Distrib. Syst. Engng **3** (1996) 20–31. Printed in the UK

Performance analysis of parallel simulation on distributed systems

Yong Meng Teo† and Seng Chuan Tay‡

Department of Information Systems & Computer Science, National University of Singapore, Lower Kent Ridge Road, Singapore 119260

Abstract. This paper presents an analytical model to evaluate the performance of parallel simulation on distributed computing platforms. The proposed model is formalized by two important time components in parallel and distributed processing: *computation time* and *communication time*. A conservative parallel simulation of multistage interconnection networks is used as an example in our analytical model. Performance metrics such as *elapsed time*, *speedup* and *simulation bandwidth* associated with different schemes for partitioning/mapping parallel simulation onto distributed processors are evaluated. Our mathematical analysis identifies the major constituents of simulation overheads in these mapping strategies necessary for improving parallel simulation efficiency. We also show that a perfectly balanced workload distribution may not necessarily translate into better performance. On the contrary, we have shown that a balanced mapping of workload may increase communication overheads resulting in a longer simulation elapsed time. Our performance model has been validated against implementation results from a parallel simulation model. The analytical framework is also practical to evaluate the runtime efficiency of other simulation applications which are based on the conservative paradigm.

1. Introduction

By the use of multiprocessors, the parallel discrete-event simulation (PDES) technique offers potential both for speeding up the sequential discrete-event simulation, and most importantly for increasing the size and complexity of models simulatable within a reasonable amount of time. Two related paradigms, *conservative approach* and *optimistic approach*, have been widely discussed [4, 5, 7, 8]. A similarity of these two approaches lies in the process-oriented methodology in partitioning a system to be simulated into loosely coupled components and simulating each component by a process. Interaction among the processes is performed by passing timestamped messages, which usually represent events scheduled, or to be scheduled if feasible, by one process at another. To ensure causality correctness, the conservative approach executes safe events only, meaning that an event, say of occurrence time τ, is selected for execution only when all the events before τ are already executed. On the other hand, the optimistic approach, also called timewarp, executes the events greedily. When causality error occurs, a rollback mechanism annuls those events simulated ahead of time.

Many factors contribute to the performance of a PDES program. Active research areas within this arena include new protocols, mathematical performance analysis, time parallelism, hardware support, load balancing and dynamic memory management [3, 9]. Among them performance analysis provides an insight into the simulation modelling and identifies the cause of unsatisfactory implementations. The work done in this area is useful to simulationists as it provides the guidance necessary to improve, or even redesign, their programs. This paper focuses on the performance modelling of the conservative parallel simulation. Instead of using self-contrived queuing network models, our mathematical analysis is based on realistic multistage interconnection networks (MIN). The proposed simulation and performance models can be practically applied to study other simulation applications which are based on the conservative paradigm. The remainder of this paper is organized as follows. Section 2 gives an overview of the conservative parallel simulation mechanism for modelling and simulating interconnection networks. We discuss the hand-shaking protocols used to ensure the correctness of finite-buffered simulation, and highlight some resolutions to tackle deadlock/livelock problems and a flushing mechanism to resolve combinatoric memory explosion. Section 3 abstracts the performance model. We derive the individual process elapsed time, and analyse the speedup and simulation bandwidth under light traffic and heavy traffic conditions. Section 4 analyses the performance of different partitioning schemes for mapping a simulation application onto a finite number of processors. Section 5 compares the derived metrics from the proposed analytical performance model with results collected from the simulation model. We comment in particular on the effect of load balancing on simulation performance. Based on the analytical results, section 6 discusses possible

† E-mail: teoym@iscs.nus.sg
‡ E-mail: taysengc@iscs.nus.sg

0967-1846/96/010020+12$19.50

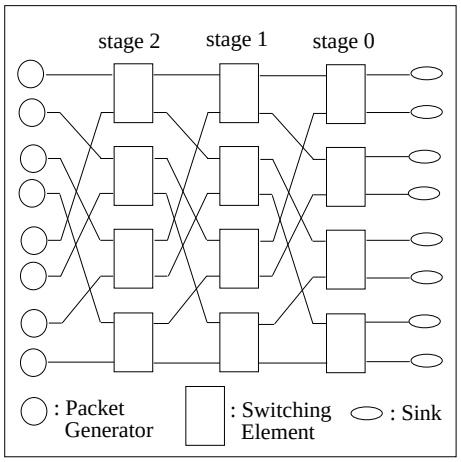


Figure 1. Components of MIN simulation.

runtime improvements. Lastly, section 7 contains our concluding remarks.

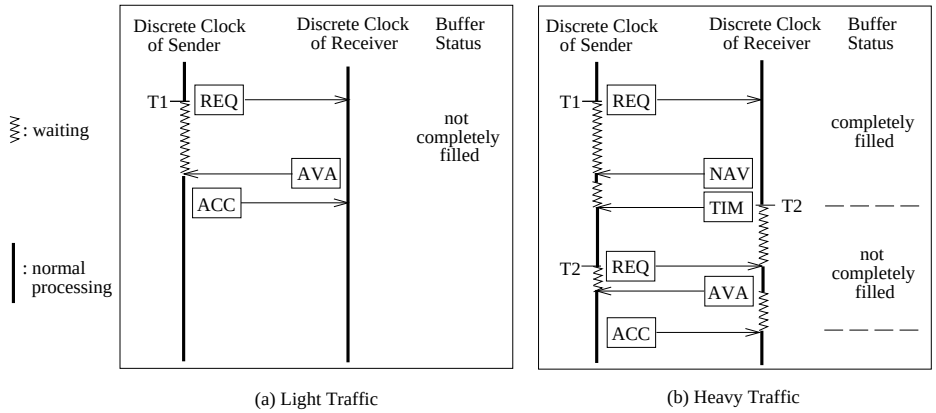
2. Conservative parallel simulation model overview

A detailed description of the MIN simulation is available in [10, 11]. In brief, the process-oriented parallel simulation model consists of three types of logical processes: *packet generator* (*G*), *switching element* (*SW*) and *sink* (*S*) (see figure 1). Based on the statistical distributions for inter-arrival time and packet destination, each *G* process generates access request packets and sends them to the *SW* process on its link. Each *SW* process serves as a relaying agent to forward the packets to their destinations. Two buffers of finite size are embedded in each *SW* process to model a blocking switch. Two types of simulation events are used: *arrival* and *departure*. A constant switch delay is used in each *SW* process. Each *S* process serves as a destination for access packets and its service time is assumed to be instantaneous. Each process has its own local clock to indicate its simulation progress. A blocked message due to the unavailability of free space in its receiving buffer will have to wait until the buffer space become available.

The simulation of a *finite-buffered* interconnection network is more sophisticated as compared to that of an infinite buffered network. For example, the execution of each departure event in *G* and *SW* processes should not send the departing packet to its successor unless the receiving buffer contains at least one unoccupied slot. Such a constraint causes some complications in the simulation as the buffer status of the receiver is not known to the sender. To ensure that packets are not lost at a *SW* process when the buffer is completely occupied, the simulation processes must adhere to some *hand-shaking protocols* during transmission [10]. That is, before a packet transmission is initiated, the sender must find out if the receiving buffer contains at least one unfilled slot to accommodate the packet. Six signals/messages are used in packet transmission. The underlined terms shown below indicate that the signal/message needs to be time-stamped: *Request Progress* (REQ), *Slot Available* (AVA), *Slot Not Available* (NAV), *Access Request* (ACC), *Transmit Time* (TIM), *Stop* (STO). Figures 2(a) and 2(b) illustrate the hand-shaking protocols used in light traffic and heavy traffic conditions respectively.

