TECHNICAL ARTICLE

SIMULATION 75:4, ###-###
©2000, Simulation Councils Inc.
ISSN 0037-5497/00
Printed in the United States of America

A Parallel Simulation Methodology for Speedup and Obtaining Performance Estimates with Specific Accuracy: Experiences of its Application in Studies of Metropolitan Area Networks

Victor Yau
Department of Computer Science and Engineering
University of Texas at Arlington
Box 19015, Arlington, Texas, 76019-0015
E-mail: yau@cse.uta.edu

Krzysztof Pawlikowski
Department of Computer Science
University of Canterbury
Christchurch, New Zealand

Performance measures obtained by stochastic simulation are estimates, and one must consider the precision of the estimates before making any constructive conclusions about the investigated systems. This paper applies an automated distributed simulation method, called Spectral Analysis in Parallel Time Streams, to speed up production of performance estimates, and for run-length determination in the simulation of High-speed Metropolitan Area Networks (MANs). This method makes sequential simulators suitable for parallel execution on multiprocessors and/or networked computers. At runtime it estimates the information content of observations generated during simulation, generates a point estimate and confidence interval, and directs the run to continue until an estimate is obtained that achieves or exceeds our required level of precision. The application of this methodology for studying high-speed MANs, the speedup, intermachine communication and "warm-up" overhead, and the run lengths needed to produce estimates with a specific level of precision, are reported for each of the parameters investigated. Practical implications are discussed.

Keywords: Distributed simulation, speedup, precision, spectral analysis, bandwidth multiuse, distributed queue dual bus, delay, initial transient

Note: *This article continues the work described by the same authors in last month's article, "Analysis of Dual-Bus Metropolitan Networks Using Distributed Quantitative Stochastic Simulation" (SIMULATION, Vol. 75, No. 3, pp. 157-169).—Ed*

1. Introduction

Obtaining reliable estimates of steady-state performance measures by means of stochastic simulation is difficult since observations generated by the simulated process are autocorrelated. Thus their information content *is unknown and typically less* than that of observations collected by random sampling so that classical statistical techniques cannot be applied for inferring the precision of estimates [1–7]. Moreover, each simulated process traverses an *initial transient* ("warm-up") period. Observations generated during this "warm-up" phase do not characterise the steady-state behaviour of the simulated process [1, 6]. Thus simulation output data have to be properly, statistically analysed before any constructive conclusions about the investigated networks are made. As stated by J. Kleijen, a world authority on simulation:

"...computer runs yield a mass of data but this mass may turn into a mess" if the random nature of output data is ignored, and then "...instead of an expensive simulation model, a toss of the coin had better be used..."

Unfortunately, simulations in electrical and telecommunications engineering are often computationally intensive and can require excessively long run times in order to obtain results at a desired level of precision [8, 9, 10]. Moreover, our simulation studies are often iterative, so once results are at hand, they often motivate questions which require more runs to answer, and so on.

Recognising the need for reliable results within a practical time, a special method, called Spectral Analysis in Parallel Time Streams (SA-PTS), based on the method of Spectral Analysis [11] for sequential simulations, was developed to produce performance estimates with a given level of precision, and for run-length determination [7, 12, 13, 14]. SA-PTS also speeds up simulation using multiple Simulation Engines (SEs) which run in parallel on multiple workstations. SA-PTS has being implemented in AKAROA, an object-oriented simulation package developed by us for automated precision control of steady-state estimates and automatic generation and parallel execution of quantitative simulations. This paper is devoted to the application of AKAROA in simulation studies of standard Distributed Queue Dual Bus (DQDB) high-speed metropolitan area networks (as adopted by the IEEE, ISO, ANSI and ETS standards organisations [15, 16]) as well as enhanced versions of DQDB. In particular, the *real-time speedup, inter-machine communication and "warm-up" overheads, and run lengths needed, will be assessed for each DQDB performance parameter*.

Section 2 introduces DQDB networks, their modelling assumptions, and the performance parameters to be estimated during each simulation run. Section 3 reviews the SA-PTS methodology implemented by AKAROA and explains its application in the studies of DQDB. The effectiveness of AKAROA in speeding up DQDB network simulations, as well as its overheads, are analysed for each of the target performance parameters using 600 benchmark experiments. Results from the benchmarking and from actual production runs are summarised in Section 4. Section 5 focuses on some practical implications.

2. DQDB Networks

DQDB (Distributed Queue Dual Bus) is the Medium Access Control (MAC) protocol adopted by the IEEE as well as by the International Standard Organisation (ISO) as a Metropolitan Area Network (MAN) standard [15, 17, 18, 19, 20, 21, 22, 23]. DQDB has also been adopted as a European Telecommunications Standard (ETS 300). Already services[1] provided by networks using the DQDB protocol have being introduced in the United States, Europe, and Australia. Also, methods are under research for upgrading standard DQDB in the future; e.g., see [24]. The standard DQDB architecture and a full protocol definition are given elsewhere [15, 24], and will only be outlined here.

The DQDB network is composed of two high-speed (approximately 150 Mbps) unidirectional busses, carrying fixed-size slots in opposite directions; see Figure 1(a). Each slot comprises an Access Control Field, and a payload field that can carry one fixed-length data packet (called a *segment*); see Figure 1(b). Interconnecting stations using a (dual) bus topology introduces a "pecking-order" for stations which is a function of their relative position along the respective bus. Stations compete for empty slots subject to the media access rules of DQDB. The DQDB protocol tries to treat all stations equitably. Thus the objective of many simulation studies of the "fairness" of DQDB networks is to estimate the mean access delay of station i over bus A,

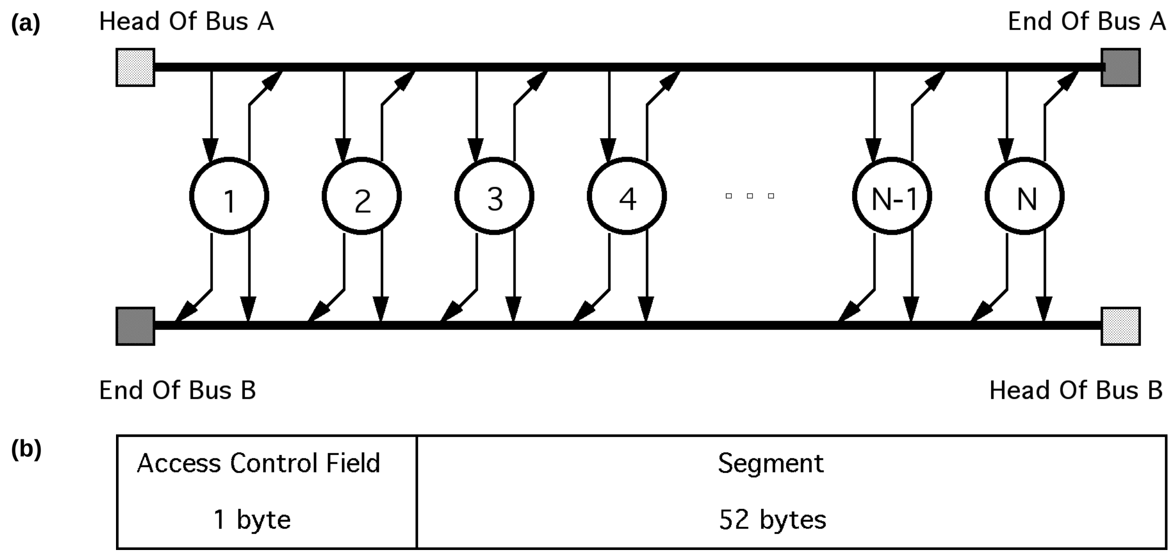

Figure 1. DQDB network: (a) dual bus technology; (b) the slot format

[1] For example, the Switched Multi-Megabit Data Service (SMDS) and the Connectionless Broadband Data Service [16].

$E[D_i]$, defined as the mean length of time between the arrival of a segment to the head of the transmission buffer of station i for transmission on bus A, and its transmission. Results for bus B can be derived by symmetry. Let N be the number of stations.

The stream of segments arriving at station i from its local data sources is modelled as a Poisson process with arrival rate λ_i segments per slot time (for i = 1, 2, .., N). Traffic generated by each station is uniformly distributed over all possible destinations. Thus, in the case of bus A, where stations from head-of-bus (HOB) to end-of-bus (EOB) are numbered from 1 to N, the traffic generated by station i and addressed to its downstream neighbours has the rate:

$$\lambda_{i,A} = \frac{N-i}{N-1}\lambda_i$$

The rate of traffic generated by the same station i to its downstream neighbours on bus B is:

$$\lambda_{i,A} = \frac{N-i}{N-1}\lambda_i \qquad \text{for } i = 1, 2, ..., N$$

Each new segment arriving at a station is stored in an input buffer. Two such input buffers are assumed per station, one per bus, and each of capacity of M segments. The assumed interstation propagation delay equals one slot time. This gives a bus length of approximately 4 km, for N = 10 stations and transmission speed of 155 Mbps.

In all benchmarks and production runs, we simultaneously estimate $E[D_i]$, i = 1, 2, ..., N – 1 during a single experiment, plus the mean network throughput.

3. Simulation Methodology

AKAROA accepts sequential un-instrumented simulators written in C or C++, or one built using AKAROA's Build toolkit and transparently transforms it into one suitable for parallel execution on a network of workstations. Figure 2 outlines the DQDB simulator development. It comprises three phases: (1) Sequential DQDB simulator construction using C/C++ or using AKAROA's Build toolkit; (2) Specify desired level of precision (using AKAROA's Control module), and (3) Parallel Simulation Engine generation (automated using AKAROA's Parallel Simulation Manager module).

