The Impact of Lookahead on Conservative Simulation

Jain J. Wang and Marc Abrams
Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0106

February 24, 1995

Abstract

This paper studies the impact of lookahead on the performance of conservative simulation for both open and closed models. For open models, we derive an upper bound on the performance improvement due to the lookahead. We show that the benefit of lookahead diminishes as the simulation length increases. For closed models, on the other hand, the performance improvement converges to a constant as the simulation length increases.

1 Introduction

For the conservative simulation [1, 2], many previous empirical studies (e.g., [4, 5, 8]) have suggested that good performance relies on good lookahead capability. However, there has been a lack of formal arguments to support this observation. In this paper, we develop stochastic models to analyze the impact of lookahead to conservative simulation. A review of conservative simulation is given first in this section. In Section 2, the relation between the simulation time and lookahead is derived for both open and closed simulation models. Conclusions are given in Section 3.

In conservative simulation, the communication channel between any pair of logical processes (LP) is modeled by a *link* which is specified statically. That is, all possible communication between two LPs has to be identified before the simulation run. For any pair of LPs, say LP_a and LP_b, we say that LP_b is a *descendent* of LP_a (likewise, LP_a is a *precedent* of LP_b) if LP_a can send LP_b messages. In each LP, incoming messages through each link are stored in a message queue in FIFO order. Each incoming link is associated with a clock whose value is equal to the timestamp of the first message in the corresponding message queue or the timestamp of the last received message if the corresponding message queue is empty. The term *least-valued link* of an LP refers to the incoming link that has the smallest clock value. Also, *logically-next message* refers to the first message in the message queue corresponding to the least-valued link, if the queue is non-empty. Let $t(m)$ denote the timestamp of a message m. In the conservative simulation, each LP repeatedly performs the following steps:

1. Identify the logically-next message m and process all local events (from the local event queue of the LP) that have a timestamp no greater than $t(m)$ in the order of their timestamps. For each event executed, update the local clock of the LP to the timestamp of the event.

2. Process message m (which may in turn generate some local events and output messages for other LPs). Place all newly-created local events and output messages, respectively, in the local event queue and output message queues.

3. Process all output messages (in output message queues) that have a timestamp no greater than $t(m)$ in the order of their timestamps.

4. Repeat steps 1 to 3 until the simulation termination condition is satisfied.

In this paper, we assume message processing is non-preemptive. Note that in step 1, when the message queue corresponding to the least-valued link is empty, the logically-next message can not be determined and thus the LP has to *block* until some messages

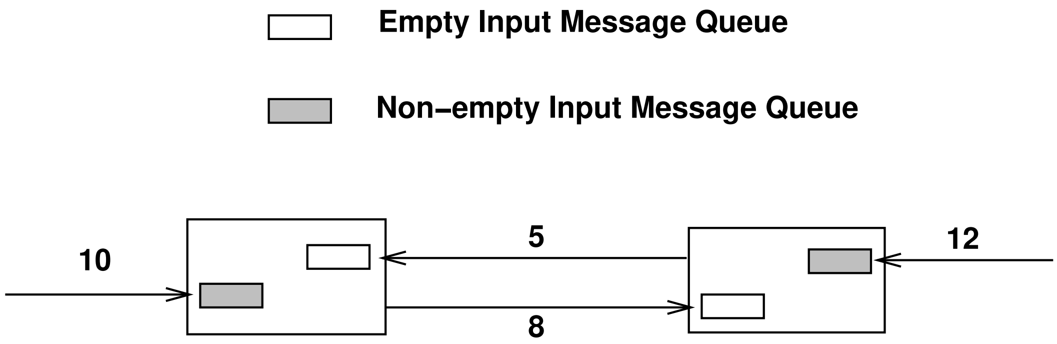

Figure 1: A deadlock. Processes A and B are waiting for each other to send messages. The number labeled with each link is the clock value of the link.

arrive at that message queue. A deadlock occurs if such blocking forms a circular dependency. An example of a deadlock is given in Figure 1.

