Proceedings of the 2013 Winter Simulation Conference
R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, eds.

ON THE TRANSIENT RESPONSE OF OPEN QUEUEING NETWORKS USING AD HOC DISTRIBUTED SIMULATIONS

Ya-Lin Huang
Richard Fujimoto

Computational Science and Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA

Christos Alexopoulos

Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA

Michael Hunter

Civil and Environmental Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA

ABSTRACT

Ad hoc distributed simulation, a methodology for embedded online simulation, has been studied for the steady-state simulation of open queueing networks. However, for most online simulation applications, the capability of a simulation approach to respond to system dynamics is at least as important as the performance in steady-state analysis. Hence, this paper focuses on the prediction accuracy of the ad hoc approach in open queueing networks with short-term system-state transients. We empirically demonstrate that, with slight modification to the prior ad hoc approach for steady-state studies, system dynamics can be modeled appropriately. Furthermore, a potential livelock issue that arises with the modification is addressed.

1 INTRODUCTION

This paper explores the ability of ad hoc distributed simulation to predict the transient behavior of physical systems. Ad hoc distributed simulation (Fujimoto et al. 2007) is an approach to embedding online simulations into a network of sensors that monitors the system under investigation (SUI). An online simulation is a predictive computational model that utilizes the data pertaining to the current state of a SUI to project future system states. This usage of real-world, up-to-date data allows model adaptation, which in turn potentially improves prediction accuracy; this facilitates system monitoring and control.

Online simulation is also referred to as dynamic data-driven application systems (Darema 2004), symbiotic simulation (Fujimoto et al. 2002), and cyber-physical systems (Lee 2008) in the literature. These approaches have been widely studied and applied to various science and engineering disciplines for a myriad of purposes (Davis 1998). One typical application concerns system monitoring, such as examining the structural and material health (Cortial et al. 2007, Farhat et al. 2006), and tracking wildfires (Douglas et al. 2006, Mandel et al. 2005) and hurricanes (Allen 2007). Another popular application is to optimize the operations of a physical system. For example, in an emergency situation, alternate evacuation scenarios may be modeled and evaluated in order to minimize evacuation time. The evacuation plan may need to adapt as the evacuation evolves when unforeseen events arise (Chaturvedi et al. 2006). Additional exam-

ples include planning paths for unmanned aerial vehicles (Kamrani and Ayani 2007), tuning parameters for computer networks (Ye et al. 2008), managing semiconductor manufacturing systems (Low et al. 2005), and optimizing surface transportation systems (Hunter et al. 2009a, 2009b).

Capturing system dynamics is crucial to online simulations, e.g., to trigger modifications to the configuration of a physical system in response to events. For example, a sudden increase in traffic volume may indicate that the changes in traffic signal plans for the responsive transportation system are necessary to help reduce congestion. This work adopts the queueing model as the benchmarking application because the queueing model is well known for its generality and flexibility to model real-world operational systems. Examples include vehicular/air traffic, computer systems, communication networks, supply chains, and many others that involve distributing limited resources/services among users. Since most of these are time-varying systems, transient-state analysis is vital to the success of corresponding applications.

Transient-state analysis can be complex and computationally expensive, especially in fulfilling real-time requirements. Transient-state analysis concerns the system state varying over the span of time; it takes into account not only the system state at every time point but also the correlations among the prior and posterior system states. Instead, to evaluate the ad hoc approach, this study simplifies the analysis: we consider a sufficient number of time points and compare the respective state predictions from the ad hoc approach against those from the corresponding sequential simulations. Specifically, the evaluation discretizes simulation time into small time intervals, and the output measures of interest (e.g., queue-length estimates) are calculated by averaging the numbers within each interval.

The rest of this paper is organized as follows. First, in Section 2, we examine the effectiveness of the preliminary ad hoc queueing simulation method introduced by Huang et al. (2012) in a scenario with increases in external arrival rates; the method reveals a delayed response in capturing the propagation of the expanded number of arrivals across modeled queueing networks. To resolve this delayed-response issue, Section 3 proposes a method and discusses the possible livelock issue following the new design. The new method, termed "iterative ad hoc queueing simulation method," is evaluated empirically in Section 4 under several network configurations, including one with a large increase in arrivals over a short period of time (which leads to rapid increases in queue occupancy). Finally, Section 5 concludes this study.

