Stability of Event Synchronisation in Distributed Discrete Event Simulation

Anurag Kumar and Rajeev Shorey
Dept. of Electrical Communication Engg.
Indian Institute of Science
Bangalore, 560 012, INDIA
e-mail: anurag, shorey@ece.iisc.ernet.in

Abstract

This paper is concerned with the behaviour of message queues in distributed discrete event simulators. We view a logical process in a distributed simulation as comprising a message sequencer with associated message queues, followed by an event processor. We show that, with standard stochastic assumptions for message arrival and time-stamp processes, the message queues are *unstable* for conservative sequencing, and for conservative sequencing with *maximum lookahead* and hence for optimistic resequencing, and for any resequencing algorithm that does not employ interprocessor "flow control". These results point towards certain fundamental limits on the performance of distributed simulation of open queueing networks.

1 Introduction

In a distributed discrete event simulation, the simulation model is partitioned into several logical processes (LPs) which are assigned to the various processing elements. The time evolution of the simulation at the various logical processes is synchronised by means of time stamped messages that flow between the logical processes. We are concerned with the situation in which the messages are not just for synchronization, but also carry "work" which when done modifies the state of the receiving logical process. A typical example is the distributed simulation of a queueing network model, in which one or more queues is assigned to each logical process, and the messages indicate the motion of customers between the queues in the various logical processes. The simulation makes correct progress if each logical process processes the incoming events, from all other logical processes, in time stamp order.

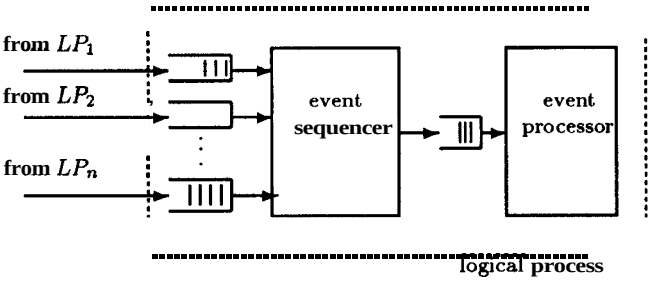

Figure 1: Schematic View of a Logical Process

Each logical process may be viewed as comprising an input queue for each channel over which it can receive messages from another logical process (i.e., $LP_1, LP_2, \ldots LP_n$) (see Figure 1). Since the messages must be processed in time-stamp order, the event processor must be preceded by an event sequencer. The messages must emerge from the sequencer in time stamp order.

It is the event sequencer that is at the core of much of the research in distributed discrete event simulation. Event sequencing algorithms fall in one of two classes: *conservative* or *optimistic*. A conservative event sequencer allows a message to pass through only if it is sure that no event with lower time-stamp can arrive in the (real time) future [3], [15]. An optimistic event sequencer, on the other hand, occasionally lets messages pass through without being sure that no lower time stamped event can arrive in the future. If then a lower time stamped event does arrive, corrective action is taken resulting in a roll-backof the simulation [8], [6].

Considerable work appears to have been done on performance models of distributed simulation with the objective of obtaining estimates or bounds on simulation speed-up with respect to centralised simulation [16],

[17], [23], [11], [5], [18]. In these models, speed-up is defined as the ratio of the real time rate of advancement of correctly simulated virtual time in a distributed simulation and a centralised simulation. These analyses are usually made under considerably simplifying assumptions, and generally yield bounds on the expected speedup. In particular, little attention seems to have been paid to the behaviour of the interprocessor message queues in distributed discrete event *simulators.* The simulation progresses by processing these messages. Thus we can view a simulator as a queueing network in which the customers are these interprocessor messages; the *throughput* of this network would correspond to progress of the simulation.

A notable exception in the literature is a recent paper [19], that was brought to our attention just as we had completed the work reported in this paper. Using mainly simulation results and heuristic reasoning the authors have anticipated some of the results we report here. Our results are based on a complete analysis of a formal stochastic model.

In this paper we study a particular class of stochastic models for message and time-stamp arrivals at an event sequencer in a logical process. We first show that for this class of models, and for conservative sequencing, the message queues that precede the sequencer are essentially unstable. Next, we show that even with *maximum lookahead* (i.e., prescient knowledge of the time-stamp of the next message yet-to-arrive on the channel with the empty message queue) these queues are still unstable. We then compare these instability results with certain stability results obtained, for synchronisation in a different context,, by Baccelli and Makowski [2]. We show the reason for the stability in their case. In this process we also obtain an instability result for a generalised version of our original model.

Maximum lookahead is an unrealisable algorithm as it requires information that is in practice not available. Hence messages left unsequenced by it at any time are at least as many as those for any realisable algorithm. It follows that the resequencing problem is fundamentally unstable, and some form of interprocessor "flow control" is necessary in order to make the message queues stable (without message loss).

The paper is organised as follows. In Section **2** we prove the instability of conservative resequencing. In Section **3** we show that even with maximum lookahead, resequencing is unstable. In Section **4** we use a slightly different model to generalise the results of Sections **2** and **3.** In Section *5* we discuss the implications of these results and directions for future work.