Two approaches have been used to handle a deadlock, namely *deadlock avoidance* and *deadlock detection and recovery* (or simply *deadlock recovery*). Deadlock avoidance algorithms [1, 2] use *null messages* to prevent deadlocks from occurring. This is achieved in a way that whenever an LP finishes processing an event, a null message, whose timestamp indicates a lower bound on the simulation time of the next message created by the LP, is sent to all its descendants. It can be shown that as long as there does not exist any cycles in which the collective timestamp increment of the null messages is zero, no deadlock will occur [7]. Note that, a null message does not have any corresponding object in the target simulation model.

The use of null messages may pose a great amount of overhead. To reduce null message traffic, a conservative approach in which null messages are created on a demand basis has been proposed [7]. In this approach, a null message is created only if any of an LP blocks.

Unlike the deadlock avoidance approach, the deadlock recovery approach does not use any null messages and allows deadlocks to occur. Instead, a special mechanism is applied to detect and remove deadlocks. Lin [6] has shown that the deadlock recovery approach is in general inefficient. Unless mentioned otherwise, we assume that the deadlock avoidance approach is used for conservative simulation. Throughout this paper, the terms 'simulation time' and 'simulation execution time' refer to the *simulated* time and the *real* (i.e., wall clock) computation time for the simulation, respectively.

2 The Impact of Lookahead on Performance

An LP is said to have a *lookahead* ℓ if the clock value of the LP is t and the LP can predict that the timestamps of all events that arrive later are no smaller than $t + \ell$, for all $t \geq 0$. A number of empirical studies (e.g., [4, 5, 8]) have suggested that lookahead plays an important role in the performance of the conservative simulation. That is, good lookahead ability (i.e, a larger lookahead value) results in shorter execution time. Intuitively, this is because a good lookahead reduces the processor blocking time. Consider the example shown in Figure 2, where the LP has two input links with clock values 8 and 10, respectively. The LP has processed a message from the top message queue and updated its local clock value to 8. Because the top message queue becomes empty, the LP has to block if the LP does not have a lookahead of at least 2. Therefore, it appears that a larger lookahead implies a smaller blocking probability, hence a shorter execution time.

2.1 Does Lookahead Help?

Before we analyze the impact of lookahead on simulation performance with respect to different system topologies, this section first argues that a larger lookahead guarantees no worse simulation performance.

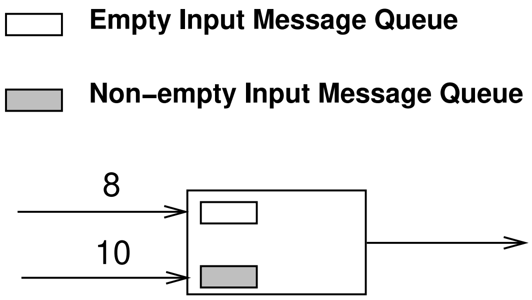

Figure 2: An LP which has two input links with clock values 8 and 10, respectively. Unless the LP has a lookahead of at least 2, the LP has to block until a message arrives at the empty input message queue.

Suppose that the target simulation model is mapped into P LPs, denoted LP_1, LP_2, ..., LP_P. Let G^a and G^b be two conservative simulation implementations for an arbitrary simulation model. Let ℓ_i^a (respectively, ℓ_i^b) denote the lookahead of the i^{th} LP for G^a (G^b). Suppose that G^a and G^b are identical (including all random variates involved in the simulation) except that $\ell_i^a \geq \ell_i^b$ for all i, $1 \leq i \leq P$. For convenience, the same set of symbols are used for variables in G^a and G^b except that each symbol is superscripted with 'a' or 'b' for distinction. Let T_r denote the total simulation execution time. Then T_r^a and T_r^b represent the simulation execution times for G^a and G^b, respectively.

Assume that G^a and G^b are run simultaneously on two identical, dedicated (i.e., to only the execution of G^a and G^b) multiprocessor systems. Let $e_{i,1}, e_{i,2}, \ldots$ be the sequence of events (ordered by their timestamps) that are executed by LP_i during the simulation. Also, let $t_r(e_{i,j})$, where $1 \leq j$, denote the real time at which event $e_{i,j}$ is executed.

Lemma 1 *For any simulation model,* $t_r^a(e_{i,j}) \leq t_r^b(e_{i,j})$, *for all* $1 \leq i \leq P$ *and* $1 \leq j$.