The use of hand-shaking protocols during packet transmissions can result in deadlock when the processes happen to be waiting in a loop. A detailed description of *relaying protocols* to resolve the problem is presented in [12]. In general, a process on receiving a null message or requesting a buffer status should also send null messages, with lookahead timestamps if available, to all the other three links. In this way the null messages are circulated among the processes for MIN simulation. If a received null message indicator is greater than all other outstanding event times, a process is safe to proceed on with the simulation, thereby resolving a deadlock.

Although the circulation of null messages is able to resolve the deadlock problem, they can cause some livelock situations where null messages are generated progressively on a circuit. The way we break the livelock is by giving a higher priority to the execution of departure event whenever a departure time is equal to a null-message indicator. By using this priority scheme in the event scheduling, a process



(a) Light Traffic (b) Heavy Traffic

Figure 2. Hand-shaking protocols for packet transmission.

Table 1. Performance model parameters.

Parameter	Description
t	Duration of simulation
λ	Arrival rate used in generator
λ_{eff}	Effective arrival rate used in each generator
μ	Service rate used in switching element
g	Total number of packets sent per generator ($\lambda_{eff} \times t$)
n	Number of inputs/outputs of MIN, i.e. network size
p	Number of processors
T_{event}	Event (arrival or departure) execution time
T_{buffer}	Buffer (receive or transmit) access time
$T_{transit}$	Packet/message/signal transmission time
$T_{generate}$	Packet generation time
$T_{account}$	Packet accounting time
T^{G}_{total}	Total elapsed time of packet generator process
T^{SW}_{total}	Total elapsed time of switching element process
T^{S}_{total}	Total elapsed time of sink process
T_1	Total elapsed time when one processor is used to execute simulation processes
T_p	Total elapsed time when p processors are used to execute simulation processes
T_{first_p}	Total elapsed time incurred by the leftmost processor
T_{travel}	Travelling time for the last transmission to reach its destination
$S_p(scheme)$	Speedup metric when p processors are used for a particular mapping scheme
$BW_p(scheme)$	Bandwidth (number of events simulated per unit time) for a particular mapping scheme

will proceed with the simulation instead of ceaselessly relaying null messages to other processes.

A flushing mechanism is also proposed in [12] to handle the combinatoric explosion in memory utilization. That is, when a null message is received, the process will flush in all the null messages that have already arrived on that link and select the *largest* time indicator. The forwarding of null messages is only on the largest time indicator and the rest are discarded. Such a flushing mechanism is also able to maintain the liveness of simulation as at least one null message is circulating in the network throughout the simulation duration.

3. Analytical performance model

Table 1 contains a list of performance parameters used in our analysis. Based on problem size, MIN's characteristics, data transmission protocols and simulation mechanism, the model formalizes the workload of each simulation component in terms of their elapsed time.

We assume that the destination of each packet transmission is uniformly distributed. Each data transmission among the simulation processes is modelled by two components: *buffer access time* and *transit time*. The reception of data is modelled by buffer access time only and transit time is excluded to prevent double accounting. We divide the analysis of simulation performance into two parts: *light traffic* ($\lambda \leq \mu$) and *heavy traffic* ($\lambda > \mu$) conditions. In the speedup analysis and simulation bandwidth analysis, we assume that $p = 2 \times n + \frac{n \times \log_2 n}{2}$, i.e., the number of processors is sufficient to achieve the maximum degree of parallelism for the MIN simulation. While modelling T_p, we assume that the elapsed time of each SW process outweighs those of G and S processes.

3.1. Light traffic

In this case we let $\lambda_{eff} = \lambda$. We assume that the hand-shaking protocols used in each packet transmission include only REQ and AVA signals. In other words, when a process sends a REQ signal to its receiver to find out the buffer status, the reply is always 'available' (see figure 2(a)). It follows that the time incurred in each set of hand-shaking protocols is $2 \times T_{buffer} + T_{transit}$, and the time incurred in each packet transmission is $T_{buffer} + T_{transit}$.

3.1.1. Packet generator The elapsed time of each G process is modelled by three components: *packet generation* (T^G_1), *hand-shaking protocols* (T^G_2) and *packet transmission* (T^G_3). Thus,

$$T^G_1 = g \times T_{generate} \quad (1)$$

$$T^G_2 = g \times (2T_{buffer} + T_{transit}) \quad (2)$$

$$T^G_3 = g \times (T_{buffer} + T_{transit}). \quad (3)$$

By equations (1), (2) and (3), the total elapsed time of each G process becomes

$$T^G_{total} = g \times (T_{generate} + 3T_{buffer} + 2T_{transit}). \quad (4)$$

3.1.2. Switching element We model the total elapsed time of each SW process by four components: *event execution* (T^{SW}_1), *hand-shaking protocols* (T^{SW}_2), *packet transmission and reception* (T^{SW}_3) and *null-message transmissions* (T^{SW}_4).

As there are two input links to each SW process, the number of received packets is $2 \times g$. Each packet, corresponding to one arrival event, will generate one departure event and therefore the total number of events to be executed in a SW process is $2 \times 2g$. It follows that

$$T^{SW}_1 = 4g \times T_{event} \quad (5)$$

$$T_2^{SW} = 2g \times (2T_{buffer} + T_{transit}) + 2g \times (2T_{buffer} + T_{transit}) = 4g \times (2T_{buffer} + T_{transit}). \quad (6)$$

For each SW process, $2g$ packets are received and $2g$ packets are transmitted. Therefore

$$T_3^{SW} = 2g \times T_{buffer} + 2g \times (T_{buffer} + T_{transit}) = 4g \times T_{buffer} + 2g \times T_{transit}. \quad (7)$$

For each transmission of AVA signal and REQ signal and ACC message on an in-coming link and out-going link respectively, null messages will be sent on the other three links of the SW process to prevent deadlock. We assume that the null messages are then discarded by their receivers. Since there are $2g$ packets to be received and $2g$ packets to be transmitted in each SW process, we have

$$T_4^{SW} = 2g \times 1 \times 3 \times (T_{buffer} + T_{transit}) + 2g \times 2 \times 3 \times (T_{buffer} + T_{transit}) = 18g \times (T_{buffer} + T_{transit}). \quad (8)$$

By equations (5), (6), (7) and (8), the total elapsed time of each SW process becomes

$$T_{total}^{SW} = 2g \times (2T_{event} + 15T_{buffer} + 12T_{transit}). \quad (9)$$

