FLOW-CONTROLLED THROUGHPUT IN DISTRIBUTED SIMULATORS OF FEEDFORWARD QUEUEING NETWORKS

Rajeev Shorey*and Anurag Kumar
Dept. of Electrical Communication Engg.,
Indian Institute of Science,
Bangalore, 560 012, INDIA
Email: srajeev@in.ibm.com, anurag@ece.iisc.ernet.in

Abstract

Sequential simulation of complex systems can be exceedingly slow and expensive. An obvious means of obtaining faster simulation is to dedicate more resources to it. Distributed Discrete Event Simulation has emerged an important method to study the behaviour of various systems, such as, for example, wireline and wireless networks. In this paper we study the flow-controlled throughput in the distributed simulators of communication networks. In particular, we study feedforward queueing networks. It is known that, without some kind of interprocessor flow control, a distributed simulator of a feedforward queueing network is unstable, in the sense that event message queues, at event sequencers, grow without bound. Interprocessor flow control is thus necessary to stabilize the simulator. We study two flow control mechanisms: *bounded buffers* and *moving time windows*. We model a simple distributed simulator with these flow controls, and obtain the flow controlled throughput for each control, with conservative and maximum lookahead sequencing. For bounded buffer flow control we study the flow controlled throughput with varying buffer limit. We find that, in both flow control schemes, the flow controlled throughput is bounded by the throughput without flow control. For the moving time window protocol, we show that the rate of virtual time advance of the flow controlled logical processes is equal.

Keywords: Modelling, Communication Networks, Distributed Simulation, Throughput, Flow-control

1 Introduction

Performance modelling of communication networks can be quite complex, and, it is well known that except for some simple systems, mathematical analysis of communication networks is intractable. Closed form expressions are hard to find and therefore, simulation of such systems becomes necessary in order to study their detailed behaviour.

It is generally the case that sequential simulation of complex systems is exceedingly slow and expensive. Running a simulation requires substantial computational resources. An obvious means of obtaining faster simulation is to dedicate more resources to it. It is for this reason that Distributed Discrete Event Simulation [2] has received much attention in recent years.

In Distributed Discrete Event Simulation, the system being modelled, usually referred to as the physical system, is viewed as being composed of some number of *physical processes* that interact at various points in simulated time (also called the virtual time). The distributed simulator is constructed as a set of *logical processes*, one per physical process. All interactions between physical processes are modelled by time-stamped event messages sent between the corresponding logical processes (LPs). It is well known that a distributed simulation is correct if one adheres to the *local causality constraint*, i.e., if each LP processes events in nondecreasing time-stamp order [2].

In [6], we proved that in distributed simulators of feedforward queueing networks, the message queues that precede the event sequencers are unstable for conservative sequencing. We then proved that even with *maximum lookahead* (i.e., prescient knowledge of the time-stamp of the next message yet-to-arrive on a channel with an empty message queue) these queues are still unstable. We concluded in [6] that in feedforward models the resequencing problem is fundamentally unstable, and some form of interprocessor "flow control" is necessary in order to make the message queues stable.

The rate of departure of processed "customers" from the simulator is the *throughput* of the simulator. The throughput of the simulator is a useful quantity as it is a measure of progress of the simulation. For example, each departure from the simulator yields a sample of customer sojourn times; the larger the simulator throughput the greater the number of samples we get in a fixed amount of simulation time. With this in mind, in this paper, we focus on the throughput of distributed simulators of feedforward queueing networks with flow control. In the study of flow controlled throughput, we analyse two approaches, namely, bounding the buffers of the message queues, and Moving Time Windows [7], [8]. Various logical processes can be prevented from getting too far apart in virtual time by means of a mechanism like Moving Time Windows. We analyse bounded buffer flow control with conservative sequencing and with maximum lookahead sequencing; we analyse moving time window flow control with maximum lookahead sequencing. We obtain numerical results and expression for the flow

*The (corresponding) author's current address is: IBM India Research Laboratory, Block 1, Indian Institute of Technology, Hauz Khas, New Delhi 110016, India. Email: srajeev@in.ibm.com, Phone: 91-11-6861100; Fax: 91-11-6861555

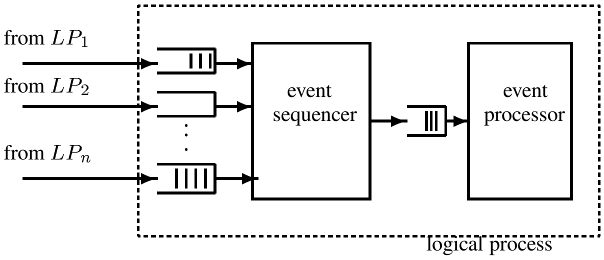

Figure 1: Schematic View of a Logical Process

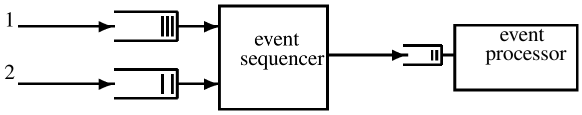

Figure 2: A Logical Process with Two Input Message Streams

control throughput; in each case the flow controlled throughput is found to be bounded by the throughput without flow control.

