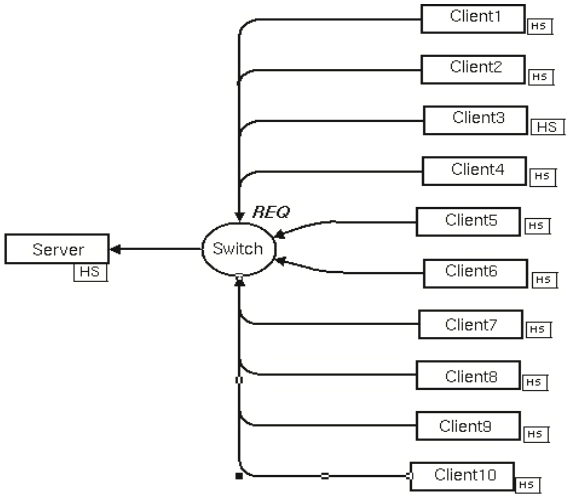
simulation the initial marking of places becomes a current marking for the CPN. When the transition occurs it will consume the token *Big* (of which only one instance exists in *Order In*) and produces two instances of *Big* into the output place.

Place fusion is one hierarchy construct that enables a place to be present physically on different CPN pages while representing a single conceptual place. All the physical places, in this case, are known to belong to the same *fusion set*. There can be two types of places that may belong to the same *fusion set* that helps construct hierarchies. *Socket places* and *port places*. *Socket places* are found one hierarchy level and they are mapped to *port places* belonging to a CPN described in a lower level of the hierarchy. Thus, this mapping of *socket places* and *port places* ensures the connection between CPNs constructed at different hierarchical levels. The place *Order Out* is declared as a *socket place* that will be used to connect to another CPN page in which a more complete shipping process with additional operation is described. *Substitution transition (ST)* is the second hierarchy construct that enables to hide lower level design details into the lower abstraction level. The *ST construct* allows the designer to relate a transition (and its surrounding arcs) in an upper level to a more complex activity (modeled by a detailed CPN) hidden in a lower level of the hierarchy. The transition *Process Order* can be designed to represent a *ST* that hides more elaborate details of product shipping that is hidden in the current level of the model.

Finally, the concept of *time stamps* that may be associated with *timed* tokens is used for the purpose of performance evaluation. Each timed token will bear an associated time stamp at each creation in accordance to a *global clock* maintained by the simulator. In Figure 2, the transition *Process Order* has an associated *time region* (denoted by @+5) that specifies that the occurrence of the transition takes 5 time units. The associated timestamps of the produced *Big* tokens will be augmented with 5 units. See Jensen (1992, 1994, 1996) for more details.

3.2 CPN models for the database systems

A top down approach was adopted in the modeling. A top page that models the overall CPN layout is constructed. In this top page, a single *ST* transition represents the server, and a set of *ST* transitions represent the different clients connected to the server. Figure 3a and Figure 3b depict the top page for the centralized model and the replicated model respectively.



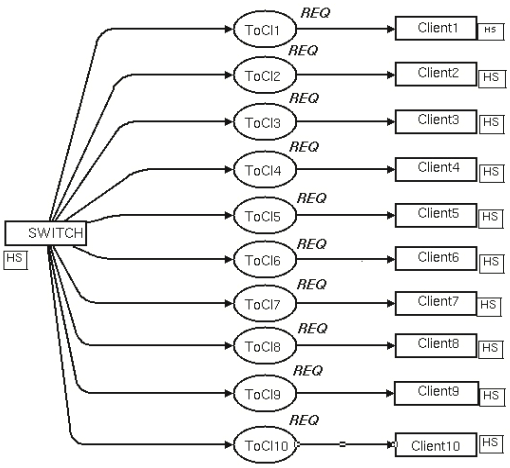


Figure 3a. Centralized model CPN top page. **Figure 3b.** Replicated model CPN top page.

Notice that in Figure 3a, the communication goes from the clients to the server through the switch node. Clients usually get replies to their queries back from the server. However, to simplify the model, computation of the delay is done at the server considering a round trip delay. In Figure 3b, the communication goes from switch to clients. This abstracts the fact that clients broadcast their write queries to all the other clients to maintain consistency.

Figure 4 shows the declaration node used for both models where color sets, variables, function declarations and definitions are given.