3.1.3. Sink The elapsed time of each S process is modelled by two components: *packet accounting* (T_1^S) and *hand-shaking protocols* (T_2^S). Since a S process is connected to each output link of the MIN, the number of received packets is g. It follows that

$$T_1^S = g \times T_{account} \quad (10)$$

$$T_2^S = g \times (2T_{buffer} + T_{transit}). \quad (11)$$

By equations (10) and (11), the total elapsed time of each S process becomes

$$T_{total}^S = g \times (T_{account} + 2T_{buffer} + T_{transit}). \quad (12)$$

3.1.4. Speedup Speedup for p processors is defined as the execution time for the best serial algorithm in a single processor (T_1) divided by the execution time for the parallel algorithm using p processors (T_p) [2]. In our analysis, T_1 is modelled by the elapsed time of MIN simulation incurred by a sequential program. As causality correctness is easily ensured in sequential simulation, the use of transmission protocols is not necessary in the simulation program and is discarded in our derivation of T_1. This presents a fairer speedup measure than sometimes reported in the literature whereby the value of T_1 is measured by running the parallel algorithm on one processor. This later instance certainly gives better speedup. Given that the number of simulated packets transmitted across the MIN is $n \times g$, the elapsed time of the simulation program becomes

$$T_1 = n \times g \times T_{generate} + \frac{n \log_2 n}{2} \times 2g \times 2T_{event} + n \times g \times T_{account} = ng \times (T_{generate} + 2\log_2 n \times T_{event} + T_{account}). \quad (13)$$

Assuming that the elapsed time of each SW process outweighs those of G and S processes, we have

$$T_p = T_{total}^{SW} = 2g \times (2T_{event} + 15T_{buffer} + 12T_{transit}). \quad (14)$$

By equations (13) and (14), under light traffic condition the speedup when p processors (where $p = 2n + \frac{1}{2}n\log_2 n$) are used becomes

$$S_p(nil) = \frac{ng \times (T_{generate} + 2\log_2 n \times T_{event} + T_{account})}{2g \times (2T_{event} + 15T_{buffer} + 12T_{transit})} \approx \frac{p}{1 + \frac{15T_{buffer} + 12T_{transit}}{2T_{event}}} \quad (15)$$

3.1.5. Simulation bandwidth Simulation bandwidth, a measure of simulation implementation efficiency, is defined as the number of events executed per second. The total number of events to be executed is

$$2 \times 2g \times \frac{n \times \log_2 n}{2}. \quad (16)$$

By equations (14) and (16), the simulation bandwidth under light traffic condition becomes

$$BW_p(nil) \approx \frac{p}{T_{event} + \frac{15}{2}T_{buffer} + 6T_{transit}}. \quad (17)$$

3.2. Heavy traffic

When the arrival rate is greater than the service rate of the SW processes connected to the G processes, the buffer spaces of SW processes will be filled up progressively until all slots are occupied. Subsequently, the packet arrival rate is moderated by the SW processes to prevent packet loss. When the steady stage is reached $\lambda_{eff} = \mu$. Our worst case analysis assumes that the time interval before the steady stage is negligible. The hand-shaking protocols used by generators for packet transmission include REQ, NAV and AVA signals and TIM message to delay the packet arrival times (see figure 2(b)). The derivations of the performance metrics are available in the appendix, and we summarize the results in tables 2(a) and 2(b).

Table 2. Analytical results for $p = 2n + \frac{1}{2}n\log_2 n$. (a) Elapsed time and (b) asymptotic speedup and simulation bandwidth.

(a)

Traffic	Elapsed time
Light	$2g \times (2T_{event} + 15T_{buffer} + 12T_{transit})$
Heavy	$2g \times (2T_{event} + 21T_{buffer} + 17T_{transit})$

(b)

Traffic	Speedup	Simulation bandwidth
Light	$\frac{p}{1+\frac{15T_{buffer}+12T_{transit}}{2T_{event}}}$	$\frac{p}{T_{event}+\frac{15}{2}T_{buffer}+6T_{transit}}$
Heavy	$\frac{p}{1+\frac{21T_{buffer}+17T_{transit}}{2T_{event}}}$	$\frac{p}{T_{event}+\frac{21}{2}T_{buffer}+\frac{17}{2}T_{transit}}$

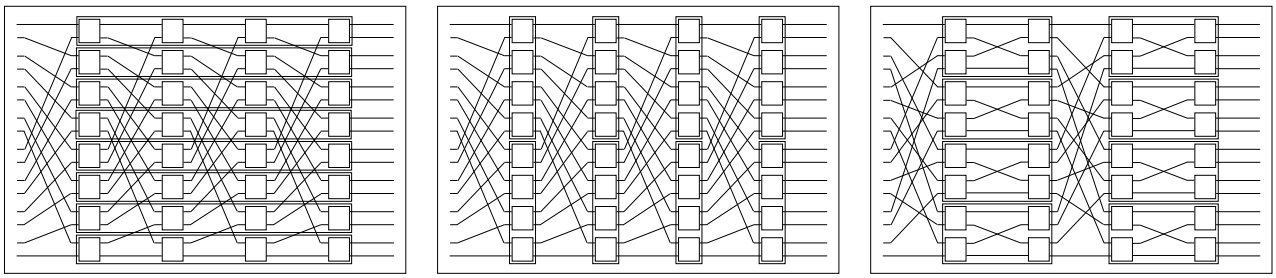


Figure 3. Mapping schemes for a finite number of workstations. (a) Horizontal partition, (b) vertical partition and (c) modular partition.

4. Analysis for different mapping schemes

Our conservative simulation model requires n G processes, $\frac{1}{2}n\log_2 n$ SW processes and n S processes to simulate a $n \times n$ MIN. Mapping of such processes onto a particular processors configuration for efficient implementation is an NP-complete problem [1]. In this paper, we introduce and analyse the performance of *three mapping schemes*, namely, the *horizontal partitioning scheme (HPS)*, the *vertical partitioning scheme (VPS)* and the *modular partitioning scheme (MPS)*. Details of the formulation and mathematical transformation used can be found in [10, 12]. For partitioning a 16×16 MIN simulation shown in figures 3(a–c), each rectangular block containing four switching elements is mapped onto *one* processor. As the G and S processes do not have to simulate the switch operations, their workloads are comparatively smaller as compared to that of a SW process. Therefore, in all mapping schemes each G process is placed together with the SW process on its out-going link in the same partition, and each S process is placed together with the SW process on its in-coming link. It is noteworthy that in MPS each module contains *four* SW processes where each process corresponding to a 2×2 switch, is executed/mapped on the same processor. However, in VPS each partition containing $\log_2 n$ SW *processes* is executed on a processor. Therefore, in the following speedup and simulation bandwidth comparison analysis we let $n = 2^{2^i}$ for some integer i. For a fairer comparison, we let $p = n/2$ so that the number of SW processes mapped onto a processor is the same in all three schemes. Intra-processor communication time incurred by processes is assumed to be negligible. The following elapsed time analysis, speedup analysis and simulation bandwidth analysis is based on light traffic condition. The performance metrics for the heavy traffic condition can be derived by the same steps.

4.1. Horizontal partitioning scheme

In HPS, the G, SW and S simulated processes on one horizontal rectangular box (see figure 3(a)) are mapped onto one processor. Parallel simulation is completed only when the sink processes in each processor have received the last packet transmission. Therefore, T_p is modelled by the maximum workload among processors. Before proceeding further, we re-compute the elapsed times of the affected processes due to this scheme.