Proof: Assume that there exists an event $e_{i,j}$ such that $t_r^a(e_{i,j}) > t_r^b(e_{i,j})$. Then there must exist an event $e_{i',j'}$, where $1 \leq i' \leq P, j' \leq j$, such that both of the following conditions are satisfied:

1. For all events $e_{i'',j''}$ such that $t_r^b(e_{i'',j''}) < t_r^b(e_{i',j'})$, $t_r^b(e_{i'',j''}) \geq t_r^a(e_{i'',j''})$.

2. $t_r^a(e_{i',j'}) > t_r^b(e_{i',j'})$.

Given condition 1, condition 2 can be true only if $LP_{i'}^a$ blocks and $LP_{i'}^b$ does not block immediately before $e_{i',j'}$ is executed. However, this is impossible because $\ell_{i'}^a \geq \ell_{i'}^b$. $\Box$

Corollary 1 $T_r^a \leq T_r^b$ *for any given simulation model.*

Proof: Follows directly from Lemma 1. $\Box$

Therefore, a larger lookahead guarantees no worse performance. Examples showing that a larger lookahead does indeed reduce simulation execution time can be easily constructed. A question arises immediately is that how much such improvement can we expect? We answer this question in the following sections.

2.2 Open Models

We first consider the simple three-LP model shown in Figure 3. This simple model is referred to as the *basic* model. Note that, for an acyclic model a deadlock can not occur and hence null messages are unnecessary. Assume that the simulation terminates immediately after the local clock of LP_3 reaches time τ.

To evaluate the effect of lookahead, we consider two implementations G^O and G^Z where G^O has an *optimal* lookahead while G^Z has zero lookahead. That is, in G^O, all LPs can exactly predict the timestamps of all messages that have not arrived at any time. Thus, it is guaranteed that no blocking occurs in G^O for any LP only if the message queue corresponding to the least-valued link is empty. For G^Z, on the other

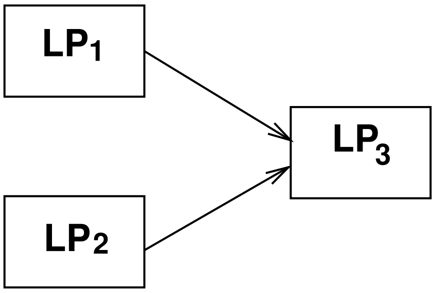

Figure 3: A simple simulation model of three LPs.

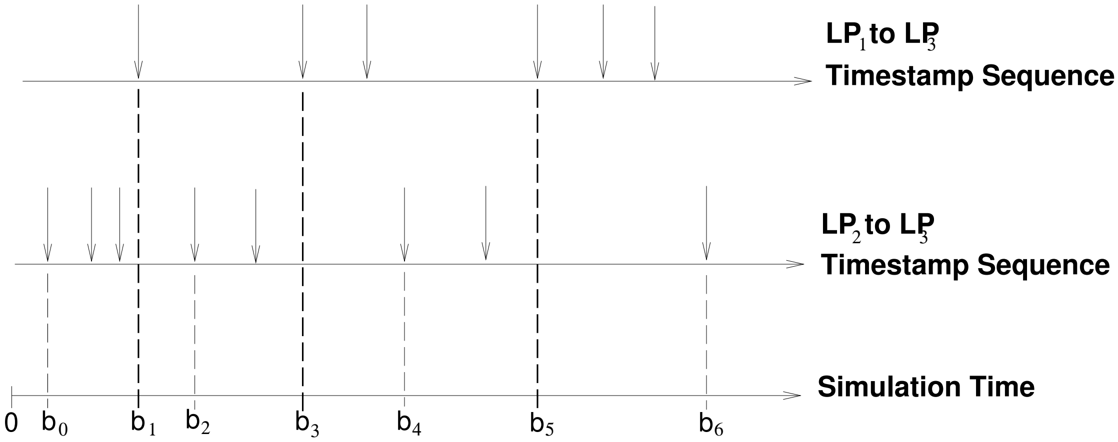

Figure 4: Timestamp sequences for the basic model. Each simulation time interval $[b_i, b_{i+1})$ consists of a set of timestamps corresponding to messages that are sent by the same LP.

hand, no prediction for the timestamps of future messages is made. Based on Corollary 1, we see that G^Z and G^O, respectively, give the upper bound and the lower bound on the simulation execution time for the conservative simulation with any amount of lookahead.