Figure 5 shows the CPN page modeling a generic client operation in the centralized model. Transition *Gen_Req* represents query request generation. The place *GEN* is used as a time trigger that initiates query generation. Its token color is *TR*, representing a timed token color. A token *t* is generated after each firing of *Gen_Req* producing a time stamp calculated using the texp(1.0/iar) function that generates a random variable exponentially distributed with mean *iar* (i.e., the interarrival time). The generated token is specified by the expression of the arc connecting the transition *Gen_Req* to the place *Generated.*

```
(* Declaration Node *)

(* Constants *)
val tcs = 10;  (* communication delay from client to switch for a speed of 100 Mbps *)
               (* delay needed for transmission of 1 KB of data query *)
               (* Average query size is set to 100 KB *)

val tss = 1;       (* communication delay from switch to server for a speed of 100 Mbps *)

val w=0.2;         (* percentage of write request *)
val N=10;          (* number of nodes *)
val kbpt = 1;      (* One Kilobyte processing time : centralized *)
val iar =1667.0; (* Interarrival time between two read requests /client *)
val iaw =2500.0; (* Interarrival time between two Write requests *)
val M=100;         (* The mean query size in KBytes *)

(* Color Sets *)
color I = int;
color B = with T|F;
color E = with e;
color CL = with c;
color SER =  with s;
color TR = with t timed;
color REQ = record rid :I * GT : I * sz:I  declare mult timed;  (* Size of the request *)
color r  = int with 10..100;      (* The starting time for the first packet generation *)

(* variables *)
var i, i1, i2, i3: I;
var t1,t2: TR;
var q, q1, q2: REQ;

(* Functions *)
fun IntTime() = IntInf.toInt(time());  (* A Function that get the time *)
fun round x = floor(x+0.5); (* A Function that takes the smallest integer of x + 0.5 *)
fun texp y = round(exponential(y)); (* A Function that find the exponential of y *)
fun run() = ran'r();(* A Function that select a random value in the interval r *)

(* End *)
```

Figure 4: The Declaration node used for both models

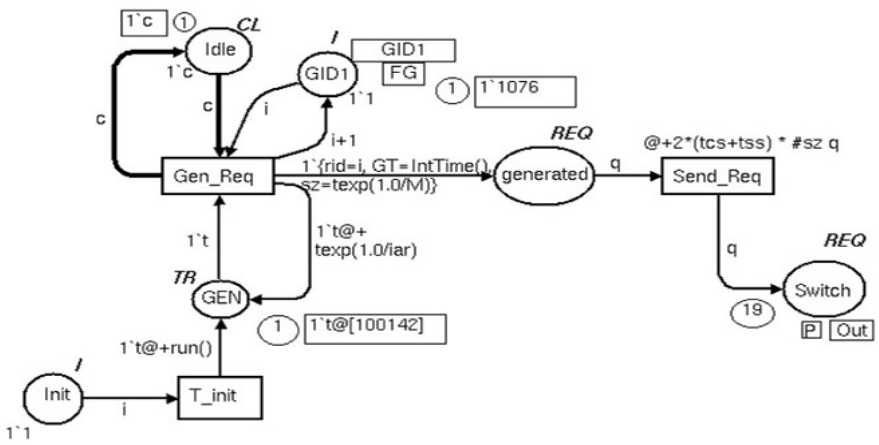


Figure 5. A CPN page modeling a generic client operation in the centralized model.

Notice that the size of the query (field *sz*) is also exponentially distributed with mean M (M is set to 100 KB considering big size transactions). The transition *Send_Req* is used to send the queries to the *Switch* place. It is at this transition that the transmission delays are applied (considering both directions) in the time stamp denoted by the mark @. The transmission depends on the query size.

The place Switch is a fusion place that links to the CPN page modeling the server depicted in Figure 6.

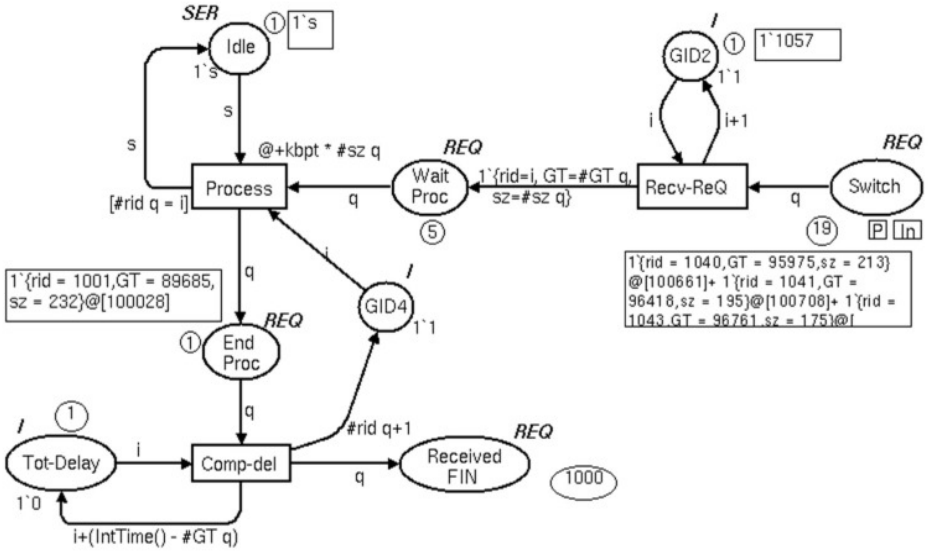


Figure 6. A CPN page modeling the server operation in the centralized model.

In Figure 6, the place *Idle* and the transition *Process* represent the availability of the server. The transition *Recv-ReQ* is used to model reception of queries from clients that are treated according to a FIFO policy, enforced as follows: any incoming query is assigned a request Id (i.e., field *rid*) extracted from the token on the place *GID2*. The FIFO enforcement is done at the transition *Process* using its guard (i.e., [#rid q = i]).