3.1 Sequential Simulation Construction

All sequential DQDB simulators were constructed using AKAROA Build. Build is an object-oriented toolkit for fast construction of discrete event simulation models in C/C++. The use of Build is optional, and AKAROA can be used with any simulator written in C or C++. The only changes needed are that the random number generator and data collection facility provided by AKAROA should be used instead, and that the user specify the desired level of precision of the results (explained in Section 3.2).

Using the Build toolkit, the structure of DQDB simulators mirrors that of real DQDB networks. From Figure 1, we see that a real DQDB network is comprised of N stations, slots on two busses, and segments. Accordingly, three classes of objects were defined to model stations, slots, and segments. Station and slot classes were defined using Build's base classes such as queues, plus user-coded member functions, mainly to model the DQDB protocol's Media Access Control (MAC) rules followed by stations. Segments were modelled directly using Build's entities base class. The scheduled events are slot arrival (to stations), and segment generation (at stations for transmission). When a scheduled event occurs, member functions of entities affected by the event are invoked. For example, when slots arrive at stations, the *processnewslots* member functions of stations are invoked, so all stations can execute the MAC procedure of DQDB for accessing and processing header information in new slots. All activities in DQDB networks were sequenced using Build's event scheduler, and all random behaviour such as the inter-generation times of segments at stations were modelled using Build's statistical support functions.

3.2 Declaring Required Level of Precision of Results

The desired level of precision of estimates is easily declared in the DQDB simulator through a SPECTRALANALYSIS object like:

SPECTRALANALYSIS sav(sigmamax, 1-alpha, net_size);

The *sav* object (of class SPECTRALANALYSIS) is also used for automatic data collection, compaction, and statistical analysis of simulation output results. This is

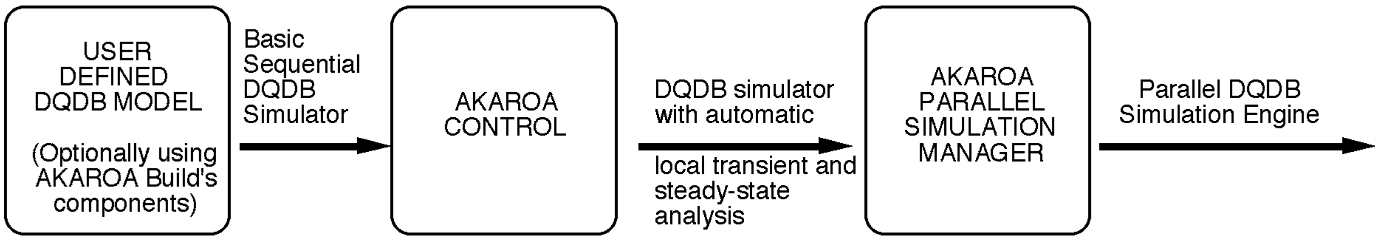
USER DEFINED DQDB MODEL
(Optionally using AKAROA Build's components)
Basic Sequential DQDB Simulator
AKAROA CONTROL
DQDB simulator with automatic
local transient and steady-state analysis
AKAROA PARALLEL SIMULATION MANAGER
Parallel DQDB Simulation Engine

Figure 2. Main phases of distributed DQDB Simulation Engine construction with AKAROA

```
vya@micky /net/u/v/vya/theaikid/theaikid/tony/simu/pdqdb/B2/P8 199 > akstat
AKAROA/AKSTAT: Host micky contacting Directory_Central on sleepy, port 2020

AKAROA/AKSTAT: DIR_CENTRAL>     ---------------------------------------------------
AKAROA/AKSTAT: DIR_CENTRAL>        Hostname   (Membership_Class)

AKAROA/AKSTAT: DIR_CENTRAL>     ---------------------------------------------------
AKAROA/AKSTAT: DIR_CENTRAL>          sneezy             ( 1 )
AKAROA/AKSTAT: DIR_CENTRAL>          bashful            ( 1 )
AKAROA/AKSTAT: DIR_CENTRAL>          happy              ( 1 )
AKAROA/AKSTAT: DIR_CENTRAL>          sleepy             ( 1 )
AKAROA/AKSTAT: DIR_CENTRAL>          daisy              ( 1 )
AKAROA/AKSTAT: DIR_CENTRAL>          chekov             ( 1 )
AKAROA/AKSTAT: DIR_CENTRAL>          fish               ( 1 )
AKAROA/AKSTAT: DIR_CENTRAL>          koala              ( 1 )
AKAROA/AKSTAT: DIR_CENTRAL>
AKAROA/AKSTAT: DIR_CENTRAL>        8 machines registered
AKAROA/AKSTAT: DIR_CENTRAL>
AKAROA/AKSTAT:
AKAROA/AKSTAT: Terminating Normally
vya@micky /net/u/v/vya/theaikid/theaikid/tony/simu/pdqdb/B2/P8 200 >
```

Figure 3. Viewing the AKAROA Domain with akstat

done by invoking the *processnewobs* member function of *sav* like:

stopsimulation = sav.processnewobs(access_delay, station_id);

whenever the *station_id*th station transmitted a segment (and hence knows the access_delay of that segment). *sav* returns STOP when the mean delays of all N – 1 stations[2] and the mean network throughput have been estimated to *sigmamax* level of relative precision or better. Otherwise CONT is returned (the simulation should continue).

This use of the SPECTRALANALYSIS object is the only change needed to a sequential DQDB simulator. All further steps in parallel simulator generation and execution, and online data analysis and precision control, are automated by AKAROA.

Meaning of Statistical Terms

Let:

$$P[X_i - H \leq E[D_i] \leq X_i + H] = 1 - \alpha$$

where X_i is the point estimate of $E[D_i]$. This means the probability that the true value of the mean access delay of station i lies within $(X_i - H, X_i + H)$ equals $1 - \alpha$. Then:

[2] The Nth station has no segments for transmission on Bus A, so its access delay need not be estimated. We let the Nth parameted processed by sav be for estimating the network's throughput. Thus N (=net_size) parameters are estimated during one run.

1. The interval (Xi – H, Xi + H) is called the *confidence interval* for E [Di]. H is often called the half-width of the confidence interval.
2. $1 - \alpha$ is called the *level of confidence*. Typical values are 0.95 or 0.99. This is specified using the second parameter (*"1 – alpha"* above) in the declaration of a SPECTRALANALYSIS object.
3. The ratio H/E[Di] is called the *relative precision* of the point estimate. The desired relative precision is specified by a user in the first parameter of the SPECTRALANALYSIS declaration (named "*sigmamax*" above).

3.3 Parallel Simulation Execution

3.3.1 Users' View

At runtime, AKAROA presents a network of workstations as a single (virtual) uniprocessor to the simulation user. Thus the user starts a distributed simulation run using almost the same command as its sequential counterpart, and can run the distributed simulation in the background, and suspend or terminate it just as if it were an ordinary simulation running on one machine. Option parameters that can be given by the user are the number of workstations to be used (the workstations can also be specified by names), and the priority levels for AKAROA processes on each workstation. For example:

key "pDQDBrunC t.dat t.sum 20 1.6" 10

runs a DQDB simulation using up to 10 machines to execute simulation engines. Similarly:

key "pDQDBrunC t.dat t.sum 20 1.6" –m chekov daisy fish koala sleepy happy bashful sneezy buntle altos

runs a DQDB simulation using workstations with host names chekov, daisy, fish, koala, sleepy, happy, bashful, sneezy, buntle, and altos.

The *key* command terminates when the simulation finishes, just like an ordinary sequential simulation. AKAROA commands can also be invoked within shell scripts. The only restriction is that AKAROA simulations should be executed on machines within the AKAROA Domain. The AKAROA Domain is the set of machines authorised to host AKAROA processes. Tools are provided for registering new machines, and for changing their membership class. Users can list machines in the Domain using *akstat*; see Figure 3.

3.3.2 Runtime Processes and Precision Control Method (Invisible to Users)

At runtime, the Parallel Simulation Manager (PSM) of AKAROA creates and maintains an environment for SA-PTS simulations. The PSM hides from the user the fact that the simulation executes as multiple processes on different machines. Simulation execution can be logically divided into a:

1. Launch phase,
2. Simulation Engine/global_precision_control process binding phase, and
3. Observation generation and analysis phase, followed by distributed termination.

Simulation Launch

The initiation of a simulation involves three main classes of PSM processes: *key*, *Directory_Central*, and *Local_Managers.*

1. The *key* processes (running on the user's machine) make a request for running a new simulation (LREQ) to AKAROA's *Directory_Central* process (possibly running on a remote machine).
2. Upon receiving the request, Directory_Central searches its Registered-AKAROA-Machine (RAM) database to find machines in the AKAROA domain that are most suitable for hosting Simulation Engines (SEs). Next Directory_Central sends an SE launch request to a Local_Manager process at each of the chosen machines.
3. Upon receiving an SE launch request, each Local_Manager[3] creates a new SE, and reports back to Directory Central.