Let $[0, \tau]$ be the simulation time interval. We may obtain a set of subintervals divided by a set of $n (n \geq 1)$ increasing simulation time points $b_0, b_1, \ldots b_n$, such that: (1) each b_i, $0 \leq i \leq n$, corresponds to the timestamp of a message received by LP_3

and (2) all messages whose timestamps in the interval $[b_i, b_{i+1})$ are sent by the same LP (i.e., LP_1 or LP_2). An example of such a partition for the basic model is given in Figure 4. Each interval $[b_i, b_{i+1})$ will be referred to as a 'b-cycle'.

Analyzing the effect of lookahead requires some definitions. For simplicity, we assume that no two messages arrive simultaneously. Let $t_r(m)$ denote the real time at which message m arrives. Let $m(t)$ denote the message that arrives at simulation time t. Let R_i be the real time interval between the arrivals of messages $m(b_i)$ and $m(b_{i+1})$ (i.e., $m(b_i)$ and $m(b_{i+1})$ are the messages that arrive at simulation times b_i and b_{i+1}, respectively). That is, $R_i = t_r(m(b_{i+1})) - t_r(m(b_i))$. Also, let $T_r^h(t)$ (respectively, $T_r^l(t)$) denote the real time at which the simulation clock corresponding to G^O (G^Z) advances to simulation time t. Then the 'execution time gap' (i.e., the real execution time gap between G^O and G^Z) when the simulation clock advances to t is given by:

$$\Delta T_r(t) = T_r^l(t) - T_r^h(t) \geq 0.$$

Assume that the message transmission time is negligible. Then we can show that (in Lemma 2 and Corollary 2) $\Delta T_r(t)$ is bounded by $\max_{0 \leq i \leq n}\{R_i\}$ for all $t \geq 0$.

Lemma 2 *For all* $n \geq 0, \Delta T_r(b_n) \leq \max_{0 \leq i \leq n}\{R_i\}$.

Proof: This lemma can be proved by induction on n. When $n = 0$, $T_r^h(b_0)$ and $T_r^l(b_0)$ correspond to the real times that LP_3^h (i.e., LP_3 in G^O) and LP_3^l (i.e., LP_3 in G^Z) start processing their first message. It is clear that $T_r^h(b_0) \geq 0$ and $T_r^l(b_0) \leq t_r(m(b_1)) - t_r(m(b_0)) = R_0$. Thus, $\Delta T_r(b_0) \leq R_0$ (induction base). Assume that, $\Delta T_r(b_i) \leq \max_{0 \leq k \leq i}\{R_k\}$ for some $i \geq 0$ (induction hypothesis). We need to show that $0 \leq \Delta T_r(b_{i+1}) \leq \max_{0 \leq k \leq i+1}\{R_k\}$. To derive $\Delta T_r(b_{i+1})$, we consider the following three possible cases that could occur when LP_3^l processes the messages in the interval $[b_i, b_{i+1})$:

Case 1: LP_3^l runs continuously without blocking. Because no time has been spent in blocking, in this case, when LP_3^l starts processing $m(b_{i+1})$, the execution time gap between G^Z and G^O will not increase (It decreases if LP_3^h blocks while LP_3^l processing the messages in $[b_i, b_{i+1})$). Therefore, $\Delta T_r(b_{i+1}) \leq \Delta T_r(b_i)$.

Case 2: LP_3^l blocks for a message, say $m(b)$, in the interval $[b_i, b_{i+1})$. Note that, for LP_3^l and LP_3^h, the real message arrival times are the same for all messages. Thus, in this case, LP_3^h also has to block for message $m(b)$. Therefore, at this point, the progress of G^Z 'catches up with' G^O. That is, $\Delta T_r(b_{i+1}) = 0$.

Case 3: LP_3^l blocks for message $m(b_{i+1})$. At this point, LP_3^l has completed all messages in $[b_i, b_{i+1})$. Because $m(b_{i+1})$ has not arrived yet, we know that G^O must be ahead of G^Z for less than R_{i+1} real time units. Because LP_3^l can not resume computing until $m(b_{i+1})$ arrives, thus $\Delta T_r(b_{i+1}) = R_{i+1}$ in this case.