The place *GID4* is acting as a semaphore with its single token consumed through the variable *i* by the transition *Process* when the processing of a new query is started. A new token pointing to the next query in sequence is produced by the transition *Comp-del* when the processing and delay computation of the previous query is finished. Then, the next query is processed. The transition *Comp-del* and the two places *Tot-Delay* and *Received-FIN* are used for computing the accumulated two-way response delay. A snap-shot of the simulation is shown in the Figure displaying token values.

Figure 7 describes the CPN model of the switch in the replicated model. Read queries are processed locally by clients. Write requests are propagated through the switch to all other nodes. In order to simplify the model, the write queries are issued by the switch and sent to all nodes.

The place *Idle* and the transition *ReQ-Gen* represent the state where the switch is available. The transition *ReQ-Gen* generates requests according to a Poisson process regulated using the place *GEN* with a timed token *t.* After each firing of *ReQ-Gen* a new request is generated into the place *WR-Req* that includes a request identifier (i.e., field *rid*), a generation time (i.e., field *GT* loaded with current time) and a query size. The transition produces also a token *t* with a time stamp computed using the function *texp(10.0/iaw)*. The inter-arrival time of write requests is denoted *iaw*, and 10.0/iaw represents the arrival rate from 10 different clients.

The transition *Broadcast* sends the write query requests to all clients through the places *ToCl1* through *ToCl10* (connecting to other CPN pages modeling clients). This allows clients to update their replicas. The delay component applied to the transition *Broadcast* represents twice the time the client switch communication delay (i.e., tcs * #sz q).

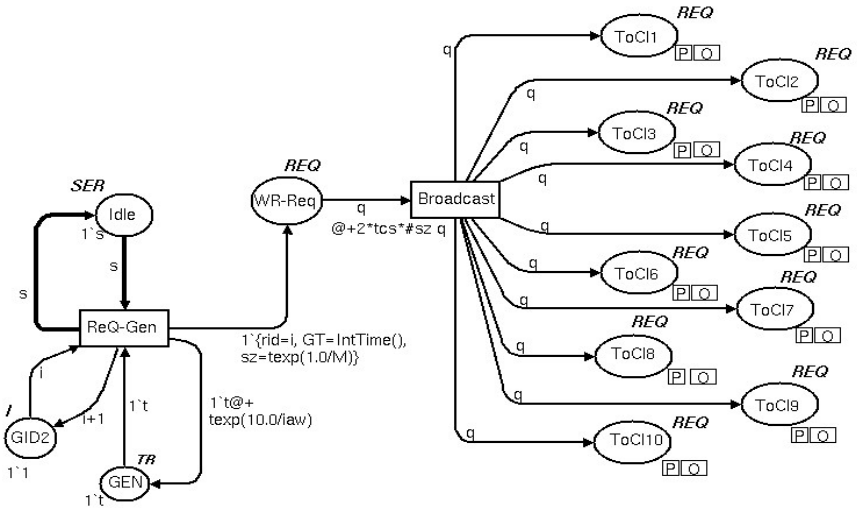


Figure 7. A CPN page modeling the switch operation in the replicated model.

Figures 8 and 9 represent the CPN page modeling the clients in the replicated model (part I and part II). In Figure 8, write requests are received through the port place *ToCl1*. The transitions *Merge 1* and *Merge 2*, are modeling the queuing of local read requests and external write requests into the same queue. The transition *Gen_ReR* represents the generation of local read requests. The place *GID1* serves as a semaphore, used for sequencing requests into the *Queue* place. The transition Process is used to process requests according to the order of their arrival.

The second part of the client model is depicted in Figure 9. Firing the transition *Process* produces a query token in the place *ToCl1_2*, and captures the synchronization token on the place *Cnt*. In addition, the transition *Process* has an associated timestamp calculated based on the query size. When the timestamp associated with the current token (i.e., query) expires, the token is consumed by the transition *Recv_WrR,* modeling the end of processing. The transition *Recv_WrR,* is used to compute the delay incurred by the query (adding up the difference between generation time and arrival time) that is added to the token residing in the place *Acc-Delay*. Firing the transition *Recv_WrR* releases the synchronization token into the place *Cnt* with a value equal to the number of the next query. The places *Acc Delay* and *Recved Req* belong to two global fusion sets present in all client models and representing one logical place each.