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.

The outline of the paper is as follows. In Section 2 we describe our model for a distributed simulator, and recall the basic instability result of [6]. In Section 3 we obtain the simulator throughput without flow control. In Sections 4.1 and 4.2 we analyse the model with bounded buffers, and moving time window flow controls, respectively. Section 5 has the Conclusions.

2 Instability of Event Sequencing

A distributed simulator comprises several logical processes (LPs) on several processors. Each logical process simulates a portion of the queueing network. The simulation is coordinated by exchange of time-stamped messages between the various LPs. Each LP may be viewed as comprising an input queue for each channel over which it can receive messages from another logical processes (e.g., $LP_1, LP_2, \ldots, LP_n$) (see Figure 1). The simulation makes progress when events are processed in an event processor associated with an LP. Since the event processor must process the events in time-stamp order, it must be preceded by an event sequencer. The event messages must emerge from the sequencer in time-stamp order. This view of a distributed simulator as itself being a queueing system has been used to study the performance of distributed simulation; see [6], [9].

Consider two time-stamped message streams arriving to a logical process (see Figure 2). 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 [6], we assumed that the two message arrival streams form independent Poisson processes, and that the successive time-stamps in each stream are independent sequences of Poisson

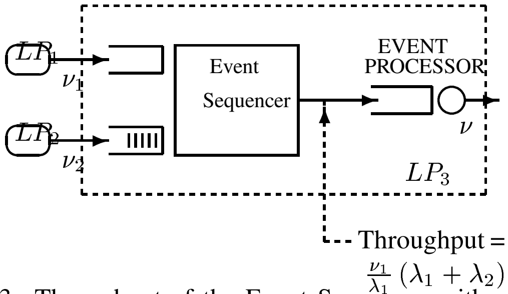

Figure 3: Throughput of the Event Sequencer with no Flow Control and $\frac{\nu_1}{\lambda_1} < \frac{\nu_2}{\lambda_2}$

epochs *independent* of the message arrival process. These assumptions apply to a simulator of a stationary, feedforward, Jackson network, and if the event processor service times are exponentially distributed. With these stochastic assumptions, we proved that in the model of Figure 2 the message queues are unstable for both conservative sequencing and sequencing with maximum lookahead[1].

We then showed ([6]) some generalizations of these instability results to point processes with certain ergodicity properties. Extensions of these instability results to simulators of queueing networks with feedback can be found in [3].

3 Throughput of the Event Sequencer with no Flow Control

In this section, we obtain the throughput of the event sequencer when there is no flow control. We consider a simple model with two arrival streams at the event sequencer (see Figure 3). We find an expression for the throughput of the event sequencer. The throughput of the event sequencer will be equal to the arrival rate of messages to the event processor that follows the event sequencer. The results can easily be generalized to the case where there are more than two arrival streams, or when the output from the event sequencer is fed to an event processor or another event sequencer.

We assume that the two message streams form 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. If now $\frac{\nu_1}{\lambda_1} < \frac{\nu_2}{\lambda_2}$ (i.e., the rate of arrival of virtual time from LP_2 is greater than that from LP_1), then the queue of messages received from LP_2 (i.e., Queue 2) will grow without bound. Thus a message arrival at Queue 1 will result in a batch of messages arriving at the event processor. In conservative sequencing, this batch consists of the arriving message in Queue 1, and all those messages in Queue 2 that have lower time-stamp than the message in Queue 1. In the maximum lookahead algorithm, since there is already a lookahead at Queue 1 (it being empty prior to the arrival), all the messages in Queue 2 with time-stamp lower than the lookahead would have already been sequenced. Now, an arrival at Queue 1 immediately leaves the sequencer resulting in a new

[1] An unrealisable, ideal algorithm in which the sequencer knows the time-stamp of the next event yet to arrive at a queue. In general purpose simulators, that use only event time stamps for sequencing, no sequencing algorithm can process more correct events than the maximum lookahead algorithm; see [5].

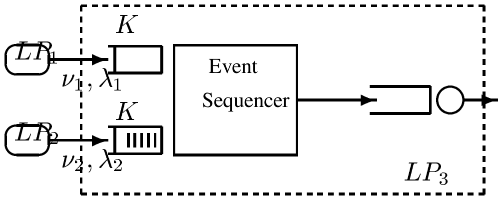

Figure 4: LP_1 and LP_2 Sending Messages to LP_3 (bounded buffers)

lookahead at Queue 1. Thus, in maximum lookahead, the departing batch consists of the arriving message at Queue 1 and all those messages in Queue 2 that have time-stamp that fall within the new lookahead.