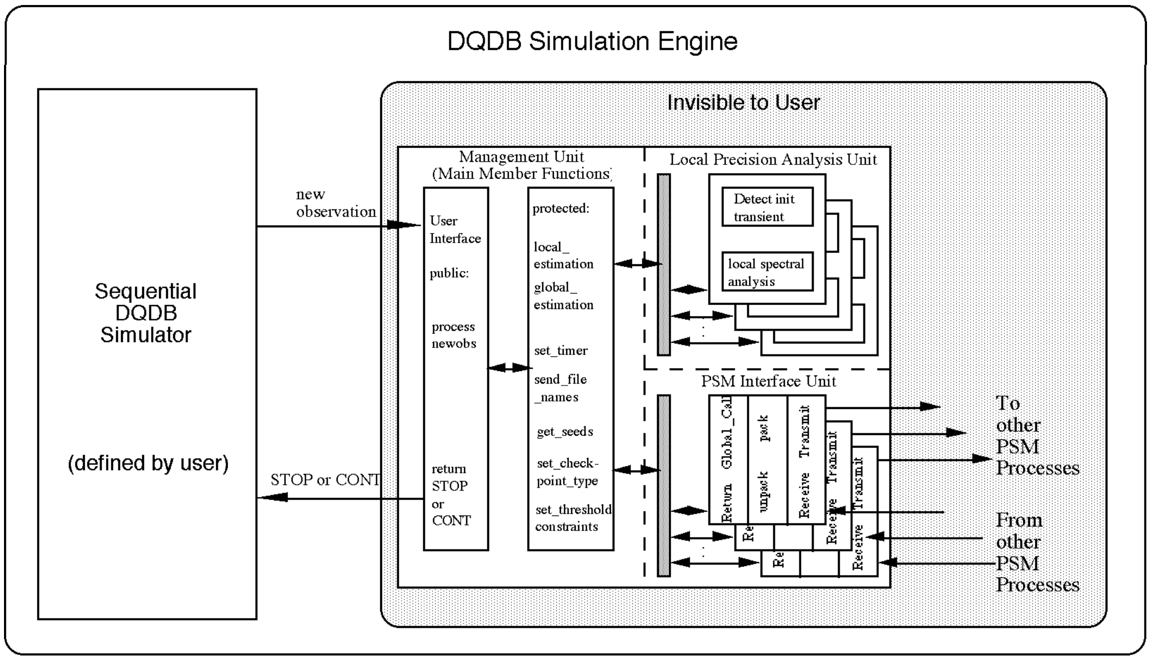

Figure 4. Structure of a DQDB Simulation Engine

[3] Each Local_Manager is responsible for fast creation and management of AKAROA processes on its machine, and for reporting on the machine status and load. We discarded the possibility of using rex, rlogin, or rsh facilities for remote process creation. They are not supported by some UNIX variants, and are inefficient for the operations AKAROA needs to perform. For example, to suspend a remote process using rsh, the invoking process must wait for a local rsh process to be created, then wait while a request is transmitted to the remote machine for a remote rsh peer to be created, then wait while environmental information is exchanged over the network, and then wait until the remote process invokes the kill call, send the process a suspend signal, and wait while the results of the kill call is returned to the local rsh process. The overhead of rsh is too high, in part because many features it provides are unnecessary for operations that are frequently used by AKAROA.

AKAROA automatically manufactures Simulation Engines (SEs) from non-parallel simulators written in C, C++, or AKAROA's Build toolkit. An SE is a replica of the DQDB model fitted with local transient and steady state analyzers (phase 2), and with a Parallel Simulation Manager interface unit (phase 3) for co-operating with multiple global precision control processes and other PSM processes during runtime. The structure of an SE is depicted in Figure 4. For efficiency, and for a model of communication which best encapsulates interactions between AKAROA processes, all inter-machine interprocess communication is done using PSM's Remote Function Invocation facility implemented on top of the Internet Domain Datagram type socket interface to UDP/IP. SEs access this facility through their PSM interface objects, depicted in the lower right quadrant of Figure 4.

Upon creation each SE will make a Generator Request (GREQ) remote function call to Directory_Central when its execution encounters the constructor of the SPECTRALANALYSIS object. The purpose of GREQ is twofold.

Each SE must use a different random number generator. The set of generators must be mutually uncorrelated, i.e., if we concatenate the sequences of random numbers of each generator into a macro sequence, then numbers in the macro sequence should also be uncorrelated. Currently such generators are created by partitioning the sequence generated by a well-tested generator into P_{max} subsequences. This yields up to P_{max} virtual generators of period L/P_{max}, or $P_{max}/2$ generators of period $2L/P_{max}$, or $P_{max}/3$ generators of period $3L/P_{max}$, and so on, where L is the period of the original generator. The seed to the start of each virtual generator is stored in Directory_Central. On receipt of a GREQ datagram, Directory_Central returns to the SE the seed to an unassigned virtual pseudo-random number generator from its generator base. The period of the virtual generators returned by Directory_Central is $\lfloor P_{max}/P \rfloor (L/P_{max})$ where $\lfloor X \rfloor$ denotes the largest integer no greater than X, and P is the number of SEs.

Secondly, Directory_Central updates its Active Simulation Engine (ASE) database with an entry for the calling SE. It holds information on the SE's status, machine address, port number, and process identifier.

Simulation Engine – Global_Precision_Control Process Binding

When P SEs are employed (each running in parallel on a different machine), then they have to cooperate with N Global_Precision_Control (GE) processes. Each GE process is responsible for estimating one parameter, using local results generated by all SEs. To enable co-operation, some facility must be provided for one process to *name* and *locate* the others.

We intentionally dismissed the possibility of including in the SE, GEs and other PSM programs the network machine addresses of the processes with which each of them need to communicate. This compile time binding of co-operating processes is undesirable, as it slaves us into using specific machines: any addition or removal or crashes of machines in the AKAROA domain cannot be accommodated without changing and recompiling several program codes. It also mean that the number and location of SE, GE, Local_Managers and other PSM processes have to be determined at simulator generation and compile time.

In the solution developed for AKAROA, when an SE initiates communication with a GE process, it first sends a Locate_Request datagram to Directory_Central (possibly on another machine), specifying the type and instance of GE process it seeks. Directory_Central is the only well known process in the AKAROA system. Upon receiving a Locate_Request, Directory_Central searches its Active_Control_Process (ACP) database for an entry with the requested type and instance. If one is found, it returns its machine address and port number to the caller SE. Otherwise, Directory_Central sends a request to a Local Manager on a selected machine, asking it to create a GE of the requested type and instance. The Local Manager returns the port number of the new GE to Directory_Central. In turn Directory_ Central updates its ACP database, and returns the machine address and port number to the (PSM interface object of the) caller SE. Subsequent communications between that SE and that specific GE are made directly, without needing to go through Directory_ Central.

This dynamic binding of cooperating processes also allows AKAROA to allocate processes *wherever* is most suitable (i.e., on which machines), and *whenever* they are needed at runtime.

Observation Generation and Runtime Data Analysis under SA-PTS

Each Simulation Engine uses a version of spectral analysis by [11] for estimating the variance of the point estimator from a regression fit to the logarithm of the averaged periodogram of the sequence of observations it generated after the estimated end of the initial transient (i.e., "warm-up") period. One object is dedicated to local transient and steady state analysis of each estimated parameter in each Simulation Engine; see the top right quadrant of Figure 4. At various checkpoints, local point and variance estimates are sent to their corresponding Global Precision Control processes, possibly on another machine (see Figure 5). The mth checkpoint of the ith parameter at the nth Simulation Engine, $C_{n,i}(m)$, is set to $C_{n,i}(1) = \max(200, 2T_{n,i})$, $C_{n,i}(m) = C_{n,i}(m-1) + 2T_{n,i}$, where $T_{n,i}$ denotes the estimated duration of the initial transient of the output process corresponding to the ith parameter as determined by a twin-gate test (see below). Setting the first checkpoint at the nth Simulation Engine to $C_{n,i}(1) = \max(200, 2T_{n,i})$ provides extra safety, as it means that the initialization