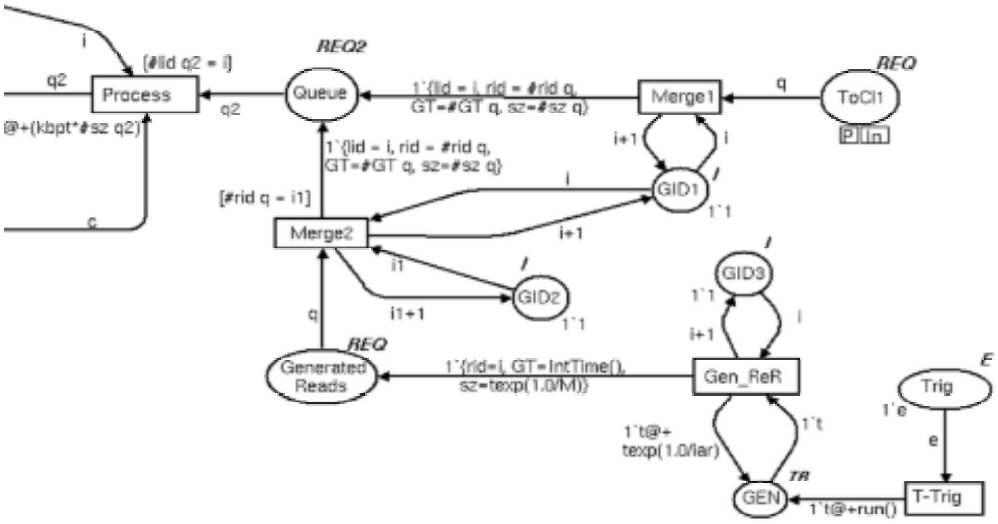


Figure 8. A CPN page modeling the client of the replicated model (part I)

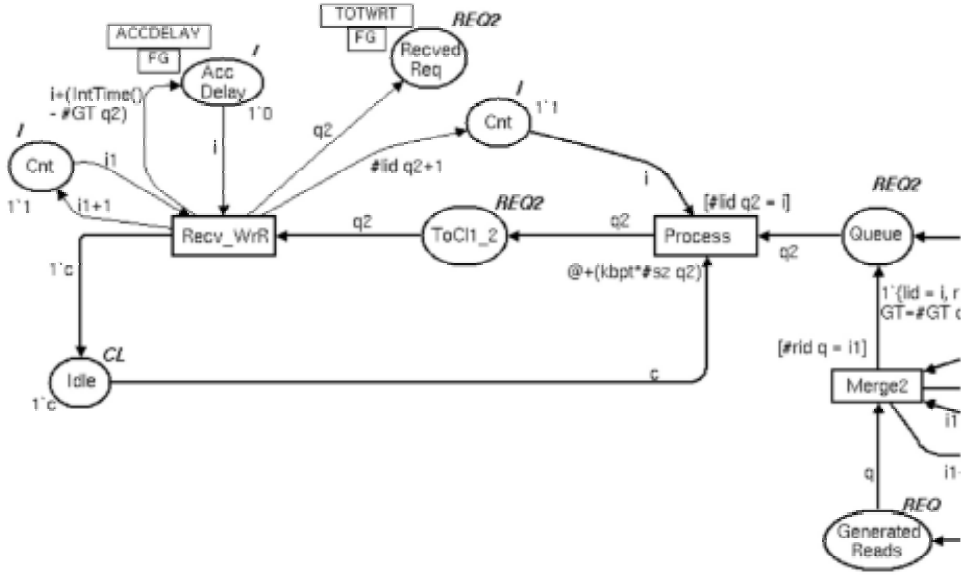


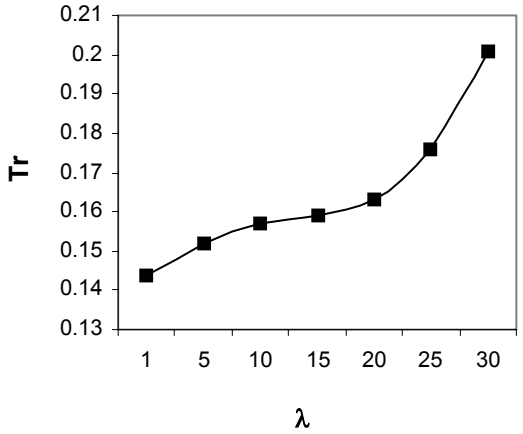
Figure 9. A CPN page modeling the client of the replicated model (part II)

4. Simulation results

A set of simulation experiments were conducted. Measures of query response time as a function of query arrival rate are obtained. In the first set we have varied the processing speed of the central server (μ_c). In the second set, different network connection speeds (τ_{ss}) between the server and the switch are studied. In the third set, we have investigated the effect of the read-write ratio on the performance of the replicated system. Along with the plots standard deviations of the means and 95% confidence intervals were computed for three selected points for each plot.

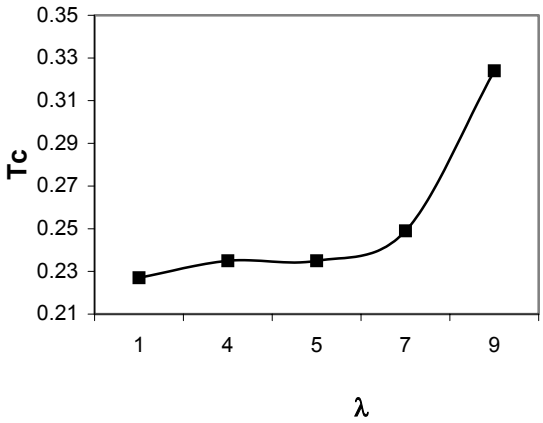
4.1 Simulation set I

In this first experiment we plot the query response time in the centralized system (T_c) and in the replicated system (T_r) as functions of the query arrival rate (λ) for different server speeds μ_c. We use the following assumptions for this experiment:
τ_{cs} = 0.1 sec, τ_{ss} = 0.01 sec, μ_r = 10 queries/sec, w = 0.2, N = 10 (# of clients).