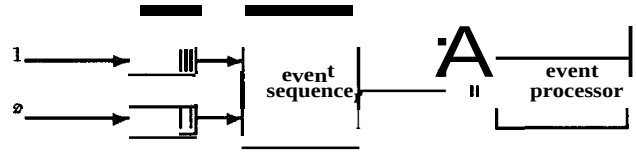

Figure 2: **A** Logical Process with 2 Input Message Streams

2 Instability of Conservative Sequencing

Two time stamped message streams arrive to a logical process. Within each stream the messages are in time-stamp order. The messages must be processed in overall time-stamp order by the event processor. In this section we assume that the sequencer uses the *conservative* sequencing algorithm.

Arriving messages queue up in their respective queues, at the sequencer, in their order of arrival. If both queues are nonempty then the sequencer takes the head-of-the-line (HOL) message with the smaller time-stamp and forwards it to the processor. We assume that the service time for doing this work is negligible and take it to be zero. If either of the queues is empty then the sequencer does not know the time-stamp order of the HOL message in the nonempty queue and does not forward any message to the processor.

It follows that at most one of the message queues at the sequencer is ever nonempty, and, in fact, exactly one of the queues is always nonempty. The queues evolve as follows. If, say, queue 1 is nonempty and a message arrives in stream 1, then it simply joins the end of queue 1, and no message is allowed to pass through to the processor. If a message arrives to queue **2,** then its time-stamp is compared with the HOL time-stamp in queue 1. If the newly arriving message has smaller time-stamp it is allowed to pass through, otherwise the HOL message in queue 1 is passed through. In the latter case as many messages from queue 1 pass through as have time-stamp less than the new arrival in queue **2.** If this leaves queue 1 empty then the arrival in queue **2** must be held until at least the next arrival epoch in stream 1, otherwise the arrival in queue **2** passes through leaving some unsequenced messages in queue 1.

We study the process of the number of unsequenced messages, embedded at the arrival epochs in the superposition of the two message arrival streams. Let X_n denote the number of unsequenced messages just *after* the n^{th} message arrival. If there are k messages in queue 1 then $X_n = +k$, whereas if there are k messages in queue 2 then $X_n = -k$. In this model for conservative sequencing, $X_n \neq 0$ for all n.

We assume that the two message arrival streams form

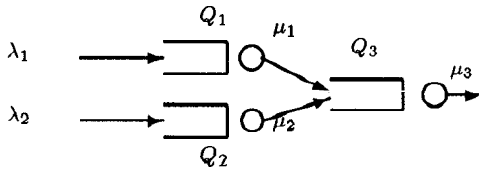

Figure 3: A Queueing Network

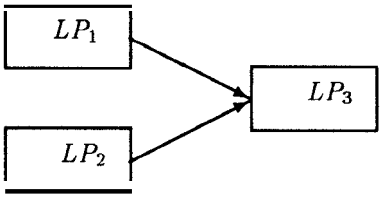

Figure 4: Distributed Simulator for the Model in Figure 3

Poisson processes with rates ν_1 and ν_2 respectively, and that the successive time-stamps in each stream are Poisson epochs with rates λ_1 and λ_2 respectively.

Remark: Before we proceed with the analysis of this model, we note here that this model is not vacuous and there exist instances of distributed discrete event simulation that yield such a model. Consider the queueing model in Figure 3 with external Poisson arrivals and exponential service times, $\lambda_1 < \mu_1$, $\lambda_2 < \mu_2$, $\lambda_1 + \lambda_2 < \mu_3$, and the queues Q_1 and Q_2 being *stationary.*

The model in Figure 3 is mapped onto the distributed simulater in Figure 4 in the obvious way. LP_1 and LP_2 simulate the work in system [10] in Q_1 and Q_2, and thus are driven by the two arrival processes. LP_i ($i \in \{1,2\}$) progresses by generating an interarrival time with distribution exponential(λ_i), updating the work in system process for Q_i and generating a departure event corresponding to the arrival. Since the queues are stationary, the departure processes in the queueing model are Poisson with rates λ_1 and λ_2, respectively. If it takes LP_i an exponentially distributed amount of time with mean ν_i^{-1} to do the work corresponding to each arrival, then we get a model for LP_3 that is exactly the same as described above. □

Define $\frac{\nu_1}{\nu_1+\nu_2} =: \alpha$, $\frac{\lambda_1}{\lambda_1+\lambda_2} =: \sigma$, and assume that $0 < a < 1, 0 < \sigma < 1$.