4.1.1. Packet generator As the intra-processor transit time between the generator process and the SW process on the $(\log_2 n - 1)$th stage is negligible, the total elapsed time for each G process (see equation (4)) becomes

$$\begin{aligned} T^G_{total} &= g \times (T_{generate} + 3T_{buffer} + 2 \times 0) \\ &= g \times (T_{generate} + 3T_{buffer}). \end{aligned} \tag{18}$$

4.1.2. SW process on the $(\log_2 n - 1)$th stage While modelling the maximum workload among processors, we do not consider the top and the bottom partitions due to the diminished inter-processor links which reduce the total elapsed time. The elapsed times due to event execution (equation (5)) and packet transmission and reception (equation (7)) remain unchanged. The elapsed time due to hand-shaking protocols (equation (6)) for the SW processes on the $(\log_2 n - 1)$th stage is reduced to

$$\begin{aligned} T^{SW}_2 &= 2g \times (2T_{buffer} + 0) + 2g \times (2T_{buffer} + T_{transit}) \\ &= 2g \times (4T_{buffer} + T_{transit}). \end{aligned} \tag{19}$$

For the same reason, the elapsed time due to null-message transmission (equation (8)) is reduced to

$$\begin{aligned} T^{SW}_4 &= 2g \times 1 \times 3 \times (T_{buffer} + 0) \\ &\quad + 2g \times 2 \times 3 \times (T_{buffer} + T_{transit}) \\ &= 6g \times (3T_{buffer} + 2T_{transit}). \end{aligned} \tag{20}$$

By equations (5), (7), (19) and (20), the total elapsed time of each SW process on the $(\log_2 n - 1)$th stage becomes

$$T^{SW}_{total} = 2g \times (2T_{event} + 15T_{buffer} + 8T_{transit}). \tag{21}$$

4.1.3. SW process on the 0th stage The elapsed time due to event execution (equation (5)) remains unchanged. The elapsed time due to hand-shaking protocols (equation (6)) is reduced to

$$\begin{aligned} T^{SW}_2 &= 2g \times (2T_{buffer} + T_{transit}) + 2g \times (2T_{buffer} + 0) \\ &= 2g \times (4T_{buffer} + T_{transit}). \end{aligned} \tag{22}$$

The elapsed time due to packet transmission and reception (equation (7)) becomes

$$\begin{aligned} T^{SW}_3 &= 2g \times T_{buffer} + 2g \times (T_{buffer} + 0) \\ &= 4g \times T_{buffer}. \end{aligned} \tag{23}$$

Table 3. Analytical results for $p = \frac{n}{2}$. (a) Elapsed time, (b) asymptotic speedup and simulation bandwidth.

(a)

Scheme	Elapsed time
HPS	$2g \times (T_{generate} + 2\log_2 n \times T_{event} + T_{account} + (5 + 15\log_2 n) \times T_{buffer} + 2(6\log_2 n - 5) \times T_{transit})$
VPS	$2g\log_2 n \times T_{generate} + 2(\log_2 n \times (2g+1) - 1)T_{event} + T_{account} + (3\log_2 n \times (12g+5) - 13)T_{buffer} + 2(2\log_2 n \times (4g+3) - 9)T_{transit}$
MPS	$g\log_2 n \times T_{generate} + 2((2g+1)\log_2 n - 2) \times T_{event} + T_{account} + (3(11g+5)\log_2 n - 28) \times T_{buffer} + ((8g+7)\log_2 n - 22) \times T_{transit}$

(b)

Scheme	Speedup	Simulation bandwidth
HPS	$\frac{p}{1+\frac{15T_{buffer}+12T_{transit}}{2T_{event}}}$	$\frac{2p}{2T_{event}+15T_{buffer}+12T_{transit}}$
VPS	$\frac{p}{1+\frac{T_{generate}+18T_{buffer}+8T_{transit}}{2T_{event}}}$	$\frac{2p}{T_{generate}+2T_{event}+18T_{buffer}+8T_{transit}}$
MPS	$\frac{p}{1+\frac{\frac{T_{generate}}{2}+\frac{33}{2}T_{buffer}+4T_{transit}}{2T_{event}}}$	$\frac{2p}{\frac{T_{generate}}{2}+2T_{event}+\frac{33}{2}T_{buffer}+4T_{transit}}$

The elapsed time due to null-message transmission for the SW processes (equation (8)) is reduced to

$$\begin{aligned} T_4^{SW} &= 2g \times 1 \times (3T_{buffer} + T_{transit}) \\ &\quad + 2g \times 2 \times (3T_{buffer} + 2T_{transit}) \\ &= 2g \times (9T_{buffer} + 5T_{transit}). \end{aligned} \tag{24}$$

By equations (5), (22), (23) and (24), the total elapsed time of each SW process on the 0th stage becomes

$$T_{total}^{SW} = 2g \times (2T_{event} + 15T_{buffer} + 6T_{transit}). \tag{25}$$

4.1.4. Sink process The elapsed time of each S process (see equation (12)) is reduced to

$$T_{total}^{S} = g \times (T_{account} + 2T_{buffer}). \tag{26}$$

By multiplying the respective numbers of processes to equations (9), (18), (21), (25) and (26), we have

$$\begin{aligned} T_p &= 2 \times g \times (T_{generate} + 3T_{buffer}) \\ &\quad + 1 \times 2g \times (2T_{event} + 15T_{buffer} + 8T_{transit}) \\ &\quad + (\log_2 n - 2) \times 2g \\ &\quad \times (2T_{event} + 15T_{buffer} + 12T_{transit}) \\ &\quad + 1 \times 2g \times (2T_{event} + 15T_{buffer} + 6T_{transit}) \\ &\quad + 2 \times g \times (T_{account} + 2T_{buffer}) \\ &= 2g \times (T_{generate} + 2\log_2 n \times T_{event} + T_{account} \\ &\quad + (5 + 15\log_2 n) \times T_{buffer} \\ &\quad + 2(6\log_2 n - 5) \times T_{transit}). \end{aligned} \tag{27}$$

4.1.5. Speedup By equations (13) and (27), the speedup when p processors (where $p = n/2$) are used becomes

$$\begin{aligned} S_p(HPS) &\approx ng \times 2\log_2 n \times T_{event}[2g \times (2\log_2 n \\ &\quad \times T_{event} + 15\log_2 n \times T_{buffer} \\ &\quad + 12\log_2 n \times T_{transit})]^{-1} \\ &\approx \frac{p}{1 + \frac{15T_{buffer}+12T_{transit}}{2T_{event}}}. \end{aligned} \tag{28}$$

4.1.6. Simulation bandwidth By equations (16) and (27), the simulation bandwidth of HPS becomes

$$\begin{aligned} BW_p(HPS) &\approx 2gn\ \log_2 n[2g \times (2\log_2 n \\ &\quad \times T_{event} + 15\log_2 n \times T_{buffer} \\ &\quad + 12\log_2 n \times T_{transit})]^{-1} \\ &\approx \frac{2p}{2T_{event} + 15T_{buffer} + 12T_{transit}}. \end{aligned} \tag{29}$$

4.2. Vertical and modular partitioning schemes

In VPS, each processor executes $\log_2 n$ SW processes belonging to the same MIN stage (see figure 3(b)). During simulation, ACC packets are created by the generators, relayed by SW processes in each stage, and absorbed by the sinks. As the mapping is performed on a stage basis, parallel simulation is completed only when the last (rightmost) processor terminates. T_p is therefore modelled by two components: *elapsed time of the first (leftmost) processor*, and the *travelling time for the last transmission to reach the sink process*.