Combining the above cases yields $0 \leq \Delta T_r(b_{i+1}) \leq \max_{0 \leq k \leq i+1}\{R_k\}$. □

Corollary 2 *For all t in $[0, \tau]$, $\Delta T_r(t) \leq \max_{0 \leq i \leq n}\{R_i\}$.*

Proof: This corollary follows directly from the proof of Lemma 2. □

Corollary 2 gives an upper bound (i.e., $\max_{0 \leq i \leq n}\{R_i\}$) on the execution time gap between G^O and G^Z at any point during the simulation. Assume that $\{R_0, R_2, \ldots\}$ and $\{R_1, R_3, \ldots\}$ are two sets of IID (independent and identically distributed) random variables. Then an expected upper bound for $\Delta T_r(t)$ can be derived easily as discussed below.

Suppose the simulation terminates after LP_3 simulates n b-cycles. For convenience, we assume that n is an odd integer. Then from Lemma 2, we have

$$E(\Delta T_r(b_n)) \leq max\{E(R_0^{(n/2)}), E(R_1^{(n/2)})\}, \tag{1}$$

where $R_0^{(n/2)}$ and $R_1^{(n/2)}$ are the $(n/2)^{th}$ order statistic with respect to R_0 and R_1, respectively. Let $E(R_u) = max\{E(R_0), E(R_1)\}$ and $E(R_u^{(n/2)}) = max\{E(R_0)^{(n/2)}, E(R_1)^{(n/2)}\}$. Then it can be shown that [3, Ch.4.2]

$$E(R_u^{(n/2)}) \leq E(R_u) + \frac{\sigma_u(n/2-1)}{\sqrt{n-1}}, \tag{2}$$

where σ_u^2 is the variance of R_u. Suppose R_u is exponentially distributed with a mean 1. Then it can be derived that

$$E(\Delta T_r(b_n)) \leq E(R_u^{(n/2)}) = \frac{n}{2}\int_0^\infty te^{-t}(1-e^{-t})^{\frac{n}{2}-1}dt = \sum_{i=1}^{n/2}\frac{1}{i}.$$

Lemma 3 $E(\Delta T_r(b_n)) \leq E(R_u) + \frac{\sigma_u(n/2-1)}{\sqrt{n-1}} = O(\sqrt{n})$.

Proof: Follows directly from (1) and (2) and the fact that $E(R_u)$ is a constant with respect to n. □

Lemma 3 gives an upper bound on the expected execution time gap between G^O and G^Z.

Corollary 3 $\lim_{n\to\infty} \frac{E(\Delta T_r(b_n))}{n} \to 0$.

Proof: Follows directly from Lemma 3. □

Corollary 3 states an important result: *no matter how large the lookahead is (assuming that it is finite), the performance improvement due to the lookahead becomes insignificant when the simulation length is long.* Note that, with arguments similar to those given above, we can easily show that this result also applies to the cases where LP_3 has more than two input message links.

The results obtained for the basic model extend to a more general acyclic model. Define a *source* LP to be one that has no precedent LP. We say that an LP has *depth d*

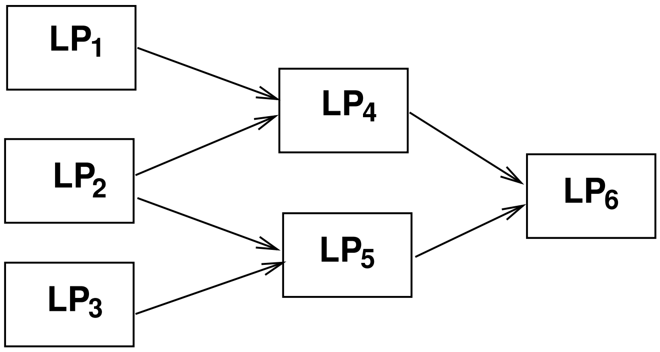

Figure 5: An open acyclic model.

(where $d \geq 0$) if the longest path from a source to the LP consists of d links. Consider the example in Figure 5. LP_1, LP_2 and LP_3 are sources with depth zero while LP_4 and LP_5 have depth one and LP_6 has depth two.

Assume that, except for source LPs, each output message is caused by exactly one input message and the output message waiting time is negligible. For example, in a queueing model, if each LP contains a single queue, then an output message (representing a job departure) can be sent to the decedent LP immediately after it is created (thus no output waiting time). Assume that the simulation terminates after an LP, say LP_x, simulates n b-cycles.