Theorem 1 *$\{X_n, n \geq 0\}$ is a Markov Chain on $\{\cdots,-3,-2,-1\} \cup \{1,2,3,4,\cdots\}$ with transition probabilities:*

for $i \geq 1$

$$p_{i,i+1} = \alpha$$

$$p_{i,i-j} = (1-\alpha)\sigma^j(1-\sigma) \quad \text{for } 0 \leq j \leq i-1$$
$$p_{i,-1} = (1-\alpha)\sigma^i$$

and

$$p_{-i,-(i+1)} = (1-\alpha)$$
$$p_{-i,-(i-j)} = \alpha(1-\sigma)^j\sigma \quad \text{for } 0 \leq j \leq i-1$$
$$p_{-i,1} = \alpha(1-\sigma)^i$$

Proof: The result is intuitively clear from the memoryless properties of the Poisson process and the exponential distribution. We present, however, a careful proof in the Appendix. □

The messages queues are found to be unstable in the following sense.

Theorem 2 *(i) For all $\nu_1, \nu_2, \lambda_1, \lambda_2$, ezcept those for which $\frac{\lambda_1}{\lambda_2} = \frac{\nu_1}{\nu_2}$, the Markov chain $\{X,\}$ is transient.*
(ii) For $\frac{\lambda_1}{\lambda_2} = \frac{\nu_1}{\nu_2}$, $\{X,\}$ is null recurrent.

Proof: These conclusions follow from standard Markov chain results. The detailed analysis is given in the Appendix. □

It follows that for all instances of the problem the message queues are unstable. In particular, we observe from the proof of Theorem 2 that if $\frac{\nu_1}{\lambda_1} < \frac{\nu_2}{\lambda_2}$ then the queue of messages received from LP_2 will grow without bound.

3 Instability of Sequencing with Maximum Lookahead

An optimistic sequencer works as in the case of conservative sequencing whenever both the message queues are nonempty. When a queue is empty, however, the processor is allowed to process messages in the nonempty queue. Messages whose time-stamps precede that of the next message to arrive in the empty queue, will get processed correctly. The rest will have to be reprocessed. Thus at any time there are messages that cannot be processed correctly even if the sequencer had *maximum lookahead,* i.e., (somehow) knew the time-stamp of the next message to arrive in the empty queue. These are the messages that optimistic sequencing (in fact, *any sequencing algorithm) cannot* process correctly *until* the next message in the empty queue is received. We show that the number of these messages forms a transient or null recurrent Markov chain under the same assumptions and conditions as before. Note that in conservative sequencing with lookahead, the *best* that lookahead can do is to let the sequencer know the time stamp of the next message to arrive to the empty message queue. Hence our term *maximum* lookahead. Maximum lookahead is an unachievable algorithm, but its analysis

should yield *fundamental limits* on the performance of any sequencing algorithm.

Assuming Poisson arrivals and Poisson time-stamps we model the sequencer with maximum lookahead as follows. When **a** number of messages is waiting at a queue, it is known that the next message to arrive in the other queue has time-stamp *smaller* than the HOL queued message. So when this message arrives it immediately passes through to the processor. Synchronisation is now complete until the time-stamp of this message. A time-stamp is sampled for the *next* message to arrive in the empty queue and this lookahead information is provided to the sequencer. This time-stamp is compared with the time-stamps of the messages in the nonempty queue and as many messages **as** precede this time-stamp are allowed to pass through. The "residual" time-stamps of the remaining messages in the queue are again distributed as epochs of a Poisson process. (see Lemma in the proof of Theorem *1).* If all the messages pass through in this way then synchronisation is complete until the time-stamp of the last message to pass through, and the residual time-stamp of the next message to arrive in the hitherto empty queue is exponentially distributed. **A** time-stamp is sampled for the *next* message to arrive in the just emptied queue. Now the sequencer has lookahead information for both empty queues. If **a** message arrives in the queue with the larger time-stamp, the message is retained, otherwise the message passes through.

Let $\{X_n, n \geq 0)$ denote the number of unsequenced messages just after n^{th} arrival, when the sequencer has maximum lookahead, with the same notation as before. Observe that now X, can be 0. Again we find that the message queues are unstable.

Theorem 3 $\{X_n, n \geq 0)$ *is a Markov chain on* $\{\cdots -3, -2, -1, 0, 1, 2, 3, 4, \cdots\}$ *with transition probabilities:*

for $i \geq$ I

$$\begin{aligned}
p_{i,i+1} &= a \\
p_{i,i-j} &= (1-\alpha)\sigma^j(1-a) \qquad 0 \leq j \leq i-1 \\
p_{i,0} &= (1-\alpha)\sigma^i \\
p_{-i,-(i+1)} &= 1-a! \\
p_{-i,-(i-j)} &= \alpha(1-\sigma)^j\sigma \qquad 0 \leq j \leq i-1 \\
p_{-i,0} &= \alpha(1-\sigma)^i \\
p_{0,1} &= \alpha(1-\sigma) \\
p_{0,-1} &= (1-\alpha)\sigma \\
p_{00} &= 1-(p_{0,1}+p_{0,-1})
\end{aligned}$$

Proof: Similar to Theorem **2.** □

Theorem 4 $(i)\{X_n, n \geq 0)$ *is transient except when* $\lambda_1\nu_2 = \lambda_2\nu_1$

$(ii)\{X_n, n \geq 0\}$ *is null recurrent when* $\lambda_1\nu_2 = \lambda_2\nu_1$

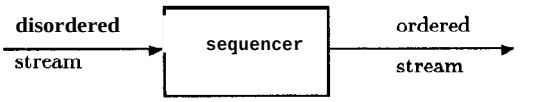

Figure 5: Ordering a Sequence Numbered Stream

Proof: Exactly the same as for Theorem **2.** □

It follows that the resequencing problem is fundamentally unstable, and *no* sequencing algorithm, that does not exercise some form of interprocessor "flow control", will yield stable message queues.