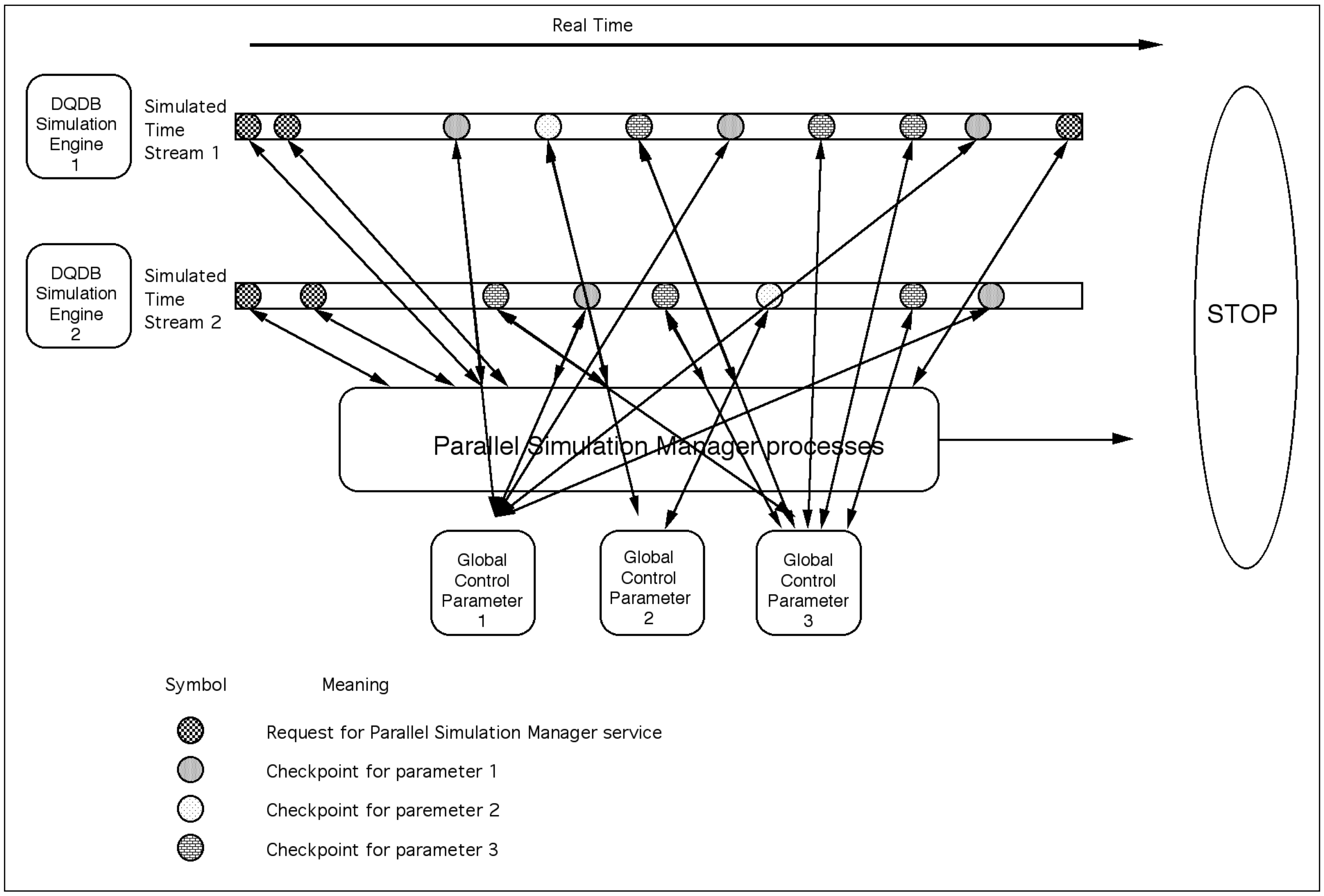
Real Time
DQDB Simulation Engine 1
Simulated Time Stream 1
DQDB Simulation Engine 2
Simulated Time Stream 2
STOP
Parallel Simulation Manager processes
Global Control Parameter 1
Global Control Parameter 2
Global Control Parameter 3
Symbol
Meaning
Request for Parallel Simulation Manager service
Checkpoint for parameter 1
Checkpoint for paremeter 2
Checkpoint for parameter 3

Figure 5. Virtual-real time interactions between main AKAROA processes during parallel execution of a DQDB simulation using two Simulation Engines and estimating three parameters

bias remaining in the first local estimate would be limited, even if P is very large. Each Global Precision Control process is responsible for collecting local estimates from all Simulation Engines, and computing a global estimate [12, 13] of its assigned parameter. Thus in the case of an N station DQDB network, N Global Precision Control processes would be created, one for estimating the access delay of each station[4], plus one for estimating the overall network throughput and delay. Figure 5 depicts the interactions between DQDB Simulation Engines and the Global_Precision Control processes when two SEs are used and three parameters are estimated during one run.

As mentioned, another pitfall in the simulation of telecommunication systems is that each simulated process traverses an *initial transient* ("warm-up") period. Observations generated during this "warm-up" phase do not characterise the steady-state behaviour of the simulated process [1, 3, 4, 6, 8]. For example, a natural point estimator of the steady-state mean access delay encountered by segments at station S_i before they are transmitted is the arithmetic average of the sequence of delay observations of segments transmitted from that station. Unfortunately, this point estimator is biased, since we must initialise our simulator to some initial state (in our simulations, the empty-and-idle state was chosen). Thus, during the start of the simulation, segments can typically be transmitted from the station with less delay than that of segments generated at the station at later simulated times, when the simulated network behaviour is closer to its normal loaded state. Including the delays of segments transmitted at the start of simulation in the point estimate gives an over-optimistic (i.e., smaller) estimate of the steady state mean segment delay.

To avoid this mistake, for each Simulation Engine and for each estimated parameter, AKAROA uses a twin-gate procedure comprising of Gafariants Heuristic [25] with k = 25 crossings to obtain a rough estimate of the end of the transient phase [1, 6, 8], followed by repeated applications of Schruben's test [26] until steady state conditions were detected for that process at the $\alpha = 0.05$ level of significance. In all simulated systems, and in the estimation of all parameters, observations collected during the transient period were discarded. This means that only observations generated after the estimated end of the initial transient were used in forming the point estimates.

[4] Except the station at the end-of-bus

Distributed Termination Phase

Each Global Precision Control processes (GE) maintains only information pertaining to the parameter it is responsible for estimating. Therefore no GE knows when all estimates have achieved the required precision. In the solution implemented in AKAROA, each SE is responsible for maintaining its copy of a progress status board. This has N squares, one per parameter. Whenever an SE reaches a checkpoint for parameter p_i, it makes a Global_Estimation call to the GE responsible for p_i. The value returned by the GE indicates whether the global estimate of p_i has attained the required precision. If p_i has achieved the needed precision, the SE puts a check in the ith square of the status board. Next, if the SE finds that all N boxes of its status board are checked, it will initiate the termination sequence by making a Stop Request (SREQ) call to Directory_Central. Upon receiving a SREQ, Directory_ Central steps through its ASE database. For each entry except the one corresponding to the SE which invoked SREQ, Directory_Central requests the Local_ Manager on the machine hosting the SE to send a Termination Request (TREQ) signal to the SE. Finally Directory_ Central completes the simulation by sending a TREQ to the *key* process.

4. Performance Results

4.1 Real-Time Speedup

Define *real-time speedup in estimating E[Di]* to a specific precision using P processors, $\text{Speedup}_i(P)$, as:

$$\text{Speedup}_i(P) = \frac{\text{real-time used to estimate } E[Di] \text{ using 1 processor}}{\text{real-time used to estimate } E[Di] \text{ using P processors}}$$

$\text{Speedup}_i(P)$ measures the mean real-time speedup seen by an AKAROA user if he/she was estimating the mean access delay of the ith station in a DQDB network to a specific precision.

With AKAROA, it is also possible to estimate a number of parameters during a single simulation experiment. Define *overall simulation speedup* using P processors to estimate N parameters to a specific precision, Speedup*(P), as:

$$\text{Speedup*}(P,N) = \frac{\text{real-time used to complete the simulation using 1 processor}}{\text{real-time used to complete the simulation using P processors}}$$

The Speedup*(P) that is achieved depends on the expected duration of the most difficult parameter being estimated. Thus we can write:

$$\textit{Speedup*}(P,N) = \frac{\max\{\text{time to estimate i}^{\text{th}} \text{ parameter using 1 processor}; i = 1, \ldots, N\}}{\max\{\text{time to estimate i}^{\text{th}} \text{ parameter using P processors}; i = 1, \ldots, N\}}$$

The performance of AKAROA was assessed using a series of 600 benchmark simulation experiments using P = 1, 2, 4, 6, 8, and 10 processors. All experiments simulated an N = 10 station DQDB network where each station has a transmission buffer for Bus A that can store M = 20 segments. An inter-station propagation delay of one time slot, and a normalized load of $E[\lambda_A] = 0.80$ was assumed. In all benchmarks, we simultaneously estimated $E[D_i]$, i = 1, 2, ..., 9 during a single run, plus the mean network throughput. *All ten estimates were required to have the relative precision of ε_{max} = 1% (or less), at the 0.95 confidence level*. Each experiment was repeated 100 times using 1, 2 3, 6, 8 and then 10 workstations.

In these experiments, the AKAROA domain consists of ten SUN ULTRA 10 workstations. All of them have a SPARC CPU operating at 300 MHz and a sparc FPU. They are interconnected by 10Mbps Ethernet. Four of them have 384 Mbytes of real memory; the remainder have 128 Mbytes. We executed the P = 1 benchmarks using one of the 384Mbyte machines, and the others using a mixture. Thus we expect our results for speedup to be conservative. To verify that all machines were idle (except for AKAROA's processes) during benchmarks, we sampled the percentage of idle CPU (available for user processes) on every employed workstation between every simulation run. A buffer time of eight seconds was provided before and after running the load sampling program to avert I/O contention from paging and/or swapping from the task switching.

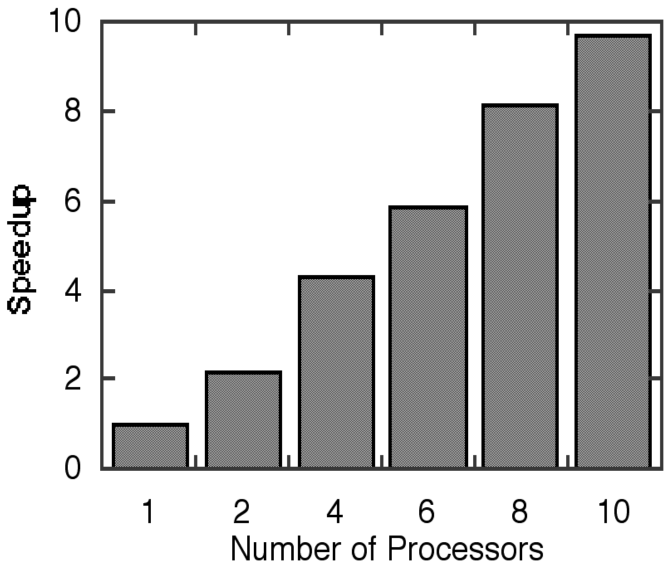

Figure 6. Overall real-time speedup when estimating 10 parameters during one experment

The overall simulation speedups when using P workstations, averaged over 100 benchmark experiments for each value of P, is graphed as a function of P in Figure 6. As shown, using P workstations on average completed the experiment P times faster. Further, the average speedup exceeded P in some cases. This can be partly credited to the reduction in overshooting when P > 1 (see below}.