The MPS is based on partitions of *four SW processes* spread over two MIN stages as shown in figure 3(c). A proof of the transformation from the Omega MIN to its equivalent modular topology can be found in [11]. Each processor executes $\log_2 n$ SW processes arranged in $\frac{1}{4}\log_2 n$ blocks. For the same reason that sink processes are contained in the rightmost module only, we also model T_p by the same two components as in VPS.

A summary of the analytical results for VPS and MPS is given in tables 3(a) and (b).

5. Model validation and performance analysis

We implemented the conservative parallel MIN simulation model in C language on a network of workstations which mimics a distributed-memory parallel computer. PVM software [6] was used for spawning the simulation processes and for handling message passing. The

Table 4. Performance parameter measurement.

Parameter	Time (μs)
$T_{generate}$	926
T_{event}	2990
$T_{account}$	479
T_{buffer}	3238
$T_{transit}$	1821

simulation model results presented are based on simulating a 16×16 Omega MIN on a network of *eight* homogeneous SUN workstations connected using a 10 Mbps Ethernet network. Altogether 64 PVM processes are spawned on the processors.

Five parameter values used in the analytical model are obtained by taking measurements on the workstation network (see table 4). Additional statistical measures used in the analysis are defined. Let s_i be a simulation process and $|s_i|$ denote its workload. In the following analysis, the workload for each simulation process is represented by the size of its object codes, and normalized to that of the S process. For simplicity, we let $|S| = 1$. Thus, the workloads for $|G|$ and $|SW|$ are 1.93 and 6.24 respectively. The normalized coefficient of $T_{transit}$ is denoted by $\mathcal{T}$. The percentage deviation between the measured and the predicted values δ is computed as $\frac{|measured-predicted|}{predicted} \times 100$. The average of δ over six different sets of simulation parameters is denoted by $\overline{\delta}$.

Let M be a set containing all s_i required in a MIN simulation. Let $\widehat{P}$ be the processor that incurs the largest workload among all processors. We define the *workload distribution coefficient* (ω) as $\omega = \sum_{s_i \in M} |s_i| / \sum_{s_j \in \widehat{P}} |s_j|$. The load balancing factor (η) is then defined as $\eta = \frac{p}{\omega}$. Thus, the larger the value of η, the more unbalanced is the workload distribution. The smallest (best) value of η equals 1, corresponding to a perfectly balanced distribution of workload.

We first compare the elapsed times among the different schemes, and between the simulation model *(measured)* and the analytical model *(predicted)*. For VPS and MPS, the latter outperforms in terms of elapsed time as shown in table 5. This phenomenon is due to the poor workload distribution and larger interprocessor communication overheads of VPS ($\mathcal{T}$ = 2) compared to those of MPS ($\mathcal{T}$ = 1). The comparison of VPS with HPS is more interesting. Although the interprocessor communication overhead of VPS is only half of that of HPS, it seems that the imbalance workload factor is more dominant, making the elapsed time of VPS (η = 1.31) larger than that of HPS (η = 1). In this aspect, our explanation is that for VPS the worst number of processes in the processors is greater than that of HPS. Therefore, the elapsed time of VPS consists of a larger amount of context switching time incurred by the task scheduler of the UNIX operating system. The comparison of HPS (η = 1) and MPS (η = 1.06) is rather counter-intuitive. We observe that the HPS, a perfect load balancing scheme, incurs a significantly larger elapsed time as compared to that of a less balanced scheme. Such an observation contradicts the common belief that the workload distribution among processors must be balanced in order to improve elapsed time. We note here that other factors such as inter-processor communication overheads incurred by a mapping scheme should also be taken into account while balancing the workload distribution. As the inter-processor communication overhead incurred by HPS is larger than that of MPS, its elapsed time is therefore aggravated despite the balanced workload distribution.

The averaged percentage deviations $\overline{\delta}$ show that the accuracy of the analytical model for predicting elapsed time is good for all the three mapping schemes. However, the predicted timings show slightly better performance than the measured simulation model timings because the context switching time incurred by the scheduler of the UNIX OS is not accounted for in our analytical model. Tables 6 and 7 show that our performance model closely predicts the parallel simulation speedups and simulation bandwidths. The predicted values for all three mapping schemes are slightly better than the measured values for the same reason. The speedup efficiency is unacceptably low, i.e. around 10%. Our analysis reveals that this is due to the higher than expected cost of PVM communication. Better speedup can be obtained on a parallel machine where communication is much more efficient and less costly. In the following section, we present a breakdown of various communication costs, and analyse the effect of reducing these overheads on simulation performance.

In summary, the measured and predicted values for elapsed time, speedup and simulation bandwidth show the following important findings:

- A perfectly balanced workload distribution, such as HPS, may not necessarily translate into better performance.

Table 5. Comparison of elapsed times (in seconds).

Scheme		No. of messages per generator						$\overline{\delta}$ (%)	$\mathcal{T}$	η
		120	240	360	480	600	720			
HPS	Measured	76.79	151.15	226.49	304.09	377.68	454.93	3.66	3	1.00
	Predicted	*73.19*	*146.38*	*219.60*	*292.76*	*365.95*	*439.14*	—	—	—
VPS	Measured	79.61	160.14	239.93	322.23	403.43	482.46	4.78	2	1.31
	Predicted	*76.75*	*153.28*	*229.81*	*306.33*	*382.86*	*459.39*	—	—	—
MPS	Measured	67.29	134.00	200.25	269.73	337.18	406.74	4.31	1	1.06
	Predicted	*64.58*	*129.03*	*193.48*	*257.93*	*322.38*	*386.83*	—	—	—

Table 6. Comparison of speedup.

Scheme		No. of messages per generator						$\overline{\delta}$ (%)
		120	240	360	480	600	720	
HPS	Measured	0.61	0.62	0.62	0.61	0.62	0.63	6.31
	Predicted	*0.66*	*0.66*	*0.66*	*0.66*	*0.66*	*0.66*	—
VPS	Measured	0.59	0.58	0.58	0.59	0.60	0.59	6.61
	Predicted	*0.63*	*0.63*	*0.63*	*0.63*	*0.63*	*0.63*	—
MPS	Measured	0.72	0.71	0.72	0.71	0.71	0.71	4.89
	Predicted	*0.75*	*0.75*	*0.75*	*0.75*	*0.75*	*0.75*	—

Table 7. Comparison of simulation bandwidth (events/second).