The time-stamp process at stream i, $i = 1, 2$, is a Poisson process with rate λ_i. Therefore, the mean number of messages that depart from Queue 2 for each arrival into Queue 1, is given by $\frac{\lambda_2}{\lambda_1}$.
It follows that,

$$\text{Departure rate of messages from Queue 1} \quad = \quad \nu_1$$

$$\text{Departure rate of messages from Queue 2} = \nu_1 \left(\frac{\lambda_2}{\lambda_1}\right)$$

$$\Rightarrow \text{Throughput of the event sequencer} = \nu_1 + \nu_1 \left(\frac{\lambda_2}{\lambda_1}\right) = \nu_1 \left(1 + \frac{\lambda_2}{\lambda_1}\right)$$

The expression for the throughput of the event sequencer involves the parameters of the simulator (processor speeds) and the model being simulated, and hence clearly demonstrates the performance impact of various ways of mapping the simulation model onto the processors. In [5], we have studied in greater detail the mapping of the queues to the processors.

In the next section, we study the throughput of the event sequencer when there is flow control at the event sequencer queues, thus, rendering the message queues stable.

4 Throughput of the Event Sequencer with Flow Control

The instability result in [6] implies that if all logical processes (LPs) are permitted to proceed at their own rates then message buffers will overflow. Thus these simulations must be stabilized by some form of interprocessor "flow control". There are various ways to flow control the LPs. We could bound the buffers at the event sequencer and send flow control messages when buffer overflow is imminent, or, various LPs can be prevented from getting too far apart in virtual time by means of mechanisms like Time Windows [7]. In this section, we will analyse each of these mechanisms and find the throughput of the event sequencer.

4.1 Analysis of Bounded Buffer Flow Control

We assume that each message queue at the event sequencer can hold at most K messages. When the buffer size at a message queue reaches K, a flow control message is sent to the sending LP instructing it not to send any more messages. In Figure 4, an event sequencer with two message queues is shown. If the buffers (messages queues) have K messages, then LP_1 or LP_2 do not send any more messages. Note that we assume zero communication delays in the distributed simulator.

We assume that the two message arrival streams form 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.

We also assume that, on receiving a control message, an LP (LP_1 or LP_2) *stops* generating any more messages. When LP_3 releases the control, the LP that was blocked resumes message generation; owing to our assumption of exponential message generation times, the time after which the next message arrives from that LP is exponentially distributed.

4.1.1 Throughput with Conservative Sequencing

We wish to study the process $\{X(t)\}$ ($\in \{-K, \ldots, -3, -2, -1\} \cup \{1, 2, 3, \ldots, K\}$), where $X(t) = i$ if the number of unsequenced messages in Queue 1 is i, and $X(t) = -i$ if the number of unsequenced messages in Queue 2 is i. Note that both queues cannot have unsequenced messages, and in conservative sequencing one queue is always nonempty.

We study the process $\{X(t)\}$ embedded at the epochs in the *superposition* of two Poisson streams of rates ν_1 and ν_2. Denote this embedded process by $\{X_n\}$. When $-K < X_n < K$ then each epoch in the embedding Poisson stream corresponds to a message arrival. When $X_n = K$, however, then only the epochs of the Poisson process of rate ν_2 correspond to arrivals from LP_2; at the other epochs of rate ν_1 there is no change in the process $\{X_n\}$. Similar comments hold for $X_n = -K$. This is the *uniformisation* approach for analysing the continuous time process $\{X(t)\}$.

Define $\frac{\nu_1}{\nu_1+\nu_2} =: \alpha$, $\frac{\lambda_1}{\lambda_1+\lambda_2} =: \sigma$. If the rates ν_1, ν_2 and λ_1, λ_2 are strictly greater than zero, then $0 < \alpha < 1$, and $0 < \sigma < 1$.

The process $\{X_n, n \geq 0\}$ is a Markov Chain on $S := \{-K, \cdots, -3, -2, -1\} \cup \{1, 2, 3, 4, \cdots + K\}$ with transition probabilities:
For $1 \leq i < K$

$$\begin{aligned} p_{i,i+1} &= \alpha \\ p_{-i,-(i+1)} &= (1-\alpha) \end{aligned}$$

and, for $1 \leq i \leq K$

$$\begin{aligned} p_{i,i-j} &= (1-\alpha)\sigma^j(1-\sigma) \qquad \text{for } 0 \leq j \leq i-1 \\ p_{i,-1} &= (1-\alpha)\sigma^i \\ p_{-i,-(i-j)} &= \alpha(1-\sigma)^j\sigma \qquad \text{for } 0 \leq j \leq i-1 \\ p_{-i,1} &= \alpha(1-\sigma)^i \end{aligned}$$

At the boundaries of the state space K and $-K$, we have

$$\begin{aligned} p_{K,K} &= (1-\alpha)(1-\sigma) + \alpha \\ p_{-K,-K} &= \alpha\sigma + (1-\alpha) \end{aligned}$$

These equations easily follow from the assumptions of Poisson event arrivals and time-stamps. For example, consider $p_{K,K}$; if $X_n = K$ then $X_{n+1} = K$ at the next embedding epoch if the next epoch corresponds to an embedding epoch of rate ν_1 (this has probability α), or if the next epoch corresponds to an embedding epoch of rate ν_2 and the time-stamp of the arriving event (note that LP_2 is not blocked) is less than the time-stamp of the first event queued in Queue 1 (this has probability $(1-\alpha)(1-\sigma)$).