Sometimes a network designer is interested in estimating the mean access delay of a specific station. What real-time speedup can be achieved by using P workstations to estimate a specific $E[D_i]$? Figures 7, 8, and 9 report the average real-time speedups when estimating $E[D_i]$, for i = 1, 2, ..., 9, using P = 4, 8, and 10 workstations, respectively. These results show excellent speedup in most cases, where using P workstations speeds up the estimation of the parameter by P times. Speedups greater than P were observed when estimating the access delay of stations near the head and end of bus (stations at positions 1 and 9).

4.2 *CPU Time Speedup in Parallel DQDB Simulations*

Define CPU-Speedup*(P) as the reduction in CPU time per processor (normalised to the average time needed to generate an observation), when P processors are used. Let $\text{CPU-Speedup}_i(P)$ be the corresponding CPU-speedup measure for the estimation of $E[D_i]$.

In contrast to Speedup*(P,N) and $\text{Speedup}_i(P)$, CPU-time speedup measures the reduction in computation time per machine engaged in simualtion. Thus it measures speedup in ideal situations, when losses of time caused by interprocess communication, simulation launch, binding of cooperating processes, and termination are negligible.

The average CPU-Speedup*(P) are reported in Figure 10. Comparing the replication lengths as a function of P, we observe that the reduction in CPU time with P workstations is near 1/P or better. The better than 1/P CPU-Speedup*(P) can be due partly to reduced overshooting, and the fact that *n* observations generated in parallel by P Simulation Engines have greater entropy than if they were collected from a single Simulation Engine.

Results for CPU-Speedup(P) as a function of i, for P = 4, 8, and 10, are reported in Figures 11, 12, and 13. Near or above linear CPU-speedups can be observed. It should be emphasized that in our benchmarks, the number of observations collected for estimation (as reported in Figures 10, 11, 12, and 13) is typically less than the total number generated. For example, after the estimate of $E[D_5]$ has achieved the specified relative precision, delay observations for segments transmitted from station 5 would continue to be generated (though not used). The whole model must evolve until all 10 parameters have been estimated to the required precision.

The number of observations needed to estimate the access delay of a station varies strikingly from approximately 80,000 (station 9) to 520,000 (station 1). This

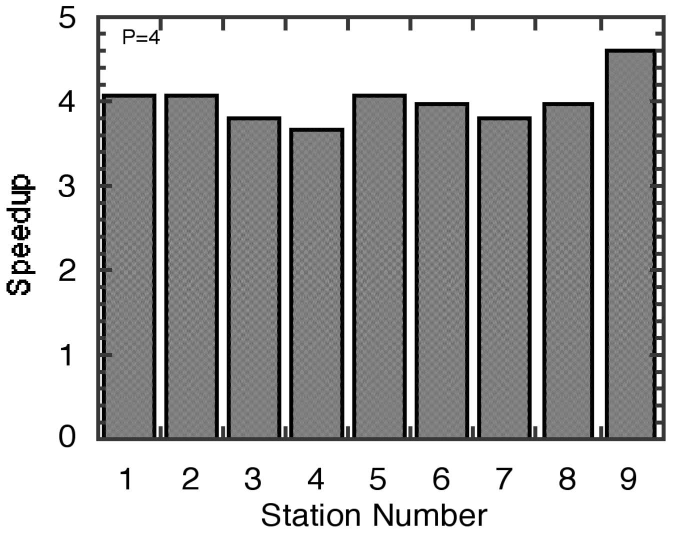

Figure 7. Real-time speedups in estimating access delays of stations 1 to 9 using P = 4 workstations

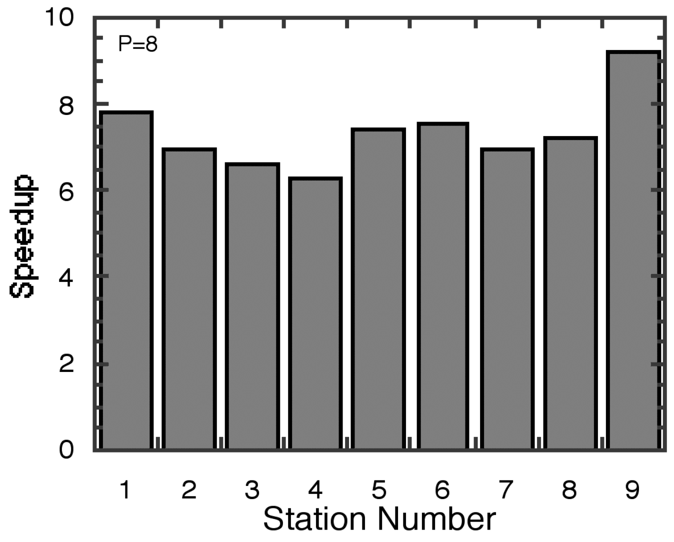

Figure 8. Real-time speedups in estimating access delays of stations 1 to 9 using P = 8 workstations

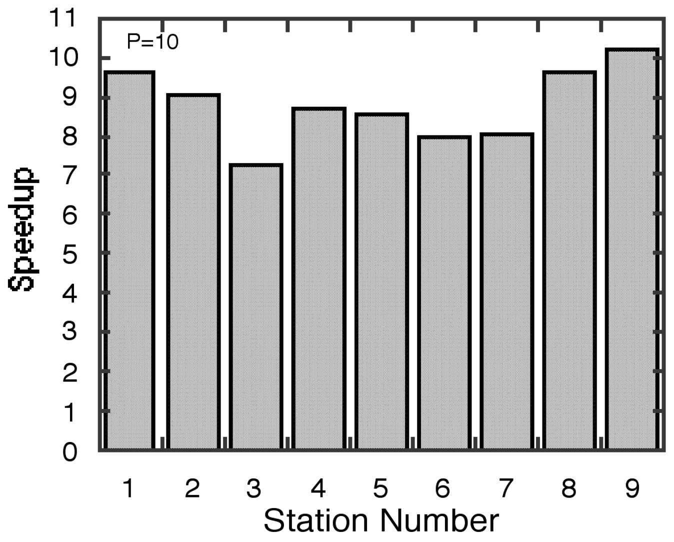

Figure 9. Real-time speedups in estimating access delays of stations 1 to 9 using P = 10 workstations

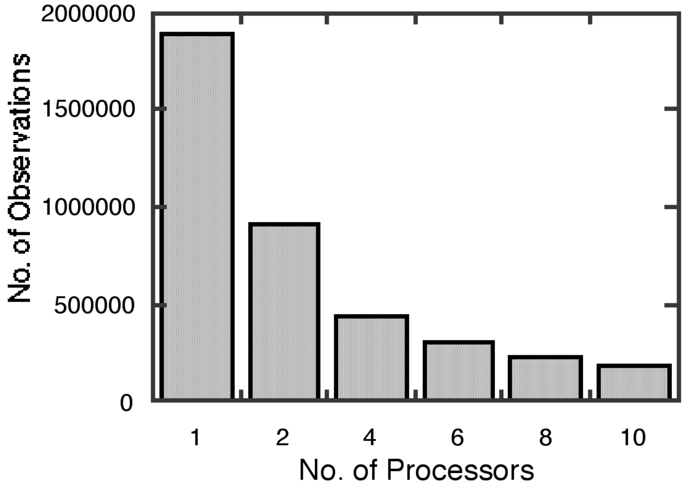

Figure 10. Average number of observations collected per simulation experiment

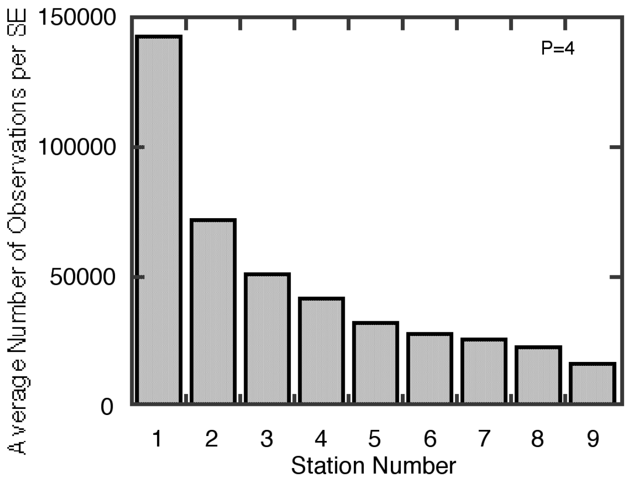

Figure 11. Average number of observations per SE for estimating E[D$_i$], i = 1, 2, ..., 9; P = 4