4 A Generalisation: Sequence Numbered Streams

Consider the model of Figure **2,** with the variation that, instead of carrying a time-stamp, each message carries a sequence number, which is unique across both streams, and within a stream the messages arrive in sequence number order. Obviously time-stamps imply such a sequence numbering, but sequence numbers provide more information to the sequencer than time-stamps alone. In fact, it is clear that, in the distributed simulation context, being provided with *sequence numbered messages is equivalent to maximum lookahead.*

The superposition of the two sequence numbered streams yields a message stream with all sequence numbers but the messages disordered. The reordering problem is depicted in Figure 5.

Let t_n be the arrival epoch of the message with sequence number n. Let r_n be the resequencing delay of this message. It follows that

$$r_{n+1} = (r_n - (t_{n+1} - t_n))^+ \qquad (1)$$

We will assume that $t_0 = 0$ and hence $r_0 = 0$. Then it is easily seen that

$$r_n = \max(0, \max(t_0, t_1, \cdots t_{n-1}) - t_n)$$

Case 1 : (Baccelli and Makowski [2], and Baccelli & Robert [1])

The disordered stream is obtained by passing an ordered stream (arrival epochs $s_n, n > 0$) through a disordering network that introduces random delays δ_n, n ≥ 0, i.e., $t_n = s_n + \delta_n$, where $s_0 \leq s_1 \leq s_2 \cdots \leq s_n \leq \ldots$ and $\delta_n \geq 0$. Hence, by Equation (1),

$$r_{n+1} = (r_n - (\delta_{n+1} - \delta_n) - (s_{n+1} - s_n))^+$$

Hence
for stationary, ergodic $\{(\delta_{n+1} - \delta_n), (s_{n+1} - s_n)\}$ with $E(\delta_{n+1} - \delta_n) + E(s_{n+1} - s_n) > 0$, r_n converges in distribution to a proper random variable. In particular if

$E\delta_n = E\delta_{n+1}$, it follows that $\{r_n\}$ converges and the resequencing system is stable (see [2]).

Case 2 :

The disordered stream is obtained by superposing two independent message streams, each message carrying a sequence number that is unique across both streams, and within each stream the messages are in time-stamp order.

Now (assuming $t_0 = 0$)

$$r_n = \max(0, \max(t_0, t_1, \cdots, t_{n-1}) - t_n)$$

Let the interarrival times in stream 1 be $a_1^{(1)}, a_2^{(1)}, \cdots$, and in stream 2 be $a_1^{(2)}, a_2^{(2)}, \ldots,$.

It follows that, for $n \geq 1$,

$\left(\sum_{i=1}^{k} a_i^{(2)} - \sum_{i=1}^{n-k} a_i^{(1)}\right)^+$
if t_n is in stream 1 and k of the messages in $\{1, \ldots n\}$ arrive in stream 2

$\left(\sum_{i=1}^{k} a_i^{(1)} - \sum_{i=1}^{n-k} a_i^{(2)}\right)^+$
if t_n is in stream 2 and k'of the messages in $\{1, \cdot\cdot, n\}$ arrive in stream 1

Theorem 5 *If*

(i) the arrival streams are independent and renewal *wzth life-time distributions $A^{(1)}(.)$ and $A^{(2)}(.)$, and mean interarrival dimes $\frac{1}{\nu_1}$ and $\frac{1}{\nu_2}$ respectively, and*

(ii) each sequence number $(1, 2, 3, \ldots)$ is assigned independently to one of the two streams, with probability σ_1 of assigning a number to stream 1 and probability $\sigma_2 = (1 - \sigma_1)$ of assigning a number to stream 2.

Then for $\frac{\sigma_1}{\nu_1} \# \frac{\sigma_2}{\nu_2}$, r_n converges in distribution to an improper distribution.