Scheme		No. of messages per generator						$\overline{\delta}$ (%)
		120	240	360	480	600	720	
HPS	Measured	200.02	203.24	203.45	202.05	203.35	202.58	2.98
	Predicted	*209.85*	*209.87*	*209.86*	*209.86*	*209.87*	*209.87*	—
VPS	Measured	191.68	191.07	192.97	191.94	191.54	192.73	4.26
	Predicted	*200.12*	*200.41*	*200.51*	*200.57*	*200.59*	*200.61*	—
MPS	Measured	228.26	229.25	230.11	227.78	227.77	226.58	4.13
	Predicted	*237.85*	*238.09*	*238.16*	*238.20*	*238.23*	*238.24*	—

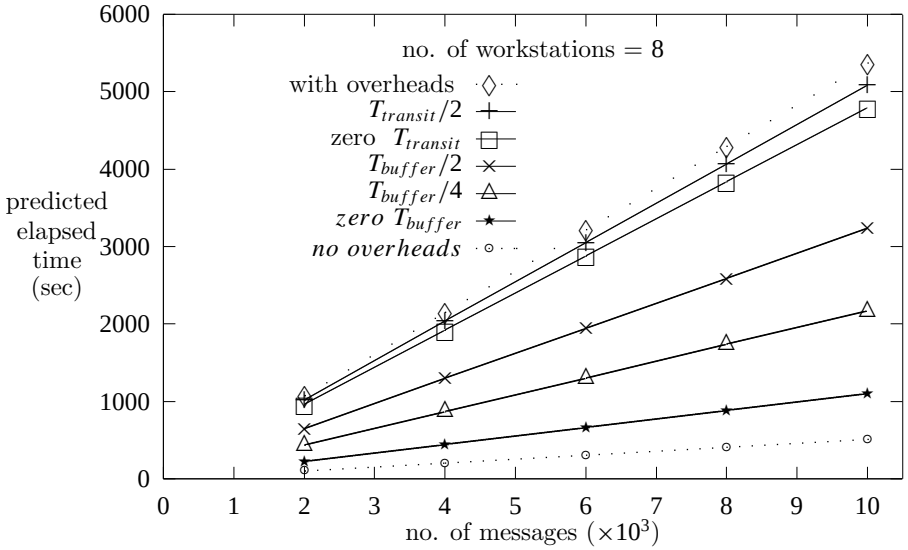


Figure 4. Elapsed time analysis (using MPS mapping scheme).

- Inter-processor communication overheads may cause dominant aggravation to the program elapsed time.
- A good mapping scheme should take into account the load balancing as well as communication overheads. Ignoring either factor may result in poor implementation performance.

6. Performance improvement

Table 8 depicts the asymptotic elapsed times for MIN simulation. The comparisons show that by transforming the MIN into a more modular equivalent interconnection, the message transit time can be significantly reduced. As also shown in our performance model, the communication overheads comprise two major components: *buffer access time* and *message transmission time*. In order to improve the implementation performance, we therefore have to reduce these two constituents.

Using the MPS as an example, we analyse how the overheads affect parallel simulation performance. We compared the implications of setting overheads to zero (*no overheads*), buffer access time to zero (*zero* T_{buffer}), message transmission time to zero (*zero* $T_{transit}$), reducing T_{buffer} to half ($T_{buffer}/2$) and to a quarter ($T_{buffer}/4$), and reducing $T_{transit}$ by half ($T_{transit}/2$).

Figures 4, 5 and 6 depict the elapsed time versus the number of messages, and the speedup and the

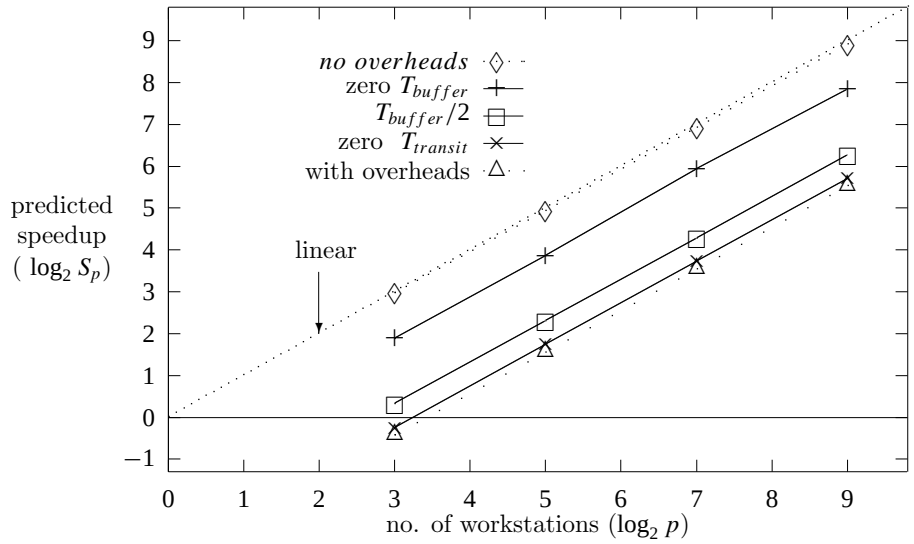


Figure 5. Speedup analysis (using MPS mapping scheme).

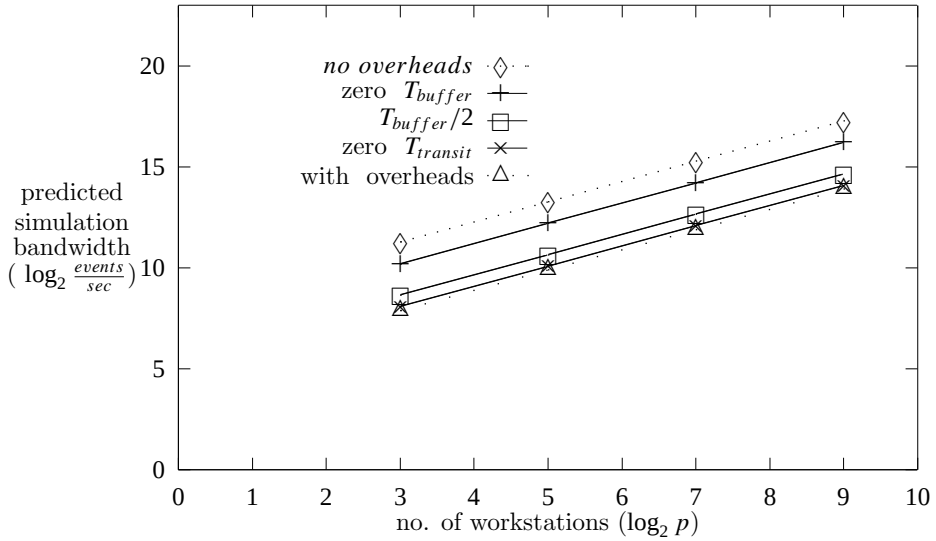


Figure 6. Simulation bandwidth analysis (using MPS mapping scheme).

Table 8. Asymptotic elapsed time for different partitioning schemes.