Ref.Value	5	15	25
Mean	0.152	0.1594	0.1763
Std.deviation	0.003	0.003	0.0037
95 % Conf. Interval of mean	0.1493 - 0.1541	0.1572 - 0.1617	0.1734 - 0.1793

Figure 10.1. T_r vs load (replicated).

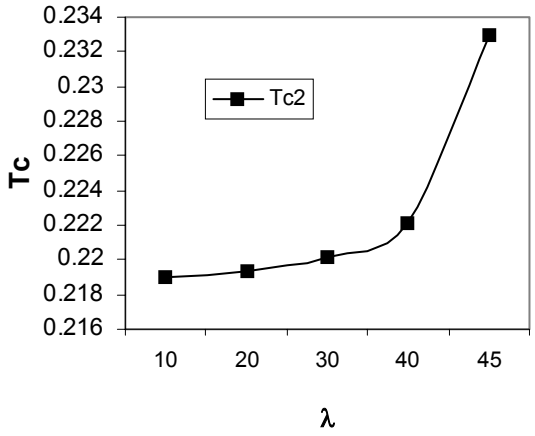


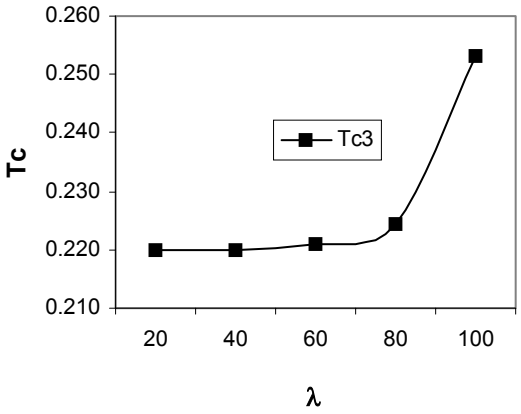
Ref.Value	4	7	9
Mean	0.2348	0.2489	0.3239
Std.deviation	0.0028	0.00092	0.0058
95 % Conf. Interval of mean	0.2325 - 0.2371	0.2482 - 0.2495	0.3198 - 0.3279

Figure 10.2. T_c vs. load (centralized : $\mu_c = \mu_r$).

The query response time T_r of the replicated model (Figure 10.1), is lower than the query response time T_c of the centralized model for the three considered server speeds
($\mu_c = \mu_r$, $\mu_c = 5\ \mu_r$, $\mu_c = 10\ \mu_r$).

$\tau_{cs} = 0.1$ sec, $\tau_{ss} = 0.01$ sec, $\mu_r = 10$ queries/sec, $w = 0.2$, $N = 10$ (# of clients).





Ref. Value	20	30	40
Mean	0.2214	0.2243	0.229
Std. deviation	0.0019	0.0007	0.0008
95 % Conf. Interval of mean	0.2200 - 0.2228	0.2238 - 0.2248	0.2282 - 0.2295

Figure 10.3. T_c vs. load (cent.: $\mu_c = 5\mu_r$).

Ref. Value	40	60	80
Mean	0.22	0.221	0.223
Std. deviation	0.0004	0.0006	0.0019
95 % Conf. Interval of Mean	0.2199 - 0.2206	0.2204 - 0.2213	0.2211 - 0.2245

Figure 10.4. T_c vs. load (cent: $\mu_c = 10\mu_r$).

The load saturation level (λ at which T becomes very large due to system congestion) of the replicated model is substantially higher than that of the centralized model in the first case ($\mu_c = \mu_r$) and are comparable when μ_c is set to five times μ_r. Only when μ_c was set to ten times μ_r (Figure 10.4) the load saturation level of the centralized model has become substantially higher than that of the replicated model. We conclude from this experiment that the replicated model performs better than the centralized model unless a very powerful server is used in the centralized model (such as ten times, or more, faster than a replica).