Let P_K be the transition probability matrix of this Markov chain. We obtain π, the stationary probability vector of the Markov chain by solving the equations [10]

$$\begin{aligned} \pi &= \pi P_K \\ \sum_{i \in S} \pi_i &= 1 \end{aligned}$$

We solve these equations numerically.

Departures from the event sequencer can occur only at epochs in the uniformising Poisson process of rate ν_1 and ν_2. The mean number of departures at such an epoch, when the sequencer buffer is in state i, is

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

The first term in this expression corresponds to the event that at this epoch an arrival occurs at Queue 2, and its time-stamp is larger than that of all messages in Queue 1; hence all the events in Queue 1 (i.e., i messages) can depart the sequencer. The second term is for the case in which the arrival at Queue 2 has a time-stamp larger than $j (0 \le j < i)$ messages in Queue 1, in which case j messages leave from Queue 1, and the arriving message also leaves.

Hence the throughput, τ_{cons}, of the system is given by

$$\begin{aligned} \tau_{\text{cons}} &= (\nu_1+\nu_2)\left\{\sum_{i=1}^{K}\pi_i\{i(1-\alpha)\sigma^i\right. \\ &+ \sum_{j=0}^{i-1}(j+1)(1-\alpha)\sigma^j(1-\sigma)\} \\ &+ \left.\sum_{i=1}^{K}\pi_{-i}\{i\alpha(1-\sigma)^i + \sum_{j=0}^{i-1}(j+1)\alpha(1-\sigma)^j\sigma\}\right\} \end{aligned}$$

4.1.2 Throughput with Maximum Lookahead Sequencing

Maximum lookahead is an idealised algorithm, in which the event sequencer is provided with the time-stamp of the next event yet to arrive at each empty queue. Suppose Queue 1 is empty. Events in Queue 2 whose time-stamps are less than the lookahead epoch in Queue 1 can be allowed to leave. When the event arrives to Queue 1 it is immediately allowed to leave (since Queue 2 has been sequenced upto the time-stamp of this arriving event), and a new lookahead is sampled for Queue 1. If now the lookahead at Queue 1 is so large that all the events in Queue 2 can leave, then a lookahead is obtained for Queue 2 also; thus both queues can become empty and virtual time moves up to the smaller of the virtual times at the two queues. Now the event arriving to the queue with the smaller (lookahead) virtual time will leave immediately, but the event arriving to the other queue will have to wait.

As in the previous section, let $\{X_n, n \ge 0\}$ denote the number of unsequenced messages just after n^{th} epoch in a Poisson stream of rate $\nu_1+\nu_2$. We use the same stochastic assumptions and notation as before. As observed above, now X_n can be 0.

With the above description of maximum lookahead sequencing, and the stochastic assumptions we have made, it is easy to see that the process $\{X_n, n \ge 0\}$ is a Markov chain on $\{-K, \cdots -3, -2, -1, 0, 1, 2, 3, 4, \cdots, +K\}$ with transition probabilities: For $1 \le i < K$

$$\begin{aligned} p_{i,i+1} &= \alpha \\ p_{-i,-(i+1)} &= 1-\alpha \end{aligned}$$

and, for $1 \le i \le K$,

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

For $i = K$

$$\begin{aligned} p_{K,K} &= (1-\alpha)(1-\sigma)+\alpha \\ p_{-K,-K} &= \alpha\sigma+(1-\alpha) \end{aligned}$$

The throughput (τ_{ML}) of the system is given by

$$\begin{aligned} \tau_{\text{ML}} &= (\nu_1+\nu_2)\left\{\sum_{i=1}^{K}\pi_i\{(1+i)(1-\alpha)\sigma^i\right. \\ &\quad + \sum_{j=0}^{i-1}(j+1)(1-\alpha)\sigma^j(1-\sigma)\} \\ &\quad + \sum_{i=1}^{K}\pi_{-i}\{(1+i)\alpha(1-\sigma)^i \\ &\quad + \left.\sum_{j=0}^{i-1}(j+1)\alpha(1-\sigma)^j\sigma\}\right\} \\ &\quad + (\nu_1+\nu_2)\left\{\{\alpha\sigma+(1-\alpha)(1-\sigma)\}\pi_0\right\} \end{aligned}$$

Consider, for example, the last term in the expression above. We give an explanation for this term. When the queues are both empty there is a lookahead in both the queues. An arrival to Queue 1 (with probability α) departs immediately if its time-stamp is less than the lookahead in the other queue (with probability σ). Similarly, we get the term $(1-\alpha)(1-\sigma)$. In either case at most 1 departure occurs.