Let d_x denote the depth of LP_x. Then, it is expected that $E(\Delta T_r(n))$ increases as d_x increases. This is because in G^Z, messages are 'propagated' from the sources to other LPs at a lower speed due to the lack of lookahead. However, from Corollary 2 and Lemma 3, we can see that for each extra level of depth for LP_x., $E(\Delta T_r(n))$ can increase by at most $c\sqrt{n}$ for some constant $c > 0$. That is,

$$E(\Delta T_r(n)) = O(d_x\sqrt{n}).$$

In this case, again, $\lim_{n\to\infty} \frac{E(\Delta T_r(n))}{n} \to 0$. Thus, for acyclic models when the sim-

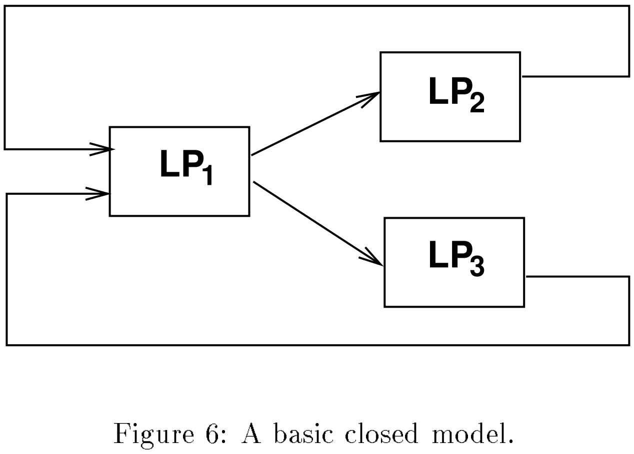

Figure 6: A basic closed model.

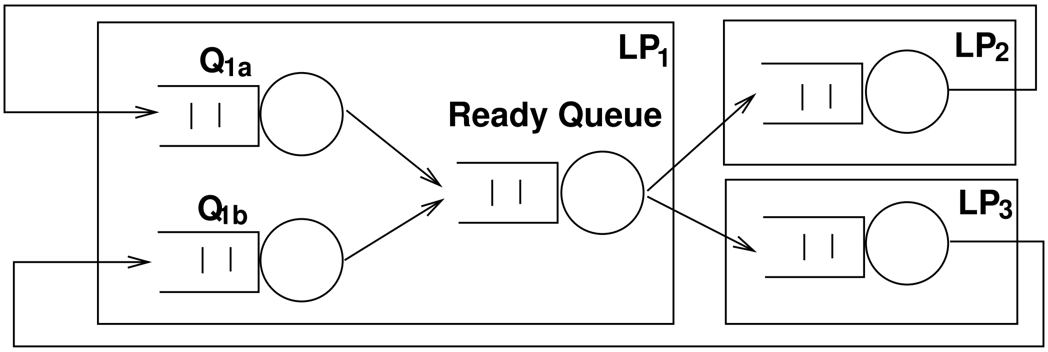

Figure 7: A queueing network representation of the basic closed model. Each queue corresponds to a message queue in the cyclic model. The extra queue for LP_1 is the 'ready' queue which buffers jobs that are ready for execution.

ulation length is long, the performance gain due to an increased lookahead becomes insignificant regardless of the amount of the lookahead.

2.3 Closed Models

This section examines the effect of lookahead for closed models. For simplicity, we consider the simple closed model in Figure 6. To derive a performance model, we view this conservative simulation system as a closed queueing network (shown in Figure 7 in which the ready queue buffers jobs that are ready for execution) such that each message queue and a message, respectively, corresponds to a queue and a job in the queueing network. In this section, the terms 'message' and 'job' are used interchangeably.

To study the effect of lookahead for such a system, again, we consider two implementations G^a and G^b. The definitions of G^a and G^b are identical to those given in Section 2.1 (i.e., all LPs in G^a have a larger lookahead than that in G^b). Define the *response* time of the queueing network to be the time required for a job to circulate the network exactly once (i.e., the time interval between two consecutive visits to LP_1). Then we *mark* an arbitrary job and let W_i denote the i^{th} response time (i.e., the time spent in the i^{th} circulation) with respect to this marked job. Assume that, for all i greater than some sufficiently large i_0, the stochastic processes $\{W_i^a, i \geq i_0 > 0\}$ and $\{W_i^b, i \geq i_0 > 0\}$ are approximately covariance stationary. That is, for $i \geq i_0$, there exist two random variables W^a and W^b such that $E(W_i^a) \approx E(W^a)$ and $E(W_i^b) \approx E(W^b)$. Let $\Delta W = W^b - W^a$. From the result of Lemma 1, we know that $E(\Delta W) \geq 0$.