Scheme	Elapsed time ($\times 2g \log_2 n$)
HPS	$2T_{event} + 15T_{buffer} + 12T_{transit}$
VPS	$T_{generate} + 2T_{event} + 18T_{buffer} + 8T_{transit}$
MPS	$\frac{T_{generate}}{2} + 2T_{event} + \frac{33}{2}T_{buffer} + 4T_{transit}$

simulation bandwidth against the number of workstations respectively. In particular, figure 5 shows that good parallel simulation speedup can be achieved even with the imposed transmission protocols, provided the communication overheads are negligible in comparison with the event grain size. We observed that the worst bottleneck is attributed to T_{buffer} followed by $T_{transit}$. Thus, more effort should be devoted to reducing PVM-based memory allocation time, and improving access protocols to reduce overall message buffering time than to improving the network transmission speed. A significant reduction in elapsed time is observed when T_{buffer} is halved as compared to setting $T_{transit}$ to zero (figure 4).

7. Conclusions

We have developed an analytical model that characterizes the performance of a conservative parallel simulation model using five timing parameters. Performance measures are derived for simulation elapsed time, speedup and bandwidth for both light and heavy traffic conditions. Validation experiments comparing the performance metrics from the analytical model and the simulation model show an acceptable 5% difference. We have analysed the performance for three process-to-processor mapping

schemes to identify the main causes of poor performance. Using the analytical framework, we also evaluated the significance of reducing buffer access time and message transit time in order to improve the runtime performance. The performance model can be easily extended to analyse other applications using the same conservative parallel mechanism. Our work reveals that in the PVM-based implementation reducing buffer access time is more critical than network transmission time. In addition, we dispel the common belief that to reduce the elapsed time of a parallel program the workload distribution among the processors must be balanced. We observed that other factors such as inter-processor communication overheads may also cause dominant aggravation to the simulation performance despite the balanced workload distribution. Therefore, a balanced workload distribution may not necessarily translate into better performance. Both the analytical and implementation results confirm such an exceptional phenomenon.

Our work also shows that although PDES is intuitively promising, it requires a lot of careful implementation considerations in practice. The simulation performed on a multiprocessor platform, if not implemented properly, may produce a runtime slow down due to high simulation synchronization overheads, fine granularity of events, strong coupling of exploitable parallelism and the system to be simulated, inherent limited lookahead, etc. Instead of building new PDES synchronization and disregarding distributed system synchronization mechanisms already available, an important thread of investigation is to harmonize and integrate the synchronization requirements in organizing PDES and in distributed processing. In this aspect, much remains to be done to realize its full potential.

Appendix A. Performance metrics for heavy traffic condition

The elapsed times incurred in each set of hand-shaking protocols used by G processes for packet transmission and in the SW process on the $(n-1)$th stage for packet reception are $5T_{buffer}+2T_{transit}$ and $5T_{buffer}+3T_{transit}$ respectively (see figure 2(b)). The hand-shaking protocols used in the SW processes on all other stages remain as the light traffic condition because the arrival rate to these processes has been moderated by their service rate.

A.1. Packet generator

Packet generation time (T_1^G) and packet transmission time (T_3^G) remain unchanged. The elapsed time due to hand-shaking protocols is as follows:

$$T_2^G = g \times (5T_{buffer} + 2T_{transit}). \tag{A1}$$

By equations (1), (3) and (A1), the total elapsed time of each G process becomes

$$T_{total}^G = g \times (T_{generate} + 6T_{buffer} + 3T_{transit}). \tag{A2}$$

A.2. Switching element on the ($\log_2 n - 1$)th stage

Event execution time (T_1^{SW}) and packet transmission and reception time (T_3^{SW}) remain unchanged. The elapsed time due to hand-shaking protocols is changed as follows:

$$\begin{aligned} T_2^{SW} &= 2g \times (5T_{buffer} + 3T_{transit}) \\ &\quad + 2g \times (2T_{buffer} + T_{transit}) \\ &= 2g \times (7T_{buffer} + 4T_{transit}). \end{aligned} \tag{A3}$$

For each transmission of AVA and NAV signals and REQ signal and ACC message on an in-coming link and out-going link respectively, null messages will also be sent on the other *three* links to prevent deadlock. Again, we assume that the null messages are then discarded by their receivers. Therefore

$$\begin{aligned} T_4^{SW} &= 2g \times 2 \times 3 \times (T_{buffer} + T_{transit}) \\ &\quad + 2g \times 2 \times 3 \times (T_{buffer} + T_{transit}) \\ &= 24g \times (T_{buffer} + T_{transit}). \end{aligned} \tag{A4}$$

By equations (5), (7), (A3) and (A4), the total elapsed time of each SW process on the $(\log_2 n - 1)$th stage becomes

$$T_{total}^{SW} = 2g \times (2T_{event} + 21T_{buffer} + 17T_{transit}). \tag{A5}$$

A.3. Speedup

With heavy traffic, T_1 can be modelled by equation (13) with $g = \lambda_{eff} \times t = \mu \times t$. By equation (A5), we have

$$T_p = T_{total}^{SW} = 2g \times (2T_{event} + 21T_{buffer} + 17T_{transit}). \tag{A6}$$

By equations (13) and (A6), the speedup for p processors (where $p = 2n + \frac{1}{2} n \log_2 n$) under heavy traffic condition becomes

$$\begin{aligned} S_p(nil) &= \frac{ng \times (T_{generate} + 2\log_2 n \times T_{event} + T_{account})}{2g \times (2T_{event} + 21T_{buffer} + 17T_{transit})} \\ &\approx \frac{p}{1 + \frac{21T_{buffer} + 17T_{transit}}{2T_{event}}}. \end{aligned} \tag{A7}$$

A.4. Simulation bandwidth

By equations (16) and (A6), the simulation bandwidth for heavy traffic condition becomes

$$BW_p(nil) \approx \frac{p}{T_{event} + \frac{21}{2}T_{buffer} + \frac{17}{2}T_{transit}}. \tag{A8}$$

Appendix B. Vertical partitioning scheme

By multiplying the number of processes to equations (18) and (21), we have

$$\begin{aligned} T_{first_p} &= 2\log_2 n \times g \times (T_{generate} + 3T_{buffer}) \\ &\quad + \log_2 n \times 2g \times (2T_{event} + 15T_{buffer} + 8T_{transit}) \\ &= 2g \times \log_2 n \times (T_{generate} + 2T_{event} \\ &\quad + 18T_{buffer} + 8T_{transit}). \end{aligned} \tag{B1}$$

By multiplying the required numbers of processes to equations (9), (25) and (26), and assuming one transit packet, we have

$$\begin{aligned}T_{travel} &= (\log_2 n - 2) \times 1 \times (2T_{event} + 15T_{buffer} \\ &\quad + 12T_{transit}) + 1 \times (2T_{event} + 15T_{buffer} \\ &\quad + 6T_{transit}) + 1 \times (T_{account} + 2T_{buffer}) \\ &= 2(\log_2 n - 1)T_{event} + T_{account} \\ &\quad + (15\log_2 n - 13)T_{buffer} \\ &\quad + 6(2\log_2 n - 3)T_{transit}. \end{aligned} \tag{B2}$$

By equations (B1) and (B2), we have

$$\begin{aligned}T_p &= T_{first_p} + T_{travel} \\ &= 2g\log_2 n \times T_{generate} + 2(\log_2 n \times (2g+1) - 1) \\ &\quad \times T_{event} + T_{account} + (3\log_2 n \times (12g+5) - 13) \\ &\quad \times T_{buffer} + 2(2\log_2 n \times (4g+3) - 9)T_{transit}. \end{aligned} \tag{B3}$$