4.1.3 Comparison of the Throughputs

In Figure 7, we show the throughput for conservative and maximum lookahead sequencing, versus K. In the figure, $\alpha = \frac{\nu_1}{\nu_1+\nu_2} = 0.5$, and $\sigma = \frac{\lambda_1}{\lambda_1+\lambda_2} = 0.5$. This implies that

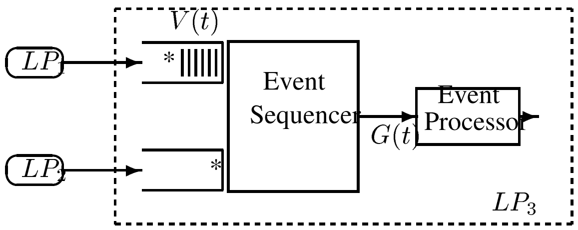

Figure 5: $V(t)$ Process at the Event Sequencer ($\star$ denotes maximum lookahead)

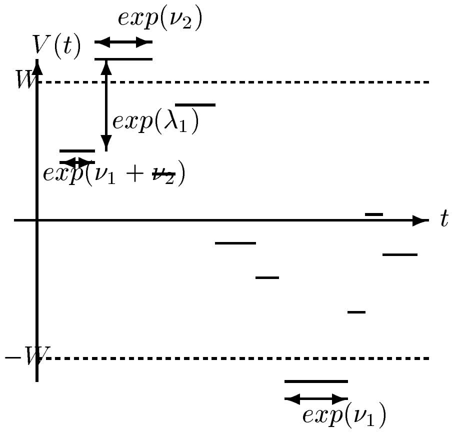

Figure 6: Flow Control with MTW of size W

$\frac{\nu_1}{\lambda_1} = \frac{\nu_2}{\lambda_2}$ and, therefore, the throughput is equal to $\nu_1 + \nu_2$ as $K \to \infty$. nter

It is seen that for reasonably small buffer sizes, the throughput in the maximum lookahead sequencer is strictly greater than that in the conservative sequencer. As the buffer size increases, we see that the throughput from both the sequencing algorithms converges to the same value. This is as expected from the discussions in Section 3.

The throughput of maximum lookahead is the best that can be achieved by any sequencing algorithm that relies only in time-stamp information. We have proved this formally in [5]. *Thus the gap between the two curves in Figure 7 indicates the scope for improvement if the conservative sequencing mechanism is replaced by, say, an optimistic sequencing mechanism.* As the available buffer size increases, throughput increases but the scope for improvement in the conservative algorithm decreases.

In the following subsection, we analyse other *flow control* mechanisms. For example, various LPs can be prevented from getting too far apart in virtual time by means of a mechanism like Time Windows. In the Moving Time Window protocol, a set of safe events is determined that can be processed concurrently in the distributed simulator. We analyse the Moving Time Window protocol with maximum lookahead sequencing.

4.2 Analysis of Moving Time Window (MTW) with Maximum Lookahead

The purpose of Moving Time Window (MTW) (introduced by Sokol et al. [7]) is to reduce the "search space" one must traverse in determining if an event is safe to process by an LP. The MTW approach uses a fixed time window of size W. Only events with time-stamps in the interval $[T, T+W]$, where T is the smallest time-stamped event in the simulation, are eligible for processing. An event outside the time window (i.e., later in virtual time) is blocked (or ineligible) until the window is advanced to include this event.

In this section, we analyse the simulator in Figure 5 with the MTW protocol. We assume maximum lookahead at the synchronizer. We recall that in maximum lookahead the synchronizer knows the next time-stamp expected from LP_1 or LP_2. Just after the arrival of a message from LP_i, $(i \in \{1,2\})$ the next time-stamp from LP_i is sampled and made known to the synchronizer.

Let $G(t)$ denote the virtual time upto which the event sequencer has completed sequencing at wall-clock time t, i.e., all messages from LP_1 or LP_2 with time-stamp less than $G(t)$ have been passed through by the sequencer. *The MTW restriction implies that LP_1 or LP_2 will not process an event if this will cause their virtual clocks to exceed $G(t)$ by more than W.* Let, for $t \geq 0$,

$X(t) (\in \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\})$:= Number of unsequenced messages at time t. $X(t)$ has the same interpretation as in Section 4.1.1.

$Y(t)$:= Sign($X(t)$) $\times$ (amount by which the time-stamp of last queued message is ahead of $G(t)$). Note that $Y(t) = 0$ if $X(t) = 0$.