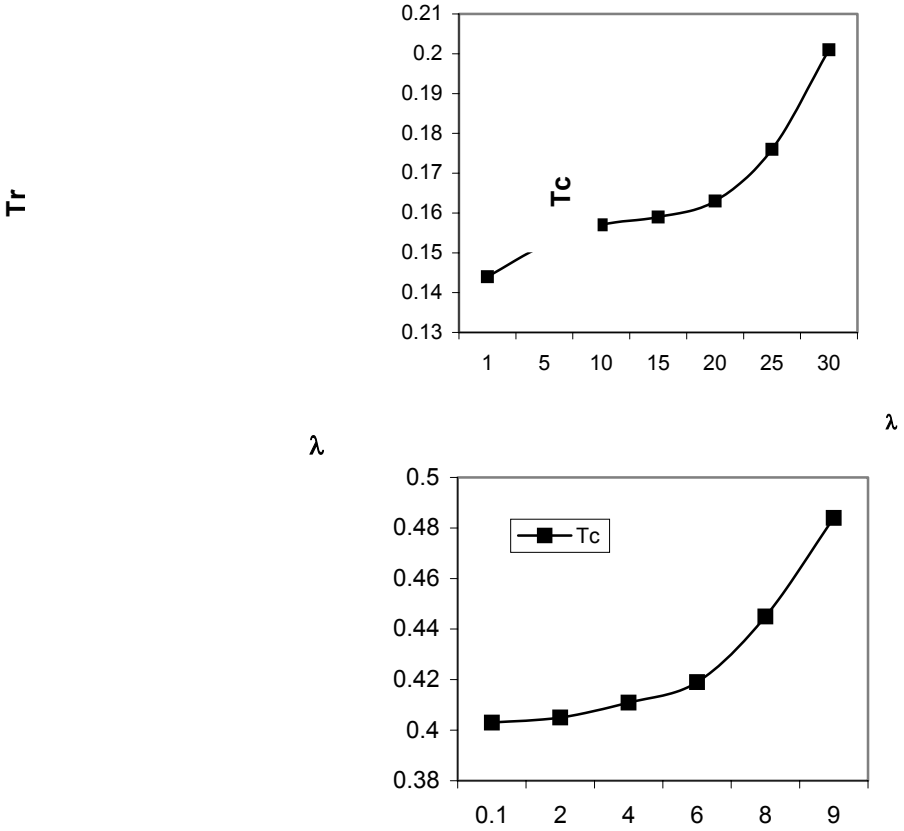
4.2 Simulation set II

In this second experiment, we plot T_c and T_r as functions of λ for different server-switch transmission delays τ_{ss}. We use the following assumptions for this experiment:
$\tau_{cs} = 0.1$ sec, $\mu_c = 100$ queries/sec, $\mu_r = 100$ queries/sec, $w = 0.2$, $N = 10$

We plot T_c and T_r as functions of λ for different server-switch transmission delays τ_{ss}. The client-switch transmission delay is computed based on a 10Mbps client-switch connection. The server-switch connection bandwidth will be tested with the values 10Mbps, 100Mbps, and 1Gbps.

Since we focus here on studying the impact of the communication speeds, we fix the centralized server and all replicas' processing speeds at 100 queries per second.

The response time T_r of the replicated model (Figure 11.1) is lower than the response time T_c of the centralized model for various server-switch transmission delays τ_{ss}. In all cases the load saturation level of the replicated model is almost three times higher than that of the centralized model. It can be seen in Figures 11.2, 11.3, and 11.4 that the communication delay has little effect on the centralized system since each query (or query reply) is transmitted only once. The effect of the communication delay is however greater on the replicated system since write queries are broadcasted to all replicas, especially for high ratios of write queries (w). The effect of w is studied in the next simulation set.



Ref. Value	5	15	25
Mean	0.152	0.1594	0.1763
Std. deviation	0.003	0.003	0.0037
95 % Conf. Interval of mean	0.1493 - 0.1541	0.1572 - 0.1617	0.1734 - 0.1793

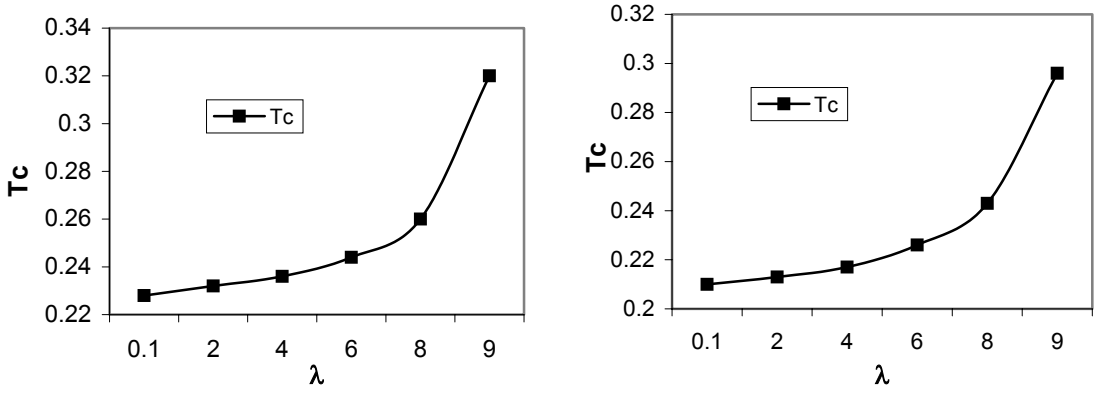
Figure 11.1. T_r vs load (replicated).

Ref. Value	40	60	80
Mean	0.414	0.419	0.445
Std. deviation	0.0034	0.0025	0.004
95 % Conf. Interval of mean	0.4112 - 0.4162	0.4174 - 0.4208	0.4420 - 0.4473

Figure 11.2. T_c vs. load (centralized : $\tau_{ss} = \tau_{cs}$).

4.3 Simulation set III

In this last experiment T_c and T_r are plotted as functions of λ for different values of the ratio (w) of write queries. The following is assumed for this experiment: $\tau_{cs} = 0.01$ sec, $\tau_{ss} = 0.001$ sec, $\mu_c = 100$ queries/sec, $\mu_r = 10$ queries/sec, $N = 10$.