Proof: Consider the message with sequence number **n.** It is in stream 1 with probability σ_1 and in stream 2 with probability σ_2. Using the representation of r_n in terms of $\{a_k^{(1)}\}$ and $\{a', '')\}$ given above.

$$P(r_n > r) = \sigma_1 P\left(\left(\sum_{i=1}^{n-1} X_i - Y^{(1)}\right)^+ > r\right) + \sigma_2 P\left(\left(-\sum_{i=1}^{n-1} X_i - Y^{(2)}\right)^+ > r\right)$$

where

with probability σ_1	$(-X_i)$ has distribution	$A^{(1)}$
with probability σ_2	X_i has distribution	$A^{(2)}$
	$Y^{(1)}$ has distribution	$A^{(1)}$
	$Y^{(2)}$ has distribution	$A^{(2)}$

i.e.,

$$EX_i = \frac{-\sigma_1}{\nu_1} + \frac{\sigma_2}{\nu_2}$$

now by the Kolmogorov Strong Law of Large Numbers

$$\frac{1}{n}\sum_{i=1} X_i \longrightarrow EX \quad \text{with probability 1}$$

Hence if $\frac{\sigma_2}{\nu_2} > \frac{\sigma_1}{\nu_1}$ then
$\lim_{n\to\infty} P(r_n > r) = \sigma_1$ for all $r \geq 0$

and if $\frac{\sigma_2}{\nu_2} < \frac{\sigma_1}{\nu_1}$ then
$\lim_{n\to\infty} P(r_n > r) = \sigma_2$ for all $r \geq 0$ □

Thus, as a summary observation, if the events to be synchronised are triggered by some common parent stream, then synchronisation may be a stable process, whereas if the event streams to be synchronised are independent then synchronisation is usually unstable.

Observe that Poisson time-stamp processes, **as assumed** in Sections **2** and **3,** yield an instance of the sequence numbering process in hypothesis (ii) of Theorem 5. Thus Theorem 5 generalises Theorem **4** to renewal arrival processes. This extension is relevant since the departure process from a sequencer can be shown to have renewal message departure epochs and Poisson time-stamps.

5 Conclusions and Work in Progress

The instability results obtained in this paper are not surprising if viewed in the light of similar results obtained for other queueing models. It is easy to see that event synchronisation is similar to the assembly problem arising in manufacturing systems. If the parts to be assembled come from independent streams, it was shown by Harrisson [7] and Latouche [13] that under fairly general conditions the queues of parts to be assembled are unstable. The assumption of independent part streams may not be always appropriate in the manufacturing context, **as** the part streams usually originate from a common order stream. This constraint would lead to the Baccelli and Makowski [2] framework and hence to stability. No such "parent" stream can be argued in the context of distributed simulation of open queueing networks. Hence if all logical processors (LPs) are permitted to proceed at their own rates then message buffers will overflow. Such simulations must be stabilized by some form of interprocessor "flow control". For example, a buffer level based backpressure control can be applied by downstream LPs, or various

LPs can be prevented from getting too far apart in virtual time by means of a mechanism like time windows [20] or bounded lag [14].

While such mechanisms will serve to stabilize buffers, our approach, of modelling and analysing the message flow processes in the simulator, points towards certain fundamental limits of efficiency of distributed simulation, imposed by the synchronization mechanism. With reference to the model in Figure **4**, it is clear that the rate of departure of messages from LP_3 corresponds to the rate of progress of the simulation. But, if, for example, $\frac{\nu_1}{\lambda_1} < \frac{\nu_2}{\lambda_2}$ then the rate of resequenced messages from the sequencer is bounded above by $\nu_1(1 + \frac{\lambda_2}{\lambda_1})$ $(< (\nu_1+\nu_2))$, for both conservative and maximum lookahead sequencing, and for any level of flow control of LP_1 and LP_2. Firstly, this implies a stability condition for LP_3, and secondly it yields a computable limit on the efficiency of distributed simulation of such a problem. Ongoing work indicates that such results can be obtained for the simulation of more general open queueing networks [12].

APPENDIX

Proof of Theorem 1: Let τ_n denote the "virtual" time up to which synchronisation is complete just after the n^{th} arrival epoch. Note that τ_n is the time-stamp of the last message allowed to pass through at the n^{th} arrival. Time-stamps of queued messages and messages yet to arrive are viewed relative to τ_n, as increments beyond τ_n.

The result follows from the following Lemma.

Lemma: Let $X_n = i$, and the time-stamps of the queued messages relative to τ_n are $S_1, S_1+S_2, \cdots, S_1 + S_2 + \cdots + S_i$. $\{S_1, S_2 \cdots\}$ are i.i.d., $Exp\ (\lambda_1)$. Let T denote the time-stamp of the message arriving at the $(n+1)^{st}$ arrival epoch relative to τ_n. Let $\{T_1, T_1 + T_2, \cdots\}$ denote the time-stamps, relative to τ_{n+1}, of the messages left in queue after the $(n+1)^{st}$ arrival.

Then

$$P(X_{n+1} = j,\ T_1 > t_1, T_2 > t_2, \cdots, T_j > t_j \mid X_n = i)$$

$$= \begin{cases} \alpha \prod_{k=1}^{j} e^{-\lambda_1 t_k} & j = i+1 \quad (i) \\ (1-\alpha)\sigma^{i-j}(1-\sigma) \prod_{k=1}^{j} e^{-\lambda_1 t_k} & 1 \le j \le i \quad (ii) \\ (1-\alpha)\sigma^i e^{-\lambda_2 t_1} & j = -1 \quad (iii) \end{cases}$$

Proof: (i) $j = i+1$ if the $(n+1)^{st}$ arrival is from stream **1** (probability α). In that case $\tau_{n+1} = \tau_n$, and $T_1 = S_1$, $T_2 = S_2, \cdots, T_i = S_i$, $T_{i+1} \sim Exp(\lambda_1)$ and is independent of the others; (here $\sim$ is to be read "is distributed **as**").