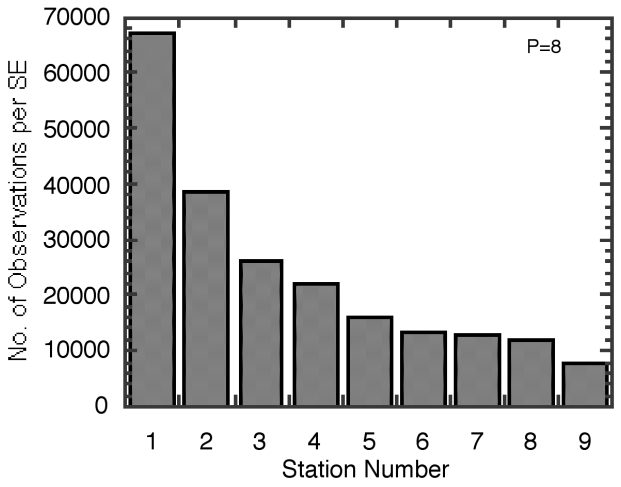

Figure 12. Average number of observations per SE for estimating E[D$_i$], i = 1, 2, ..., 9; P = 8

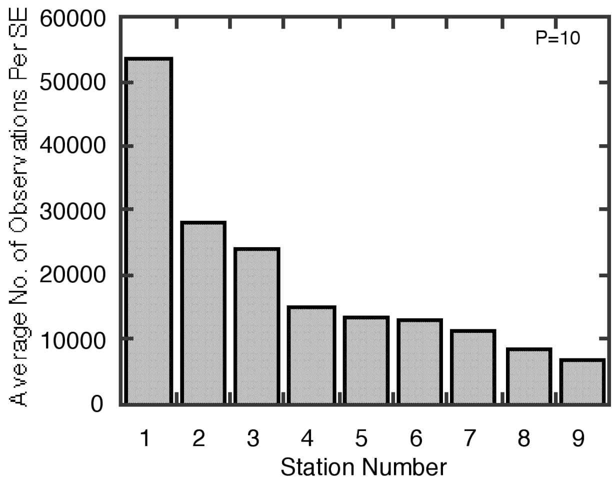

Figure 13. Average number of observations per SE for estimating E[D$_i$], i = 1, 2, ..., 9; P = 10

strongly highlights the importance of using dynamic run-length control, if results of specific accuracy are required. *In all cases the average number of observations needed for estimating $E[D_i]$ decreased as a function of .i* DQDB's MAC mechanism can restrain a station from using empty slots for a nondeterministic period (duration of its COUNTDOWN state [24]), so that some empty ones are left for stations downstream. However, DQDB is known to be unfair, e.g., see [24], and stations closer to the Head-of-Bus are less likely to be restrained (have better access to empty slots). Also, they have more segments for transmission on that bus. Consequently we intuitively expect that the sequences of access delays of stations near the Head-of-Bus are more correlated (since segments are less likely to be delayed by nondeterministic periods) than sequences from stations further downstream. This could explain the decrease in the average number of observations needed for estimating $E[D_i]$ as i increased.

4.3 Overshooting

In all benchmarks, the estimates can have higher relative precision than $\varepsilon_{max} = 1\%$; i.e., the estimation process *overshoots*. Figure 14 reports the relative precision of the final estimates, averaged over all 10 parameters and over all 100 experiments. As can be seen, *the level of overshooting is reduced as P grows*.

Within each Simulation Engine, local estimates of the ith parameter are produced at various checkpoints. The distance between consecutive checkpoints in the nth Simulation Engine equals $2T_{n,i}$ observations (refer to Section 3.3.2). A Simulation Engine using a smaller inter-checkpoint spacing typically invokes the corresponding GE more frequently. The minimum inter-checkpoint spacing when using P Simulation Engines is $\min\{2T_{n,i}, n = 1, 2, ..., P\}$. The expected minimum inter-checkpoint spacing thus decreases as P is increased. Thus the expected frequency of production of global estimates increases, lowering the "jumps" in relative precision of successive global estimates. This reduces overshooting.

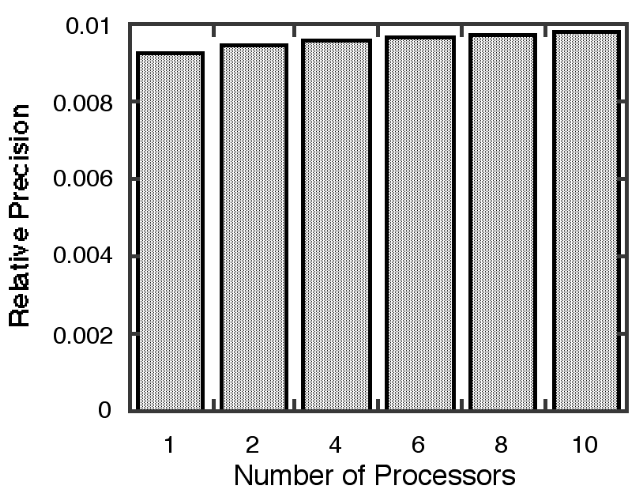

Figure 14. Relative precision of estimates at stopping point, averaged over all 10 parameters and over

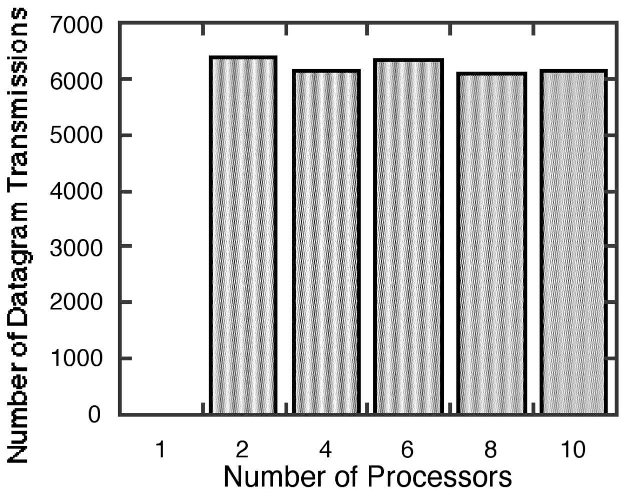

Figure 15. Average total number of datagrams transmitted per simulation experiment

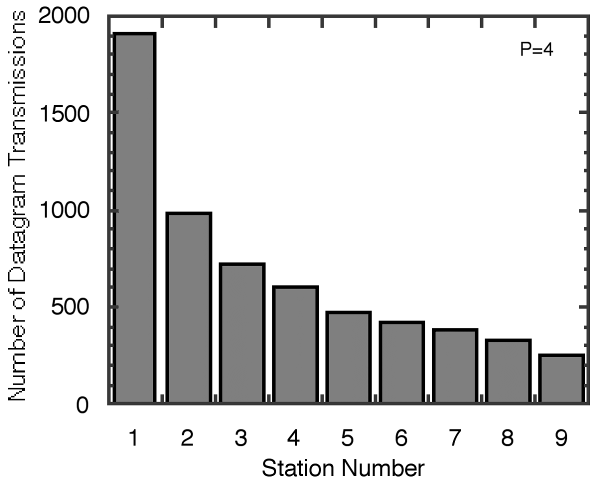

Figure 16. Average number of datagrams transmitted for estimating E[D_i], i = 1, 2, ..., 9; P = 4

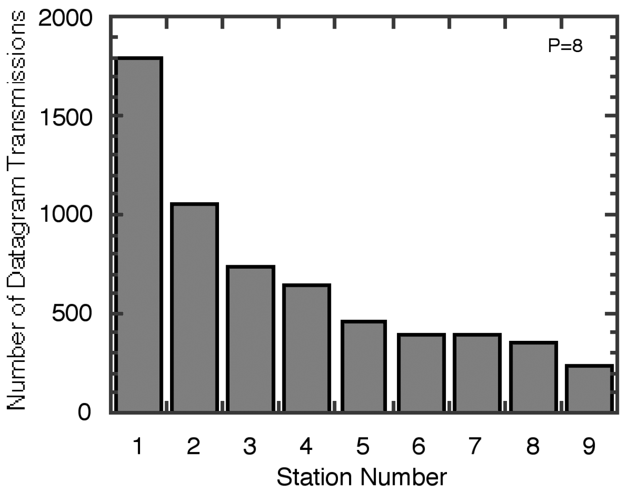

Figure 17. Average number of datagrams transmitted for estimating E[D_i], i = 1, 2, ..., 9; P = 8

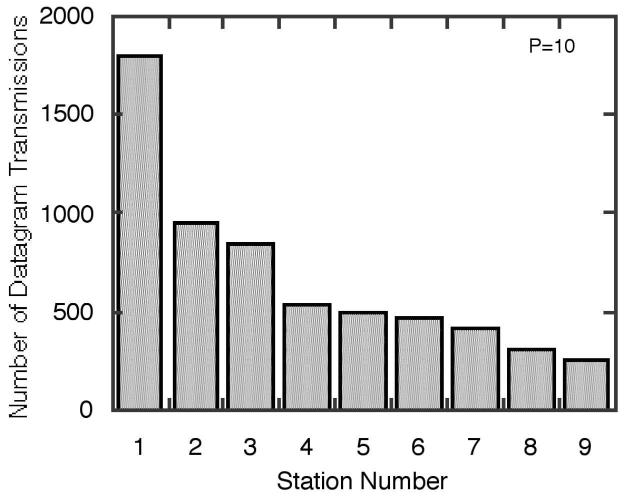

Figure 18. Average number of datagrams transmitted for estimating E[D_i], i = 1, 2, ..., 9; P = 10