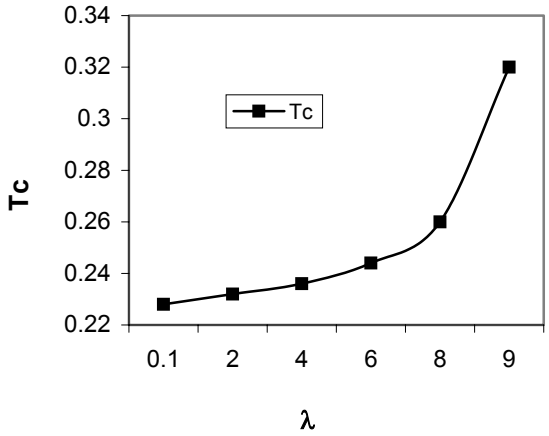
Ref. Value	4	6	8
Mean	0.236	0.244	0.26
Std. deviation	0.0006	0.0029	0.0011
95 % Conf. Interval of mean	0.2357 - 0.2365	0.2417 - 0.2456	0.2593 - 0.2610

Ref. Value	4	6	8
Mean	0.217	0.226	0.243
Std. deviation	0.0009	0.0005	0.0016
95 % Conf. Interval of mean	0.2161 - 0.2174	0.2260 - 0.2267	0.2414 - 0.2436

Figure 11.3. T_c vs. load (cent: τ_{ss} =0.1 τ_{cs}).

Figure 11.4. T_c vs. load (cent.: τ_{ss} = 0.01τ_{cs}).

Even though we use here for the centralized model a ten times faster server (as compared to the replica speed) and a ten times faster server-switch connection (as compared to the replica-switch connection), the replicated model was able to perform better in terms of delay and load saturation level, especially for small ratios of write queries (*w*) (Figures 12.1, 12.2, 12.3, and 12.4). (Note that Figure 12.1 and 12.2 are the same as Figures 11.3 and 10.1, reproduced here for the sake of the comparison).



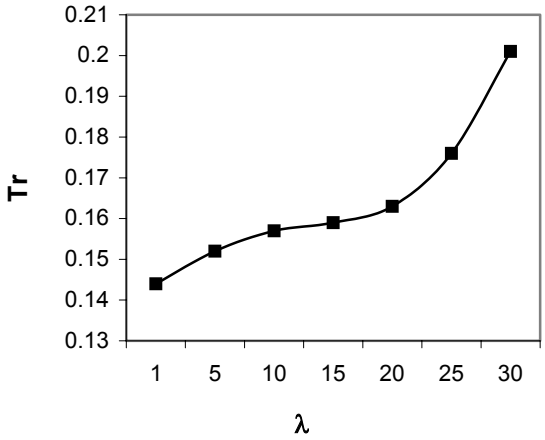
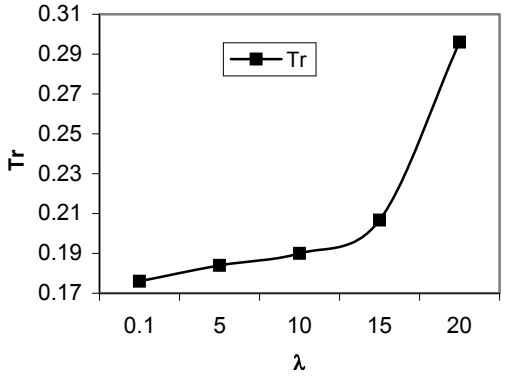
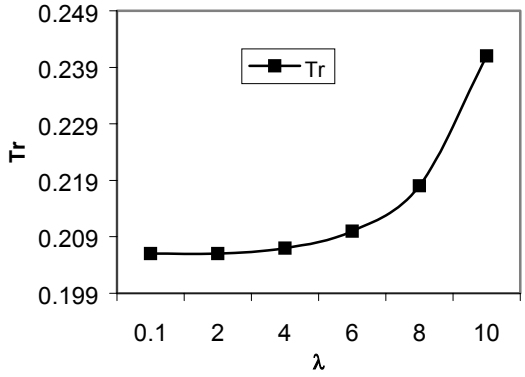


Figure 12.1. T_c vs. load (centralized).

Figure 12.2. T_r vs. load (replicated: *w*=0.2).

The gain in response time and saturation load of the centralized model over the replicated model is not as substantial as the invested resources (ten times more powerful server and ten times faster switch-server communication link).





Ref. Value	5	10	15
Mean	0.184	0.19	0.203
Std. Dev.	0.0013	0.0024	0.0025
95 % Conf. Int. of mean	0.1835 - 0.1854	0.1884 - 0.1919	0.2011 - 0.2047

Figure 12.3. T_r vs. load (replicated: w=0.4). **Figure 12.4.** T_r vs. load (replicated: w=0.8).

5. Conclusion

We have proposed a Colored Petri Net–based performance evaluation of the relative merits of the centralized approach and the replicated approach for network-based database systems. A number of simulation experiments have been carried out. In addition to the well-known benefits of higher reliability and data availability of replicated solutions (as compared to centralized ones), our simulation results have revealed that the replicated solutions may offer smaller query response time and higher load saturation levels. It can also be concluded that in order to compete with replicated systems, the centralized solution requires using much more powerful database servers.