(ii) Let $\ell = i - j$ for $1 \le j \le i$. In this case $\tau_{n+1} = T + \tau_n$, $T_1 = \sum_{k=1}^{\ell+1} S_k - T$, $T_2 = S_{\ell+2}, \cdots, T_j = S_{\ell+j} = S_i$, and $T \sim Exp(\lambda_2)$

$$P(X_{n+1} = j,\ T_1 > t_1, \cdots, T_j > t_j \mid X_n = i)$$

$$= (1-\alpha)P(G < T < G + S_{\ell+1},\ G + S_{\ell+1} - T > t_1,\ S_{\ell+2} > t_2, \cdots, S_{\ell+j} > t_j)$$

where $G := \sum_{k=1}^{\ell} S_k$, and which, letting $g(.)$ be the probability density of G,

$$= (1-\alpha) \int_0^\infty g(u)du \cdot \cdot \int_u^\infty \lambda_2 e^{-\lambda_2 t}\, dt\ e^{-\lambda_1(t_1+t-u)} e^{-\lambda_1 t_2} \cdots e^{-\lambda_1 t_j}$$

which, on simplification, yields

$$= (1-\alpha)\, \sigma^{i-j}(1-a) \prod_{k=1}^{j} e^{-\lambda_1 t_k}$$

(iii)

$$P(X_{n+1} = -1, T_1 > t_1 \mid X_n = i)$$

$$= (1-\alpha)P\left(\sum_{k=1}^{i} S_k < T,\ T - \sum_{k=1}^{i} S_k > t_1\right)$$

letting $G = \sum_{k=1}^{i} S_k$, and $g(.)$ be the probability density of G,

$$= (1-\alpha) \int_0^\infty g(u)du\ e^{-\lambda_2(t_1+u)}$$

$$= (1-\alpha)\sigma^i e^{-\lambda_2 t_1} \qquad \square$$

Thus, after each arrival epoch, the time-stamps of the queued messages are successive epochs of a Poisson process. Returning to the proof of Theorem *1*, let $\tau_0 = 0, X_0 = i_0\ (\ge 1)$, and let the time-stamps of these queued messages have the same distribution **as** the first i_0 epochs of a Poisson process of rate λ_1.

Consider

$$P_{\mathcal{P}}(X_{n+1} = j \mid X_0 = i_0, \cdots, X_{n-1} = i_{n-1},\ X_n = i_n)$$

where the subscript $\mathcal{P}$ denotes that the initial time stamps form a segment of a Poisson process. Now writing this out

$$= \frac{P_{\mathcal{P}}(X_1 = i_1, \ldots, X_n = i_n, X_{n+1} = j \mid X_0 = i_0)}{P_{\mathcal{P}}(X_1 = i_1 \cdots X_n = i_n \mid X_0 = i_0)}$$

Consider the numerator

$$P_{\mathcal{P}}(X_1 = i_1, \cdots, X_n = i_n, X_{n+1} = j \mid X_0 = i_0)$$

$$= \int_{(\mathcal{R}^+)^{i_1}} P_{\mathcal{P}}(X_1 = i_1, S_1^{(1)} \in ds_1, S_2^{(1)} \in ds_2, \cdots, S_{i_1}^{(1)} \in ds_{i_1} \mid X_0 = i_0) \cdot \cdot P(X_2 = i_2, X_3 = i_3, \cdots, X_{n+1} = j \mid X_0 = i_0, X_1 = i_1, S_1^{(1)} = s_1, \cdots, S_{i_1}^{(1)} = s_{i_1})$$

where $\mathcal{R}^+$ is the nonnegative real line, and $\{S_k^{(1)}\}$ are time-stamps of messages queued just after the first arrival, in relation to τ_1. Now note that since the arrival epochs form a Poisson process, and the time-stamps of the yet to arrive messages also form a Poisson process independent of the past, the conditioning on $X_0 = i_0$ in the second term under the integral can be dropped, and applying the lemma above we get

$$= P_{\mathcal{P}}(X_1 = i_1 \mid X_0 = i_0)\, P_{\mathcal{P}}(X_2 = i_2, X_3 = i_3 \cdots, X_{n+1} = j \mid X_1 = i_1)$$

Proceeding this way in the numerator and demominator we will get

$$P_{\mathcal{P}}(X_{n+1} = j \mid X_0 = i_0, \cdots, X_{n-1} = i_{n-1}, X_n = i_n) = P_{\mathcal{P}}(X_{n+1} = j \mid \mathrm{X}, = i_n)$$

where the transition probabilities are obtained from the Lemma. □

Proof of **Theorem 2:** (i) Let Q be the transition probability matrix restricted to the set of states $\{1, 2, 3 \cdots\}$. We will show that whenever $\lambda_1 \nu_2 \neq \nu_1 \lambda_2$, there exists a bounded, nonnegative, nonzero solution to (see [4])

$$Q\underline{y} = \underline{y}$$

i.e., $(y_1, y_2, \cdots)$ such that, for $i \geq 1$,

$$y_i = \alpha y_{i+1} + \sum_{j=0}^{i-1} (1-\alpha)\,\sigma^j (1-\sigma) y_{i-j}$$