B.1. Speedup

From equations (13) and (B3), the speedup for VPS becomes

$$\begin{aligned}S_p(VPS) &\approx ng \times 2T_{event}[2gT_{generate} \\ &\quad + 4gT_{event} + 36gT_{buffer} + 16gT_{transit}]^{-1} \\ &\approx \frac{p}{1 + \frac{T_{generate} + 18T_{buffer} + 8T_{transit}}{2T_{event}}}. \end{aligned} \tag{B4}$$

B.2. Simulation bandwidth

From equations (16) and (B3), we have

$$\begin{aligned}&BW_p(VPS) \\ &\quad \approx \frac{2p}{T_{generate} + 2T_{event} + 18T_{buffer} + 8T_{transit}}. \end{aligned} \tag{B5}$$

Appendix C. Modular partitioning scheme

In the following derivation, we assume that $n \geq 16$. First, we compute the elapsed time of the leftmost block. By multiplying the number of processes to equations (4), (9) and (21), and assuming $T_{transit} = 0$ in the first two equations, we have

$$\begin{aligned}&4 \times g \times (T_{generate} + 3T_{buffer} + 0) \\ &\quad + 2 \times 2g \times (2T_{event} + 15T_{buffer} + 0) \\ &\quad + 2 \times 2g \times (2T_{event} + 15T_{buffer} + 8T_{transit}) \\ &= 4g \times (T_{generate} + 4T_{event} + 33T_{buffer} + 8T_{transit}). \end{aligned} \tag{C1}$$

By multiplying the number of modules in the leftmost processor to equation (C1), we obtain

$$\begin{aligned}T_{first_p} &= \frac{1}{4}\log_2 n \times 4g \times (T_{generate} + 4T_{event} \\ &\quad + 33T_{buffer} + 8T_{transit}) \\ &= g \times \log_2 n \times (T_{generate} + 4T_{event} \\ &\quad + 33T_{buffer} + 8T_{transit}). \end{aligned} \tag{C2}$$

Next, we compute the elapsed time of each transmission across an intermediate block. By multiplying the respective numbers of processes in equations (21) and (25), and assuming one transit packet, we have

$$\begin{aligned}&(2T_{event} + 15T_{buffer} + 6T_{transit}) \\ &\quad + (2T_{event} + 15T_{buffer} + 8T_{transit}) \\ &= 4T_{event} + 30T_{buffer} + 14T_{transit}. \end{aligned} \tag{C3}$$

The elapsed time across the rightmost block is computed by equation (25), setting $T_{transit} = 0$ in equations (9) and (12), and assuming one transit packet, giving

$$\begin{aligned}&2T_{event} + 15T_{buffer} + 6T_{transit} \\ &\quad + 2T_{event} + 15T_{buffer} + 0 \\ &\quad + T_{account} + 2T_{buffer} + 0 \\ &= 4T_{event} + T_{account} + 32T_{buffer} + 6T_{transit}. \end{aligned} \tag{C4}$$

By multiplying the respective numbers of blocks to equations (C3) and (C4), we have

$$\begin{aligned}T_{travel} &= (\tfrac{1}{2}\log_2 n - 2) \times (4T_{event} \\ &\quad + 30T_{buffer} + 14T_{transit}) \\ &\quad + 1 \times (4T_{event} + T_{account} + 32T_{buffer} + 6T_{transit}) \\ &= (2\log_2 n - 4)T_{event} + T_{account} + (15\log_2 n \\ &\quad - 28)T_{buffer} + (7\log_2 n - 22)T_{transit}. \end{aligned} \tag{C5}$$

By equations (C2) and (C5), we have

$$\begin{aligned}T_p &= T_{first_p} + T_{travel} \\ &= g\log_2 n \times T_{generate} + 2((2g+1)\log_2 n - 2) \\ &\quad \times T_{event} + T_{account} + (3(11g+5)\log_2 n - 28) \\ &\quad \times T_{buffer} + ((8g+7)\log_2 n - 22) \times T_{transit}. \end{aligned} \tag{C6}$$

C.1. Speedup

By equations (13) and (C6), the speedup for MPS becomes

$$\begin{aligned}S_p(MPS) &\approx ng \times 2T_{event}[gT_{generate} + 2(2g+1)T_{event} \\ &\quad + 3(11g+5)T_{buffer} + (8g+7)T_{transit}]^{-1} \\ &\approx \frac{p}{1 + \frac{\frac{T_{generate}}{2} + \frac{33}{2}T_{buffer} + 4T_{transit}}{2T_{event}}}. \end{aligned} \tag{C7}$$

C.2. Simulation bandwidth

By equations (16) and (27), we have

$$\begin{aligned}&BW_p(MPS) \\ &\quad \approx \frac{2p}{\frac{T_{generate}}{2} + 2T_{event} + \frac{33}{2}T_{buffer} + 4T_{transit}}. \end{aligned} \tag{C8}$$

References

[1] Bokhari S 1981 On mapping problem *IEEE Trans. Comput.* **C-30** 207–14

[2] Decegama A L 1989 *The Technology of Parallel Processing—Parallel Architectures and VLSI Hardware* vol 1 (Englewood Cliffs, NJ: Prentice-Hall)

[3] Ferscha A 1995 Parallel and distributed simulation of discrete event systems *Handbook of Parallel and Distributed Computing* (New York: McGraw-Hill)

[4] Fishwick P A 1995 *Simulation Model Design and Execution* (Englewood Cliffs, NJ: Prentice-Hall)

[5] Fujimoto R M 1990 Parallel discrete event simulation *Commun. ACM* **33** (10) 31–53

[6] Geist A, Beguelin A, Dongarra J, Jiang W, Manchek R and Sunderan V 1993 *PVM 3 User's Guide and Reference Manual* (Oak Ridge, TN: Oak Ridge National Laboratory)
[7] Jefferson D R 1985 Virtual time *ACM Trans. Program. Lang. Syst.* **7** 404–25
[8] Misra J 1986 Distributed discrete-event simulation *Comput. Surveys* **18** 39–65
[9] Nicol D and Fujimoto R 1994 Parallel simulation today *Annals of Operations Research: Simulation and Modeling* vol 53, ed O Balci (Amsterdam: Baltzer)
[10] Tay S C and Teo Y M 1994 Conservative parallel simulation of finite buffered multistage interconnection networks *Technical Report TRE6/94* Department of Information Systems and Computer Science, National University of Singapore, p 24
[11] Teo Y M and Tay S C 1994 Efficient algorithms for conservative parallel simulation of interconnection networks *Proc. Int. Symp. on Parallel Architectures, Algorithms and Networks (Japan)* (Los Alamitos, CA: IEEE Computer Society Press) pp 286–93
[12] Teo Y M and Tay S C 1995 Modeling and efficient distributed simulation of multistage interconnection network *Proc. Int. Conf. on Algorithms and Architectures for Parallel Processing (Australia)* (Los Alamitos, CA: IEEE Computer Society Press) pp 83–92
[13] Williams R D 1991 Performance of dynamic load balancing algorithms for unstructured mesh calculations *Concurrency : Practice Exper.* **3** 457–81