It is to be noted that we have assumed replicated systems with minimum synchronization overhead involving a single broadcast message per write transaction. For more complex synchronization protocols, further investigation of the effect of the consistency control method on the performance of the replicated system is needed.

References

BEERI, C., BERNSTEIN, P.A., and GOODMAN, N. 1989. A model for concurrency in nested transaction systems, *Journal of the ACM*, **36:**230–269.

DAY, K., BEN MNAOUER, A., MASOUD, F.A., and TOWAIQ, M. 2001. A Fault Tolerant Asynchronous Data Replication Protocol, *Proceedings of the IASTED: International Conference on Applied Informatics (AI2001)*, February 19 to February 22, 2001, in Innsbruck, Austria, 2001.

GRAY, J.N., HELLAND, P., and D.O'NEIL, S.P. 1996. The Dangers of Replication and a Solution, *Proceedings of the 1996 ACM SIGMOD Intl. Conf. on Management of Databases*, pp. 173-182, Montreal, Canada, June 1996. SIGMOD.

GUERRAOUI, R., and SCHIPER, A. 1997. Software-based replications for Fault Tolerance, *IEEE Computer,* **30:** 68-74.

Ref. Value	4	6	8
Mean	0.207	0.212	0.218
Std. deviation	0.0034	0.0012	0.0007
95 % Conf. Int. of mean	0.2048 - 0.2098	0.2114 - 0.2131	0.2171 - 0.2185

JENSEN, K. 1992, 1994: *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use.* Vol.1 and Vol.2, Monographs in Theoretical Computer Science, Springer-Verlag, 1992, 1994.

JENSEN, K., CHRISTENSEN, S., HUBER, P., and HOLLA, M. 1996. *Design/CPN: A reference Manual*, Computer Science Dept, University of Aarhus, Denmark, 1996.

JORGENSEN, J. B., KRISTENSEN, L. M. 1999. Computer Aided Verification of Lamport's Fast Mutual Exclusion Algorithm Using Coloured Petri Nets and Occurrence Graphs with Symmetries. *IEEE Transactions on Parallel and Distributed Systems*. **10:** 714-732.

KEMME, B., and ALONSO, G. 1998. A suite of database replication protocols based on group communication primitives, *Proceedings of the 18th International Conf. On Distributed computing systems (ICDCS'98),* Amsterdam, The Netherland, May 1998.

KEMME, B., and ALONSO, G. 2000. A new Approach to developing and implementing eager database replication protocols", *ACM transactions on Database Systems* **25:** 333-379.

MNAOUER, A.B., SEKIGUCHI, T., FUJII, Y., ITO, T., and TANAKA, H. 1999. Colored Petri Nets Based Performance Evaluation of the Static and Dynamic Allocation Policies of the

Asynchronous Bandwidth in the Fieldbus Protocol, *Advances in Petri Nets (LNCS* **1605:** 93-130), *Special issue on the Application of Petri Nets to Communication Networks*, Springer-Verlag 1999.

MNAOUER, A. B., DAY, K., SHIHAB, K. 2002. Impact of Leaky Bucket Regulation on the Performance of Combined I/O Buffering in ATM Switches, the *2002 IEEE Intl. Conf. on Systems, Man, and Cybernetics*, Hammamet, Tunis, 6-9 October 2002, MP1A2, 1-6.

MORERA, P.H., GONZALEZ, T.M. P. 1999. *A CPN Model of the MAC Layer*. In K.Jensen (ed.): *Proceedings of the 2nd Workshop on Practical Use of Coloured Petri Nets and Design/CPN*, Aarhus 1999, Department of Computer Science, University of Aarhus, 153-172.

PEDONE, F., GHERRAOUI, R., and SCHIPER, A. 1997. Transaction Reordering in Replicated Databases, *Proceedings of the 16th Symposium on Reliable Distributed Systems (SRDS-16)*, Durham, North Carolina, USA, October 1997.

PEDONE, F., GHERRAOUI, R., and SCHIPER, A. 1998. Exploiting Atomic Broadcast in Replicated Databases, *proceedings of EuroPar'98*, Sept. 1998.

SHAPIRO, R.M. 1991. Validation of a VLSI Chip Using Hierarchical Coloured Petri Nets. Journal of Microelectronics and Reliability, Special Issue on Petri Nets, Pergamon Press, Vol. 31, No. 4, pp. 607-625, 1991.

JENSEN, K and ROZENBERG, G. 1991. *High-level Petri Nets. Theory and Application*. Springer-Verlag, pp. 667-687.

WIESMANN, M., PEDONE, F., and SCHIPER, A. 1999. A Systematic Classification of Replicated Database Protocols Based on Atomic Broadcast, *Proceedings of the 3rd European Research Seminar on Advances in Distributed Systems (ERSADS'99)*, Madeira Island, Portugal, April 1999.

WIESMANN, M., PEDONE, F., SCHIPER, A. KEMME, B., and ALONSO, G. 2000. Database replication techniques: a three parameter classification, *Proceedings of 19th IEEE Symposium on Reliable Distributed Systems (SRDS2000)*, Nürenberg, Germany, October 2000. IEEE Computer Society.

Received 24 March 2002
Accepted 19 April 2003