Consider queues Q_{1a} and Q_{1b} which correspond to the two message queues of LP_1 in Figure 7. Normally, if neither queue is empty, the jobs are passed to the ready queue with zero service times (representing no blocking). When one of the queues, say Q_{1b}, becomes empty and if the timestamp of the first message in Q_{1a} is greater than the sum of the lookahead and the clock value of LP_1, messages in Q_{1a} have to wait for some time δ (representing message blocking) until a job arrives at Q_{1b}. Then the waiting time δ can be treated as though it is the service time of the first job in Q_{1a}. Therefore, the larger the lookahead the less likely it is that a job will be assigned a service time greater than zero. As a result, a smaller response time is expected. Therefore, ΔW increases as the lookahead difference between G^a and G^b

increases. Suppose the simulation terminates after the marked job circulates through the network n times, where $n \gg i_0$. Then the expected execution time gap (i.e., $E(\Delta T_r(n))$) between G^a and G^b is given by:

$$E(\Delta T_r(n)) \approx nE(\Delta W) = \Theta(n). \tag{3}$$

Then $E(\Delta T_r(n))$ grows linearly with the simulation length. Therefore, the amount of lookahead is critical to the simulation performance of closed models. From (3), the expected speedup of G^a over G^b due to a larger lookahead is given by:

$$E\left(\frac{T_r^b(n)}{T_r^a(n)}\right) \approx \left(\frac{E(T_r^a(n)) + nE(\Delta W)}{E(T_r^a(n))}\right) = 1 + E\left(\frac{\Delta W}{W^a}\right).$$

Therefore the speedup is independent of n.

Note that in our analysis null messages are ignored. With null messages, however, it will only increase the performance gap between G^a and G^b because G^b is likely to require more null messages. This agrees with our conclusion that lookahead is important for conservative simulation for closed models.

3 Summary

The effect of lookahead to conservative simulation is investigated. For open acyclic models, we derive an upper bound on the performance improvement due to lookahead. We show that when the simulation length is long, increasing the amount of lookahead will not improve the simulation performance significantly. For closed models in which message departure times determine future message arrival times, the amount of lookahead is critical to simulation performance. In particular, the speedup due to a larger lookahead converges to a constant as the simulation length increases.

References

[1] R. E. Bryant. Simulation of Packet Communication Architecture Computer Systems. MIT-LCS-TR-188, Massachusetts Institute of Technology, 1977.

[2] Chandy, K. M., Misra, J. Asynchronous Distributed Simulation via a Sequence of Parallel Computations. *Commun. ACM* 24, 11 (Nov. 1981), 198-205.

[3] David, H. A., *Order Statistics*, 2nd ed., John Wiley, New York, 1981.

[4] Fujimoto, R. M. Lookahead in Parallel Discrete Event Simulation. *Proceedings of International Conference on Parallel Processing.* (August 1988), 34-41.

[5] Fujimoto, R. M. Parallel Discrete Event Simulation. *Commun. ACM* 33, 10 (Oct. 1990), 31-53.

[6] Lin, Y. B. Understanding the Limits of Optimistic and Conservative Parallel Simulation. Technical Report 90-08-02, Department of Computer Science and Engineering, University of Washington, 1990.

[7] Misra, D. Distributed-Discrete Event Simulation. *ACM Comput. Surv. 18*, 1, (Mar. 1986), 39-65.

[8] Nicol D. M. Parallel Discrete-Event Simulation of FCFS Stochastic Queueing Networks. *SIGPLAN Not.* 23, 9 (Sept. 88), 124-137

[9] Rönngren R., Rajaei, H., Popescu, A., Liljenstam, M., Ismailov, Y., Ayani, R. Parallel Simulation of a High Speed LAN *Proceedings of the 1994 Workshop on Parallel and Distributed Simulation* (July 1994), 132-138.