$S_i(t)$:= Amount by which the maximum lookahead epoch on channel i at wall-clock time t is ahead of the known virtual time on channel i. If $X(t) > 0$, then $S_1(t) > 0$ and $S_2(t) = 0$ (the latter because with $X(t) > 0$ the lookahead on channel 2 would have been utilized, and the virtual time on channel 2 would be equal to the lookahead epoch on channel 2); if $X(t) < 0$, then $S_2(t) > 0$ and $S_1(t) = 0$; if $X(t) = 0$, then exactly one of $S_1(t)$ or $S_2(t)$ is zero with probability 1. Define

$$V(t) := \begin{cases} Y(t) + S_1(t) & \text{if } X(t) > 0 \\ Y(t) - S_2(t) & \text{if } X(t) < 0 \\ S_1(t) - S_2(t) & \text{if } X(t) = 0 \end{cases}$$

Observe that if $X(t) > 0$, then $G(t)$ is less than the time-stamp of the first event in Queue 1, and $V(t)$ is the amount by which LP_1 will be ahead of $G(t)$ if it were to process its next event. Figure 6 shows a sample path of $V(t)$ versus time t in the event sequencer.

For a MTW of W the process $V(t)$ evolves as follows (see Figure 6). Suppose $0 < V(t) < W$. Now a message arrival at t_k into Queue 2 will just pass through, since, owing to maximum lookahead, the effect of its time-stamp has already been accounted for in $V(t_k^-)$. A new sample from $Exp(\lambda_2)$ is taken (say, S_2') and

$$V(t_k^+) = V(t_k^-) - S_2'$$

An arrival at t_k into Queue 1 results in a new sample (say, S_1') being taken from $Exp(\lambda_1)$; this is the incremental time-stamp of the next event expected from LP_1 and

$$V(t_k^+) = V(t_k^-) + S_1'$$

Now suppose $V(t_k^+) \geq W$. This implies that the next event to be processed by LP_1 lies outside the time window of W, and LP_1 will not process this event until $V(t)$ falls below W; which it must as LP_2 can always process its next event since this will only take it 0 ahead of $G(t)$!, and the next lookahead at Queue 2 will be positive with probability 1! Parallel remarks hold for $V(t_k^+) \leq -W$. *Thus we note that MTW with maximum lookahead will not deadlock.*

We now consider the case with $X(t_k^+) = 0$. Such a situation arises, for example, if $X(t_k^-) > 0$, and an event arrives from LP_2. A maximum lookahead S_2' is sampled from Exp(λ_2), and $S_2' > Y(t_k^-)$, so that all the queued events can be allowed to immediately leave. Now $Y(t_k^+) = 0$ and $V(t_k^+) = V(t_k^-) - S_2$. If $V(t_k^+) \geq W$, then LP_1 will not process its event, and the next arrival will be from LP_2 at rate ν_2 (see Figure 6). This will pass through the sequencer. Another time-stamp will be sampled at LP_2, and hence, eventually, LP_1's event will fall within the time window. Suppose $\left|V(t_k^+)\right| < W$, then both LP_1 and LP_2 will process their events. If $W > V(t_k^+) > 0$ and LP_1's event arrives first, it will get queued and another sample from $Exp(\lambda_1)$ will be added to $V(t)$.

From the description above and Poisson event arrivals and Exponential time-stamp increments, $V(t)$ is a Markov Process. Assume that it is positive recurrent and let $f(t)$ be its *stationary density* on $(-\infty, \infty)$. For any $v \in (-\infty, \infty)$, the average rates of entrances into and exits from the composite state $[v, \infty)$ are equal in the limit [1]. Exits from composite state $[v, \infty)$ must necessarily correspond to entrance into composite state $(-\infty, v]$, and vice versa. Balancing the long run average rates of downcrossings and upcrossings af level v yields an equation that will involve the density function $f(\cdot)$. Now observe that, owing to Poisson arrivals and i.i.d. exponential time-stamp increment processes,

$$\text{Upcrossing rate of level } v = \nu_1 \int_{-\infty}^{(v \wedge W)} f(u) e^{-\lambda_1 (v-u)} du$$

$$\text{Downcrossing rate of level } v = \nu_2 \int_{(v \vee (-W))}^{\infty} f(u) e^{-\lambda_2 (u-v)} du$$

where the symbol "$\vee$" denotes maximum and "$\wedge$" denotes minimum. Equating these rates

$$\nu_1 \int_{-\infty}^{(v \wedge W)} f(u) e^{-\lambda_1 (v-u)} du = \nu_2 \int_{(v \vee (-W))}^{\infty} f(u) e^{-\lambda_2 (u-v)} du$$

Now, taking Laplace transform, consider the expression

$$\begin{aligned} &\int_{-\infty}^{\infty} \int_{(v \vee (-W))}^{\infty} f(u) e^{-\lambda_2 (u-v)} du \;\; e^{-sv} dv \\ = &\int_{-W}^{\infty} f(u) e^{-\lambda_2 u} du \int_{-\infty}^{u} e^{-v(s-\lambda_2)} dv \end{aligned}$$

which, for $s < \lambda_2$,

$$\begin{aligned} &= \int_{-W}^{\infty} f(u) e^{-\lambda_2 u} \, \frac{e^{-u(s-\lambda_2)}}{\lambda_2 - s} du \\ &= \frac{1}{\lambda_2 - s} \int_{-W}^{\infty} f(u) e^{-us} du \end{aligned}$$