2 DELAYED RESPONSE IN THE ORIGINAL AD HOC QUEUEING SIMULATION METHOD

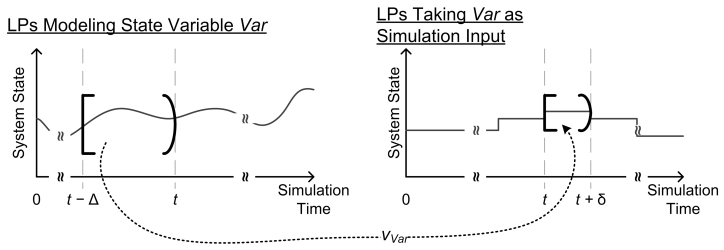


Figure 1: Information Sharing Mechanism Leading to Delayed Response

The existing ad hoc queueing simulation method introduced by Huang et al. (2012) is prone to delayed response to system dynamics because logical processes (LPs) share locally-observed current state information as predictions of the future. Specifically, consider the case where an LP models an object and shares the state information with other LPs that use the information as simulation input. As shown in Figure 1, at simulation time t, the LP computes a value based on the behavior of the object over the time period $[t - \Delta, t)$. The value becomes public as a predicted value of the object with respect to $[t, t + \delta)$, rather than $[t - \Delta, t)$. As a consequence, observations are not immediately reflected to the simulation model requiring the information, which results in delayed response.

The extent of this delayed-response issue depends on δ and Δ. The value δ is determined by the frequency in which LPs publish state information; smaller δ values allow changes to be revealed more frequently but introduce greater communication overhead in sharing information. On the other hand, the value Δ indicates the time interval needed to collect state information. While a large Δ may be used to reduce the variability of computed values, this may hide important system dynamics as the significance of system changes is mitigated by the last Δ-long history.

The following experiments illustrate the effects of δ and Δ. Three configurations are evaluated, all with $\delta = \Delta = 60$, 300, and 600 seconds. The rest of the simulation model is the same as that in the previous work, and the fundamental mechanisms are as follows. Each LP models an arbitrary queueing subnetwork using a sequential, discrete-event simulation. Every Δ seconds, LPs update the mean interdeparture times on the links they simulate, and request (or estimate if the data is unavailable upon request) the same information on the input links entering their individual modeled subnetworks. The arrivals on those input links are modeled as Poisson processes, and the rates at the beginning of simulations are all set to λ (same as the external arrival rate to every queueing station); thereafter, the rates are dynamically estimated using the data from rollbacks. The rollback criteria are based on acceptable ranges, constructed by a quality control paradigm. Since there may be several predictions from the LPs modeling the same link, the data retuned to the requesting LPs are generated using a kernel density estimation (KDE) approach.

The experiments involve two open queueing networks in Figure 2 with the intention to show the various degrees of impact due to the delayed-response issue. The network in Figure 2(a) is an 8-node, partially-bidirectional tandem network. Each node represents a single-server queueing station with an unlimited buffer, FIFO/FCFS service discipline, and independent and identically distributed (IID) exponential service times with the mean equal to 1 second. Each node has external Poisson arrivals with the rate λ. The probability of a processed unit moving to another node is p. Hence a processed unit leaves the network with probability $1 - p$ (at nodes 0, 4, or 7) or $1 - 2p$ (otherwise). The network in Figure 2(b) (referred to as a "completely-bidirectional tandem network") is almost the same to the former one with an exception that the processed units at node 4 may leave for node 3 with probability p.

The experiments on both the networks deploy 20 LPs in each replicate run: ten LPs modeling the leftmost 4 nodes and the remaining ten for the rightmost 4 nodes. In simulating the partially-bidirectional tandem network, shared information always goes from the left subnetwork to the right one. The right subnetwork "learns" the system dynamics in the left one through the changes in the flow rate (or, equivalently, the mean interdeparture time) of link 10. It is anticipated that the larger the Δ, the later the dynamics are detected. This phenomenon is expected to be worse for the completely-bidirectional tandem network since both the left and right subnetworks require information from each other.