Multiplying by z^i, for $0 < z < 1$, and summing from 1 to ∞

$$\sum_{i=1}^{\infty} z^i y_i = \sum_{i=1}^{\infty} \alpha z^i y_{i+1} + (1-\alpha)(1-\sigma) \sum_{i=1}^{\infty} \sum_{j=0}^{i-1} z^i \sigma^j y_{i-j}$$

i.e.,

$$\tilde{y}(z) = \frac{\alpha}{z}(\tilde{y}(z) - z y_1) + \frac{(1-\alpha)(1-\sigma)}{(1-\sigma z)}\, \tilde{y}(z)$$

from which, on simplification, we get

$$\tilde{y}(z) = \frac{z(1-\sigma z)}{(1-z)\left(1-\frac{\sigma}{\alpha} z\right)} y_1$$

Case (i) $\alpha \neq \sigma$ (i.e., $\frac{\nu_1}{\nu_1+\nu_2} \neq \frac{\lambda_1}{\lambda_1+\lambda_2}$, i.e., $\nu_1 \lambda_2 \neq \lambda_1 \nu_2$). Using partial fraction expansion

$$\tilde{y}(z) = \frac{\alpha}{\sigma} z \frac{y_1}{\frac{\alpha}{\sigma} - 1} \left(\frac{1-\sigma}{1-z} - \frac{1-\alpha}{\frac{\alpha}{\sigma} - z} \right)$$

$$= \sum_{j=1}^{\infty} z^j \left(\frac{(1-\sigma) - (1-\alpha)\left(\frac{\sigma}{\alpha}\right)^j}{1 - \sigma/\alpha} \right) y_1$$

hence solutions to $Q\underline{y} = \underline{y}$ for $\sigma \neq a$ are of the form, for $j \geq 1$,

$$y_j = \left(\frac{(1-\sigma) - (1-a)\left(\frac{\sigma}{\alpha}\right)^j}{1 - \sigma/\alpha} \right) y_1$$

It is clear that there is a bounded nonzero solution between 0 and 1 if $\sigma < a$ and none if $\sigma > a$. Thus for $\sigma < a$ there are states in $\{1, 2, 3, \ldots\}$ from which there is a positive probability of never leaving this set. Hence $\{X, \}$ would be transient. It is similarly clear that for $(1-\sigma) < (1-a)$, i.e., $\sigma > a$, there are states in $\{\cdots -3, -2, -1\}$ from which there is a positive probability of never leaving $\{\cdots, -3, -2, -1\}$. Thus for $\sigma \neq a$, {X,} is transient.

Case (ii) $\sigma = a$

$$\tilde{y}(z) = z \frac{1-\sigma z}{(1-z)^2} y_1 = \sum_{i=1}^{\infty} z^i (i(1-\sigma) + \sigma)\, y_1$$

Recalling that $\sigma < 1$, there is no bounded solution to $Q\underline{y} = \underline{y}$; hence $\{X, \}$ recurrent for $\sigma = a$.

Proof: (ii) From (i) we know that for $\frac{\lambda_1}{\lambda_2} = \frac{\nu_1}{\nu_2}$ (i.e. $\alpha = \sigma$) {X,} is recurrent. We show now that for $a = \sigma$, $\{X_n\}$ is not positive recurrent, and hence is null.

Consider the Markov chain $\{X'_n\}$ on the state space $\{0, 1, 2, 3, \ldots\}$ with the transition probabilities (recall that $p_{i,j}$ are transition probabilities **for** $\{X_n\}$):

$$\begin{aligned} p'_{i,j} &= p_{i,j} \text{ for } i \geq 1, j \geq 1 \\ p'_{i,0} &= p_{i,-1} \text{ for } i \geq 1 \\ p'_{0,1} &= p_{-1,1} = 1 - p'_{0,0} \end{aligned}$$

Observe that $\{X_n\}$ positive recurrent $\Longrightarrow$ $\{X'_n\}$ is positive recurrent. We show that for $a = \sigma$, $\{X'_n\}$ is not positive recurrent. To do this we use a result due to Kaplan [9] [see also [22]).

For $i \geq 1$,

$$\begin{aligned} & E(X'_{k+1} - X'_k \mid X'_k = i) \\ &= a - \sum_{j=0}^{i-1} j(1-\alpha)\sigma^j(1-\sigma) - i(1-\alpha)\sigma^i \\ &= \alpha - \left(\frac{1-\alpha}{1-\sigma}\right)\sigma\{1-\sigma^i\} \end{aligned}$$

Hence for $a = \sigma =: a$, and $i \geq 1$

$$E(X'_{k+1} - X'_k \mid X'_k = i) = a^{i+1} > 0 \text{ for } a > 0$$

Also, directly,

$$E(X'_{k+1} - X'_k \mid X'_k = 0) = a(1-a) > 0$$

for $0 < a < 1$.