Similarly, for $s > -\lambda_1$,

$$\begin{aligned} &\int_{-\infty}^{\infty} \int_{-\infty}^{(v \wedge W)} f(u) e^{-\lambda_1 (v-u)} du \;\; e^{-sv} dv \\ &\qquad = \frac{1}{\lambda_1 + s} \int_{-\infty}^{W} f(u) e^{-us} du \end{aligned}$$

Thus for $-\lambda_1 < s < \lambda_2$,

$$\frac{\nu_1}{\lambda_1 + s} \int_{-\infty}^{W} f(u) e^{-us} du = \frac{\nu_2}{\lambda_2 - s} \int_{-W}^{\infty} f(u) e^{-us} du$$

In particular, for $s = 0$,

$$\frac{\nu_1}{\lambda_1} \int_{-\infty}^{W} f(u) du = \frac{\nu_2}{\lambda_2} \int_{-W}^{\infty} f(u) du$$

But, $\int_{-\infty}^{W} f(u) du$ is the probability that LP_1 sends an event, and $\int_{-W}^{\infty} f(u) du$ is the probability that LP_2 sends an event. Calling these p_1 and p_2,

$$\frac{\nu_1 p_1}{\lambda_1} = \frac{\nu_2 p_2}{\lambda_2} \tag{1}$$

Observe that Equation 1 is interesting, as it states the intuitive result that *with flow control the rate of virtual time advance of the two flow controlled LPs is equal.*

Hence throughput under MTW with maximum lookahead is

$$\nu_1 p_1 + \nu_2 p_2 = \nu_1 p_1 \left(1 + \frac{\lambda_2}{\lambda_1}\right) = \nu_2 p_2 \left(1 + \frac{\lambda_1}{\lambda_2}\right)$$

Hence,

$$\begin{aligned} \nu_2 p_2 \left(1 + \frac{\lambda_1}{\lambda_2}\right) &= \nu_1 p_1 \left(1 + \frac{\lambda_2}{\lambda_1}\right) \\ &\leq \min\left(\nu_1 \left(1 + \frac{\lambda_2}{\lambda_1}\right), \nu_2 \left(1 + \frac{\lambda_1}{\lambda_2}\right)\right) \end{aligned}$$

Suppose now that $\frac{\nu_1}{\lambda_1} < \frac{\nu_2}{\lambda_2}$, then

$$\nu_1 \left(1 + \frac{\lambda_2}{\lambda_1}\right) < \nu_1 + \nu_2 < \nu_2 \left(1 + \frac{\lambda_1}{\lambda_2}\right)$$

$$\Rightarrow \nu_1 p_1 + \nu_2 p_2 < \nu_1 \left(1 + \frac{\lambda_2}{\lambda_1}\right)$$

Thus, $\nu_1 \left(1 + \frac{\lambda_2}{\lambda_1}\right)$, the throughput without flow control, is a bound on the throughput with flow control.

Thus, *we see that flow controlled throughput with MTW and maximum lookahead is strictly less than the throughput obtained when there is no flow control.*

Remark: The analysis of Moving Time Window assumed maximum lookahead sequencing. Note that strict Moving Time Window will *deadlock* if the event sequencer is *conservative*, for if LP_1 and LP_2 both find that their next time-stamp is ahead of $(G(t) + W)$, then neither will send messages.

Lubachevsky [4] studies the *Bounded Lag* protocol. Bounded Lag in a synchronization algorithm means that the difference in simulated time (virtual time) between events being processed concurrently is bounded from above by a known finite constant. Let $\tau(e)$ denote the time-stamp (virtual time) of an event e. The bounded lag restriction with parameter W is:

If events e_1 and e_2 are processed concurrently then $|\tau(e_1) - \tau(e_2)| \leq W$, where $0 \leq W \leq +\infty$ is a known constant.

A careful comparison of Moving Time Window and Bounded Lag [4] protocols shows that the two flow control protocols are identical in terms of their analytic performance modelling. Thus, in Figure 5, when $\frac{\nu_1}{\lambda_1} < \frac{\nu_2}{\lambda_2}$, the bound to the throughput of the event sequencer in MTW is the same as that in Bounded Lag, i.e., $\nu_1\left(1 + \frac{\lambda_2}{\lambda_1}\right)$.

5 Conclusions

In this paper, we have studied distributed simulators of feedforward queueing networks with respect to their throughput. In [6], we have shown that 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". One of the ways to flow control is to bound the buffers at the event sequencer. It was seen that for reasonably small buffer sizes, the throughput in the maximum lookahead sequencer is strictly greater than that in the conservative sequencer. As the buffer size increases, the throughput from both the sequencing algorithms converges to the same value.