The evaluation of δ and Δ considers two metrics over 10 IID ad hoc runs: the arrival rate across link 10 and the mean queue length at node 4. In one run, since the 10 LPs modeling node 4 (where link 10 enters) may use different arrival rates with respect to the same simulation time, these rates are averaged into one value. Then, the mean of the 10 values (each from one run) represents the point estimate. A similar calculation is performed to estimate the mean queue length at node 4 every 60 seconds.

The results from the corresponding sequential simulations serve as the ground truth. These results are based on 100 replicate sequential runs because 10 IID ad hoc runs deploy a total of 100 LPs to model one node. Note that these sequential simulations do not model the arrivals on link 10 as Poisson arrivals (as is done by ad hoc simulations). Instead, the arrivals are the departures from node 3 filtered by the probability p. For the output, the rate is estimated by the corresponding mean interarrival time, which is computed every 60 seconds based on the arrivals appearing since the last computation.

(a) Partially-Bidirectional Tandem Network (b) Completely-Bidirectional Tandem Network

Figure 2: 8-Node Tandem Queueing Networks

2.1 Case 1: Partially-Bidirectional Tandem Network

First, we study the partially-bidirectional tandem network with $p = 0.45$ and, for the first 4 hours in simulation time, $\lambda = 1/8$ per second; the steady-state traffic intensities of nodes range from 0.36 (nodes 1 and 4) to 0.66 (node 6). Afterwards, the external arrival rate λ increases to $1/6$ per second, causing the growth of the steady-state traffic intensities to the range between 0.48 (nodes 1 and 4) and 0.87 (node 6).

Figure 3 plots the experimental results from four different simulations: sequential simulations and ad hoc queueing simulations with $\delta = \Delta = 60$, 300, and 600 seconds. In order to focus on the transient period, the prior and later parts are removed for now. On the left is the estimated arrival rate across link 10. The results match the expectation that a larger Δ value gives rise to longer delay in incorporating state changes. The delay is approximately Δ in length except in the case of $\Delta = 60$ where the delay is slightly larger. This is because the predictions have a higher variation as they are based on a smaller amount of data. On the other hand, the right figure shows the estimated mean queue length at node 4 in the same simulation setting. Although the queue length is partly influenced by the arrivals on link 10, the discrepancy between the ad hoc runs and sequential runs is noticeable, albeit less severe.

2.2 Case 2: Completely-Bidirectional Tandem Network

This case concerns the completely-bidirectional tandem network with $p = 0.4$. Same as the above case, the external arrival rate λ increases from $1/8$ to $1/6$ per second after 4 hours in simulation time. Before the transition, the steady-state traffic intensities range from 0.31 (nodes 1 and 7) to 0.57 (nodes 4 and 5); following the rate change, they are between 0.41 (nodes 1 and 7) and 0.76 (nodes 4 and 5).

The experimental results are in Figure 4. Compared to those in Case 1, the ad hoc runs in this case take longer to pick up the state change. For example, the ad hoc runs with $\Delta = 600$ do not fully reach the expected state until approximately one hour after the change has occurred. This prolonged delay results from the "mutual dependence" of the left and right subnetworks. In other words, the projected arrival rate across link 3 relies on that along link 10, and vice versa.