Further, for $i \geq 1$, and $z \in (0,1]$

$$\sum_{j=0}^{\infty} p'_{ij}(z^i - z^j) \geq \sum_{0 \leq j < i} p'_{ij}(-(i-j))(1-z)$$

where we use the inequality

$$z^i - z^j \geq -(i-j)^+(1-z)$$

for $z \in [0,1]$ (see [21]). Hence for $i \geq 1, z \in (0,1]$, (see mean drift calculations above)

$$\begin{aligned} \sum_{j=0}^{\infty} p'_{ij}(z^i - z^j) &\geq -a(1-a^i)(1-z) \\ &\geq -a(1-z) \end{aligned}$$

Hence the required conditions in [9] are satisfied and $\{X'_k\}$ is not positive recurrent, implying that $\{X_k\}$ is not positive recurrent. It follows that $\{X_k\}$ is null recurrent for $\sigma = \alpha$. □

References

[1] Baccelli, F., and Robert, Ph., "Analysis of update response times in a distributed data base maintained by the conservative time stamps ordering algorithm". In *Performance, 83,* 415-436.

[2] Baccelli, F., and Makowski, A.M., "Synchronization in queueing systems". In *Stochastic Analysis of Computer and Communication Systems,* Ed. H. Takagi, North-Holland, 1990, 57-129.

[3] Chandy, K.M., and Misra, J. "Asynchronous distributed simulation via a sequence of parallel computations". *Commun. ACM 24,* 11 (November 1981), 198-205.

[4] Çinlar, E., *Introduction to Stochastic Processes,* Prentice-Hall, Inc., 1975.

[5] Felderman, R.E., and Kleinrock, L. "Bounds and approximations for self-initiating distributed simulation without lookahead" *ACM Trans. Modelzng Comput. Simulatzon* 1, 4 (October 1991), 386-406.

[6] Fujimoto, R.M. "Parallel discrete event simulation", *Commun. ACM* **33,** 10 (October 1990), 30-53.

[7] Harrison, J.M., "Assembly-like queues", *J. Appl. Prob.* 10 (1973), 354-367.

[8] Jefferson, D.R. "Virtual time", *ACM Trans. Prog. Lang. and Syst., 7,* 3 (July 1985), 404-425.

[9] Kaplan, M., "A sufficient condition for nonergodicity of a Markov Chain". *IEEE Trans. Inform. Theory,* IT-25, 4 (1979), pp. 470-471.

[10] Kleinrock, L., *Queueing Systems, Volume I: Theory,* John Wiley & Sons, 1975.

[11] Kleinrock, L. "On distributed systems performance". *Computer Networks and ISDN Systems* 20 (1990), 209-215.

[12] Kumar, Anurag and Shorey, Rajeev, "Stability & performance of distributed simulators for feedforward queueing networks", in preparation.

[13] Latouche, G. "Queues with paired customers". *J. Appl. Prob.* 18 (1981), 684696.

[14] Lubachevsky, B. D., "Efficient distributed event-driven simulations of multiple-loop networks". *Commun. ACM 32,* 1 (January 1989), 111-123.

[15] Misra, J. "Distributed discrete event simulation", *ACM Comput. Surv. 18,* 1 (March 1986), 39-65.

[16] Mitra, D., and Mitrani, I. "Analysis and optimum performance of two message-passing parallel processors synchronised by rollback". In *Performance '84,* Elsevier Science Pub., North Holland, 1984, 35-50.

[17] Nicol, D.M. "High performance parallelized discrete event simulation of stochastic queueing networks". In Proceedings of 1988 Winter Simulation Conference (December 1988), 306-314.

[18] Nicol, D.M., "The cost of conservative synchronization in parallel discrete event simulations". *JACM,* 40, 2 (April 1993), 304-333.

[19] Shanker, M.S., and Patuwo, B.E., "The effect of synchronization requirements on the performance of distributed simulations". In Proceedings of the 7th Workshop on Parallel and Distributed Simulation (PADS), 1993, 151- 154.

[20] Sokol, L. M., Weissman, J. B., and Mutchler, P. A., "MTW: An empirical performance study". In Proceedings of the 1991 Winter Simulation Conference (WSC), 557-563.

[21] Szpankowski, W., "Some conditions for nonergodicity of Marliov Chains". *J. Appl. Prob.* 22 (1985), 138-147.

[22] Szpankowski, W., "Towards computable stability criteria for some multidimensional stochastic processes". In *Stochastic Analysis of Computer and Communication Systems,* Ed. H. Takagi, North-Holland, 1990, 131-172.

[23] Wagner, D.B., and Lazowska, E.D., "Parallel simulation of queueing networks: limitations and potentials" *Perf. Eval. Review,* 17, 1 (May 1989), 146-155.