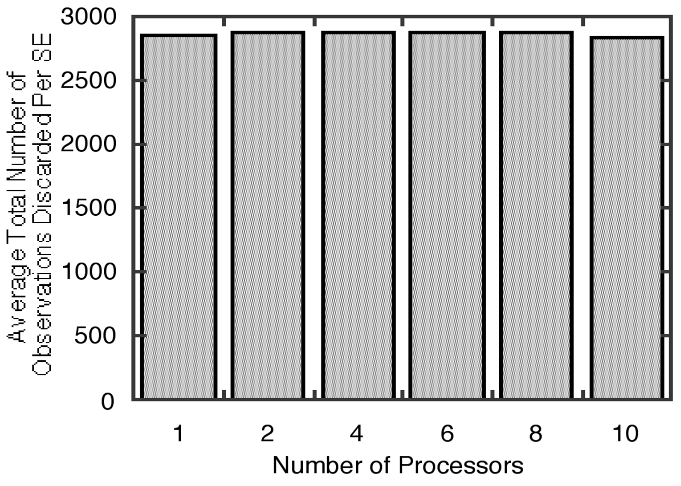

Figure 19. Average total number of observations discarded per SE

4.4 Inter-Machine Communication Overhead

The average number of datagrams transmitted during one DQDB benchmark experiment is plotted in Figure 15. As can be seen, on average over 6,000 datagrams were transmitted per simulation experiment when P > 1. The number of datagrams transmitted for the estimation of E[Di], i = 1, 2, ..., 9, when P = 4, 8, and 10 workstations were used are graphed in Figures 16, 17 and 18 respectively. The high real-time speed in spite of these large overheads shows the efficiency of AKAROA PSM's inter-machine interprocess communication facility.

4.5 Significance of the Initial Transient Periods

In all Simulation Engines, for each parameter, observations generated during the initial transient period (warmup phase) were discarded. The average total number of transient observations discarded by each Simulation Engine are reported in Figure 19. As shown, approximately 2,800 observations were discarded per Simulation Engine. This compares to an average run-length of 250,000 observations per SE when P = 10, to almost 2,000,000 observations when P = 1. Hence the overhead of the warmnp period is relatively low.

Figures 20, 21, and 22 report the average transient lengths of each parameter. As shown, the duration of the warmup phase is somewhat shorter for stations further from the Head-of-Bus.

4.6 Costs of Production Runs

AKAROA has been used to analyse in-depth the performance of standard DQDB networks, DQDB with slot reuse (DQDB/SRU), as well as DQDB networks with slot pre-and-reuse (DQDB/SMU) [25]. In these production runs, the access delays of stations and the throughput were estimated to a relative precision of 5% as determined by SA-PTS, for N = 120, M = 50, ρ = 0.4 and 0.90, respectively. A subset of run-length results are plotted in Figures 23 and 24 as a function of station index. DQDB/SRU and DQDB/SMU networks differ from standard DQDB in that stations near the End-of-Bus are also unfairly favoured, as they are more likely to re-use slots [25]. This is reflected in higher run-lengths needed to estimate their access delay.

5. Conclusions

The practicality of using the SA-PTS distributed simulation method to speed up DQDB simulations, and for obtaining performance estimates with specific accuracy, was assessed by 600 benchmark experiments, plus production runs.

The results show that using P workstations can speed up simulation experiments by almost P times. Speed-ups in the real time used to estimate each parameter were also measured. Results showed that near optimal linear speedup can also be achieved if the objective of

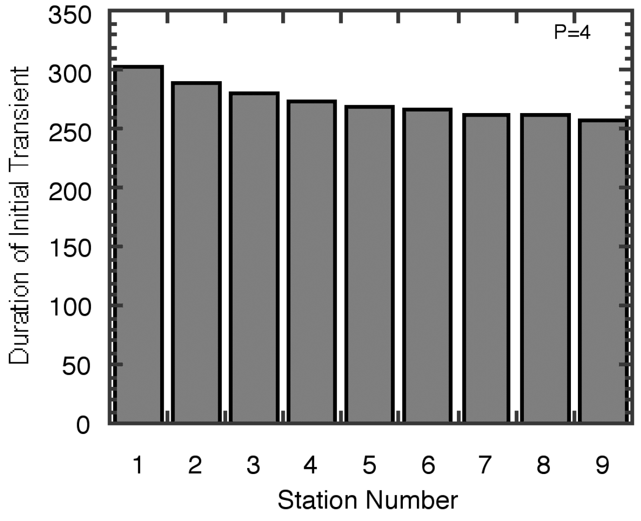

Figure 20. Average number of observations discarded per SE in estimating E[D_i], i = 1, 2, ..., 9; P = 4

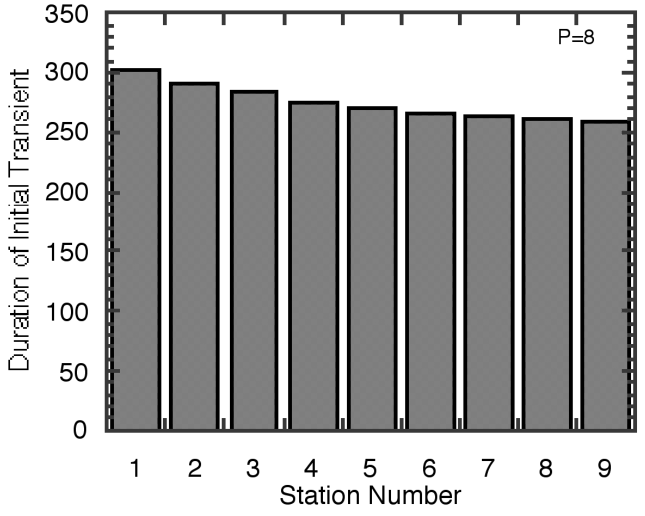

Figure 21. Average number of observations discarded per SE in estimating E[D_i], i = 1, 2, ..., 9; P = 8

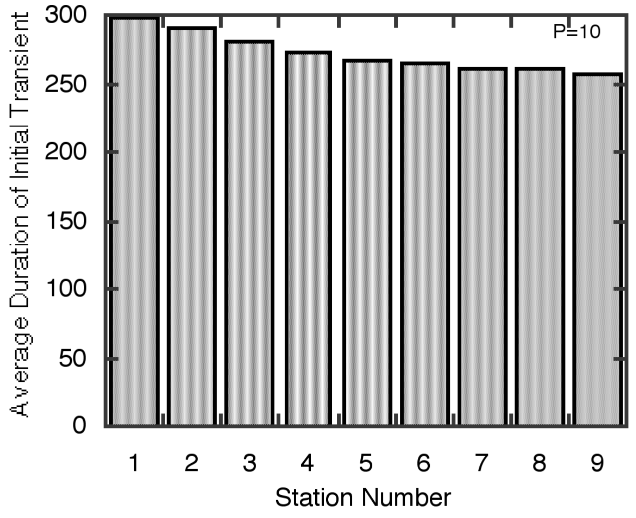

Figure 22. Average number of observations discarded per SE in estimating E[D_i], i = 1, 2, ..., 9; P = 10

the simulation was to estimate any one of the DQDB performance parameters.

The SA-PTS precision control methodology implemented by AKAROA incurred an intermachine interprocess communication overhead of 6,000 datagrams per simulation experiment, plus computation costs of data collection and producing local and global estimates at checkpoints. The high real-time speedup in the presence of these overheads suggests the efficiency of AKAROA's inter-machine interprocess communication facility, and that a good balance has been achieved between checkpoint spacing (level of overshooting) and communication. AKAROA uses discarding to improve the quality of estimates. The benchmarks showed that the lengths of initial transient periods (as estimated by a twin-gate test) of DQDB output processes were typically much less than 1% of the corresponding run-lengths. This suggests that in the case of DQDB studies, the overhead of discarding observations generated during the initial transient periods is relatively low.

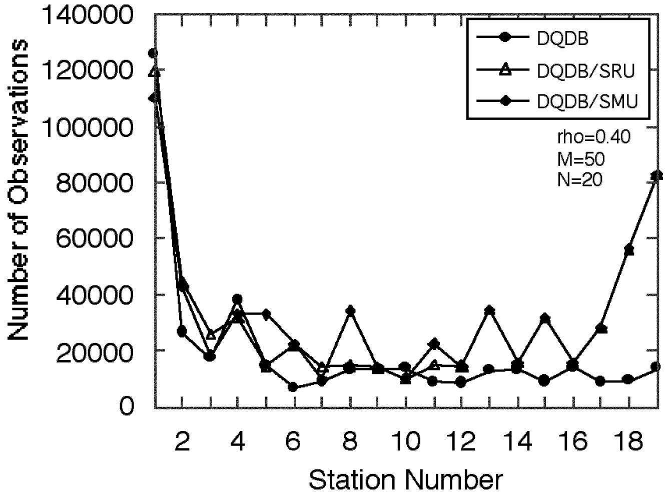

Figure 24. Run-lengths in estimating E = [D_i], i = 1 to 20, to a relative precision of 5% in simulations of DQDB, DQDB/SRU and DQDB/SMU networks; E[$\lambda_{i,A}$] = 0.90