We analysed the Moving Time Window protocol with maximum lookahead sequencing. We saw that flow controlled throughput in the Moving Time Window protocol, is strictly less than the throughput when there is no flow control. Further, the analysis of Moving Time Window protocol showed that the rate of virtual time advance of the flow controlled LPs is equal.

References

[1] P. H. Brill and M. J. M. Posner, "Level crossings in point processes applied to queues: single-server case," *Operations Research*, Vol. 25, No. 4, July-August 1977.

[2] R. M. Fujimoto, "Parallel discrete event simulation", *Commun. ACM*, Vol. 33, No. 10, pp. 30-53, October 1990.

[3] M. Gupta, A. Kumar, and R. Shorey, "Queueing Models and Stability of Message Flows in Distributed Simulators of Open Queueing Networks", In *IEEE/ACM/SCS 10th Workshop on Parallel and Distributed Simulation (PADS '96)*, Philadelphia, May, 1996.

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

[5] R. Shorey, "Modelling and Analysis of Event Message Flows in Distributed Discrete Event Simulators of Queueing Networks". Ph.D thesis, Dept. of Electrical Communication Engineering, Indian Institute of Science, Bangalore, India, 1997.

[6] R. Shorey, A. Kumar, and K. M. Rege, "Instability and Performance Limits of Distributed Simulators of Feedforward Queueing Networks". *ACM Transactions on Modelling and Computer Simulation (TOMACS)*, Vol. 7, No. 2, pp. 210-238, April 1997.

[7] L. M. Sokol, D. P. Briscoe, and A. P. Wieland, "MTW: A strategy for scheduling discrete simulation events for concurrent execution". In *Proceedings of the SCS Western Multiconference on Distributed Simulation*, 1988, pp. 34-42.

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

[9] D. B. Wagner and E. D. Lazowska, "Parallel simulation of queueing networks: limitations and potentials" in *Proc. 1989 ACM SIGMETRICS and Performance '89 Conf., 1989*.

[10] R. W. Wolff, *Stochastic Modelling and the Theory of Queues*, Prentice Hall, 1989.

Biography of the Authors

Rajeev Shorey

Rajeev Shorey completed his B.E in Computer Science, Indian Institute of Science, Bangalore, India in 1987. He then completed the M.S and Ph.D degrees in Electrical Communication Engineering Department, Indian Institute of Science, Bangalore, India in 1990 and 1996 respectively. He is currently a research staff member in the IBM India Research Laboratory, New Delhi. His areas of interest are Modeling and Analysis of Wireless and Wireline Networks. In the last one year, he has worked on the Bluetooth standard and Internet Traffic Engineering. He has IBM patents and numerous international papers to his credit.

Anurag Kumar

Prof. Anurag Kumar completed his B.Tech in the Electrical Engineering Department, Indian Institute of Technology, Kanpur, India in 1977. He then completed his Ph.D from Cornell, Ithaca, USA in 1981. After spending 7 years in AT&T Bell Laboratories in the Performance Modelling and Analysis Department, he joined the Indian Institute of Science, Bangalore, India. He is currently a Professor in the Electrical Communication Engineering Department in IISc, Bangalore. His research and consultancy interests are in Modelling, Analysis, Control, and Optimization problems arising in Communication Networks and Distributed Systems. He has been elected as a Fellow of the Indian National Academy of Engineering.

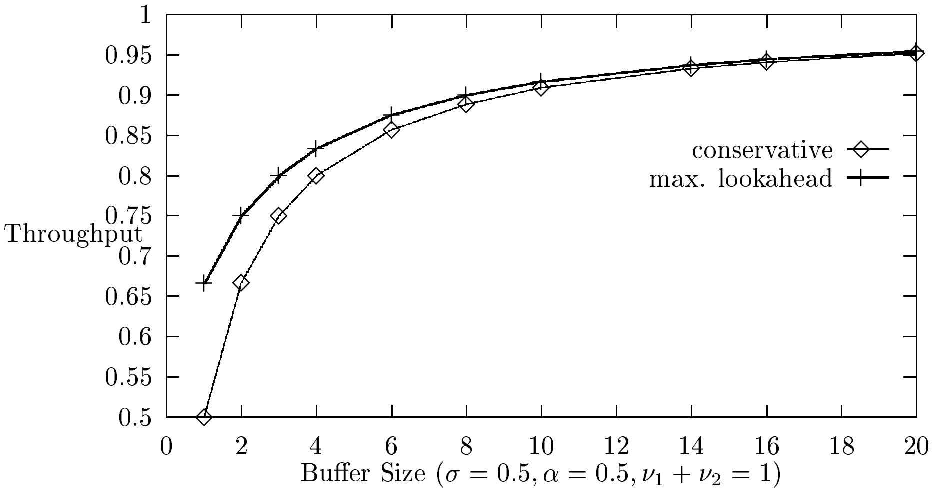

Figure 7: Throughput versus Buffer Size (K)