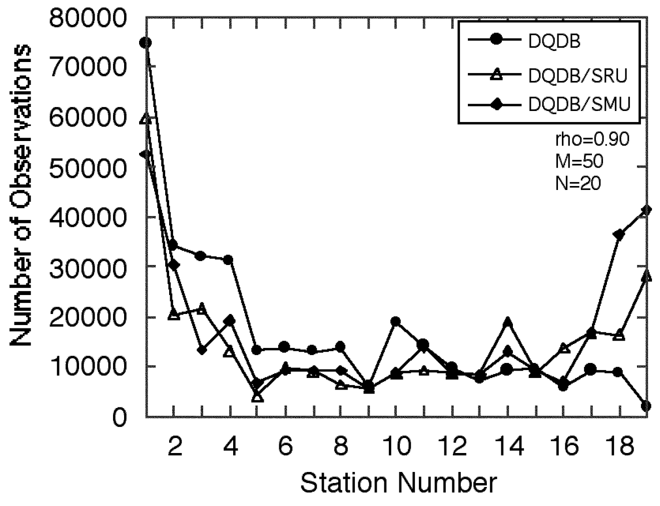

Figure 23. Run-lengths in estimating E = [D_i], i = 1 to 20, to a relative precision of 5% in simulations of DQDB, DQDB/SRU and DQDB/SMU networks; E[$\lambda_{i,A}$] = 0.40

The number of observations needed to estimate the access delay of segments transmitted from station i (as determined using SA-PTS) decreased strikingly as a function of i. DQDB is known to be unfair, e.g., see [24], and stations closer to the Head-of-Bus are less likely to be delayed from transmitting a segment. Consequently we intuitively expect that the sequences of access delays of stations near the Head-of-Bus are more positively correlated (since segments are less likely to be delayed by non-deterministic periods) than sequences from stations further downstream. Highly correlated observations have lower entropy (information content about the estimated parameter) than less correlated ones, so more observations must be collected before an estimate with the required accuracy is achieved. This could explain the increase in the average number of observations needed for estimating E[D_i] as i decreased. However, none was so large as to make obtaining reliable performance estimates of DQDB networks by SA-PTS using simulation studies infeasible. Given the availability of AKAROA (which fully automates the tasks of distributed SA-PTS simulator generation and execution, leaving to the user only the task of model specification), the development costs of distributed DQDB simulators that rapidly produce estimates with a required level of accuracy are just as low (or lower, using AKAROA Build) than that of writing "un-instrumented" simulators that produce results of questionable reliability.

6. References

[1] Pawlikowski, K. "Steady-state Simulation of Queueing Processes: A Survey of Problems and Solutions." *ACM Computing Surveys*, No. 2, pp 124-170, June 1990.

[2] Pawlikowski, K. and Yau, V. "Independent Replications Versus Spectral Analysis in Steady-State Simulation of High Speed Data Networks." *Proceedings of ATRS'91*, Nov. 1991, Wollongong, Australia, pp 182-190.

[3] Pawlikowski, K. "Simulation Studies of Telecommunication Networks and Their Credibility." *Proceedings of the European Simulation Multiconference (ESM'99)*, Warsaw, June 1999, Society for Computer Simulation, 1999.

[4] Pawlikowski, K., Ewing, G. and McNickle, D. "Coverage of Confidence Intervals in Sequential Steady-State Simulation." *Journal of Simulation Practise and Theory*, Vol. 6, No. 3, pp 255-267, 1998.

[5] Pawlikowski, K., Ewing, G. and McNickle, D. "Performance Evaluation of Industrial Processes in Computer Network Environments." *Proceedings of 1998 European Conference on Concurrent Engineering*, Erlangen, Germany, April 1998, Int. Society for Computer Simulation, pp 160-164, 1998.

[6] Yau, V., and Pawlikowski, K. "A Conflict-free Traffic Assignment Algorithm Using Forward Planning." *IEEE INFOCOM'96*, IEEE Comm. Press, Vol. 3, pp 1277-1284.

[7] Yau, V. and Pawlikowski, K. "AKAROA: A Package for Automating Generation and Process Control of Parallel Stochastic Simulation." *Proceedings of the 16th Australian Computer Science Conference*, G. Gopal, et al. (eds.), Australian Computer Science Communications, 1993, pp 71 -83.

[8] Yau, V. ''Optical WDMA Network Design and Function Placement.'' *Computer Networks Journa*l, Vol. 31, No. 22, pp 2411-2427, Dec. 1999.

[9] Yau, V. and Pawlikowski, K. "CA-STAR: A Centrally-Arbitrated Passive Broadcast-and-Select Star Architecture for Lightwave Networks." *Journal of High Speed Networks*, Vol. 8, No. 3, pp 1-21, 1999.

[10] Yau, V. and Pawlikowski, K. "An Algorithm that Uses Forward Planning to Expedite Conflict-Free Traffic Assignment in Time-Multiples Switching Systems." *IEEE Transactions on Communications*, Vol. 47, No. 11, pp 1757-1765, Nov. 1999.

[11] Heidelberger and Welch. "A Spectral Method for Confidence Interval Generation and Run Length Control in Simulations." *Communications of the ACM*, Vol. 24, pp 233-245, April 1982.

[12] Yau, V. "Automating Parallel Simulation Using Parallel Time Streams." Accepted for publication in *ACM Transactions on Modelling and Computer Simulation*.

[13] Yau, V. "Automating Parallel and Distributed Quantative Stochastic Simulation." Technical Report No. COSC 05/96, Sept. 1996, Dept. of Computer Science, University of Canterbury, New Zealand, pg 118.

[14] Pawlikowski, K., Yau, V. and McNickle, D. "Distributed Stochastic Discrete-Event Simulation in Parallel Time Streams." *Proceedings of the 1994 Winter Simulation Conference*, IEEE Press, pp 723-730.

[15] "ISO/IEC ANSI/IEEE Standard 802.6 LAN/MAN Standards DQDB Access Method Package." IEEE, ISBN 1559375760, 1994.

[16] Atkins, J. and Norris, M. *Total Area Networking: ATM, IP, Frame Relay, and SMDS Explained*. John Wiley and Sons, ISBN 0471984647, 1999.

[17] Huang, N.-F. and Liu, H.-I. "A Study of Isochronous Channel Reuse in DQD13 Metropolitan Area Networks." *IEEE/ACM Transactions on Networkin*g, Vol. 6, No. 4, pp 475-484, 1998.

[18] Yang, Y., Lai, T.-H. and Liu, M.-T. "Isonchronous Bandwidth Utilization Improvement in Distributed Queue Dual Bus-Based Personal Communicatin Networks.'' *Computer Communications*, Vol. 21, pp 1420-1433, 1998.

[19] Sharon, O. "A Proof for Lack of Starvation in DQDB with and without Slot Reuse." *IEEE/ACM Transactions on Networking*, Vol. 5, No. 3, pp 410-419, 1997.

[20] Hairong, S., Ke, H. and Lemin, L. "Performance Analysis of the Constant Bit-rate Services in an ATM DQDB MAN." *Computer Communications*, Vol. 21, pp 186-194, 1998.

[21] Dong, X. and Lai, T. "An Efficient Protocol for Call Setup and Path Migration in IEEE 802.6 based Personal Communication Networks." *IEEE Transactions on Computers*, Vol. 46, No. 3, pp 252-259, 1997.

[22] Orozco-Barbosa, L. and Ahmed, N.U. "Dynamic Modelling of the IEEE 802.6 Medium Access Mechanism." *Proceedings of the 18th Biennial Symposium on Communication*, pp 21-24, 1996.

[23] Wu, J.S. and Hsieh, Y.T. "Improving Access Delay Fairness of DQDB Mans under Overload Condition." *Computer Communications*, Vol. 19, pp 571-579, 1996.

[24] Yau, V. and Pawlikowski, K. "Analysis of Dual Bus Metropolitan Area Networks Using Distributed Quantitative Stochastic Simulation." *SIMULATION,* Vol. 75, No. 3, September 2000.

[25] Gafarian, A.V., Ancker, C.J. and Morisaku T. "Evaluation of Commonly Used Rules for Detecting Steady State in Computer Simulataion." *Naval Research Logistics Quarterly*, Vol. 8, pp 511-529, 1978.

[26] Schruben, L.W., Singh, H. and Tierney L. "Optomal Tests for Initialisation Bias In Simulation Output." *Operations Research*, Vol. 31, pp 1167-1178, 1983.

Victor Yau was born in Hong Kong, China. He received his BSc degree from the University of Otago, New Zealand, in 1987. In 1988 and 1989, he worked as a Systems Analyst with the Information Technology Service Center, Wellington, New Zealand. He received a PostGraduate Diploma in Applied Science from Victoria University of Wellington in 1990, and a PhD degree from the University of Canterbury, Christchurch, New Zealand, in 1996, both in Computer Science. The work that Dr. Yau reports in this paper was performed in part while he was working with the German National Research Center for Information Technology, GMD FOKUS, in Berlin, Germany. Currently he is a Visiting Associate Professor in the Department of Computer Science and Engineering at the University of Texas at Arlington. Dr. Yau's research interests include simulation, communication systems, and distributed computing.

Krzysztof Pawlikowski is an Associate Professor (Reader) in Computer Science at University of Canterbury, Christchurch, New Zealand. He received his PhD in Computer Engineering from the Technical University of Gdansk in Poland. Dr. Pawlikowski is the author of more than 90 research papers and four books. His research interests include stochastic simulation, cluster processing, performance modelling of telecommunication networks (Internet, ATM, optical and wireless technology) and teletraffic modelling. He is a Senior Member of IEEE. More information is available at *www.cosc.canterbury.ac.nz/~krys*.