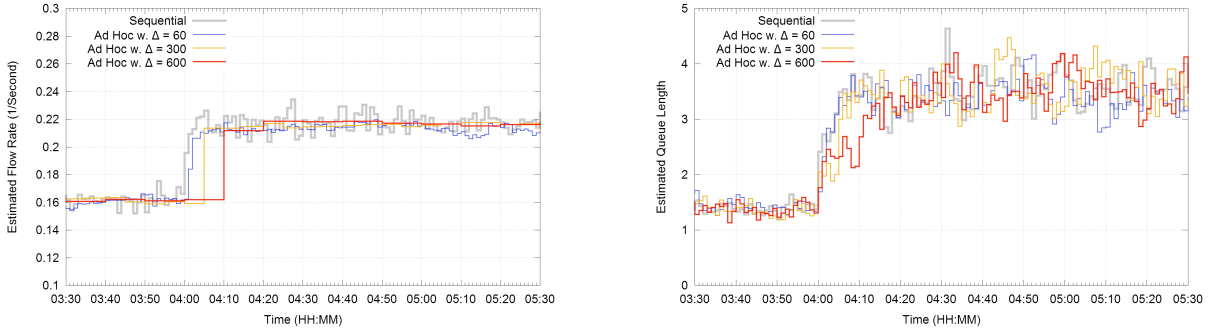


Figure 3: Estimated Arrival Rate across Link 10 and Mean Queue Length at Node 4 under Case 1

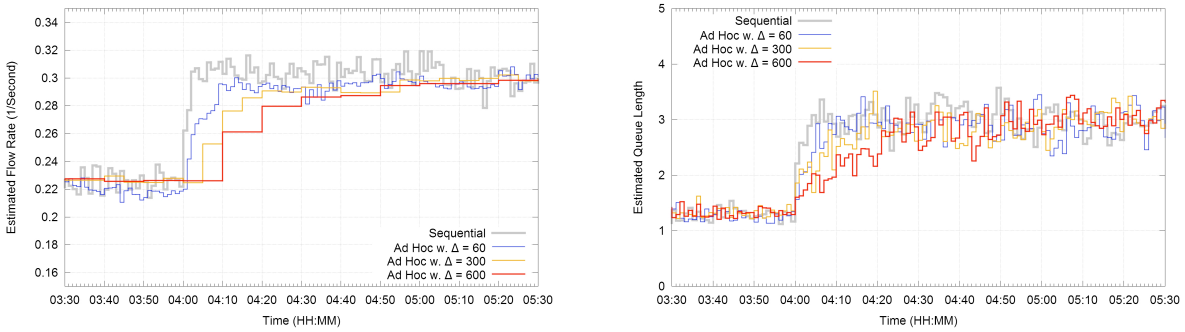


Figure 4: Estimated Arrival Rate across Link 10 and Mean Queue Length at Node 4 under Case 2

3 ITERATIVE AD HOC QUEUEING SIMULATION METHOD

This section proposes a solution to the delayed-response problem by reassigning a new meaning to the values computed during simulation executions. These values are considered as representations of the corresponding current system states, rather than future-state predictions as in the original ad hoc method. The details are described in the following subsections, followed by a discussion of the potential livelock issue.

The proposed iterative ad hoc queueing simulation method inherits most components from the original one, including the partitioning, the local simulation models, the information aggregation, and the rollback-based optimistic synchronization mechanism. These common parts are briefly summarized to provide a comprehensive view of the method without the focus deviating from the new design.

3.1 Partitioning and Local Simulation Model

As in the original ad hoc method, a queueing network of interest is partitioned into subnetworks of various sizes and shapes with the possibility that these subnetworks may overlap with each other. Every LP models one subnetwork and uses the discrete-event simulation technique to construct its local simulation model. The simulation input is the state information of incoming links to the corresponding modeled subnetwork, and the output is that of all modeled links. The specifics regarding the link state information will soon be revisited in the next subsection.

3.2 Information Sharing

Similar to the original ad hoc method, the shared link states are represented in a high-level abstraction instead of using the exact time points of job arrivals/departures. Specifically, LPs exchange estimated flow rates every Δ seconds where an estimated flow rate of a link is computed by reversing the mean interarrival time over the last Δ seconds on that particular link. For example, let $t_i \leq t_{i+1} \leq \ldots \leq t_{j-1}$ denote the arrival times within a time interval of interest. Then, the estimated flow rate r is the reciprocal of the mean interarrival time, i.e., $r = (j - i) / (t_{j-1} - t_{i-1})$. However, generating individual arrival times out of r is not straightforward because the information of the respective interarrival-time distribution is not maintained, nor is the correlation relationship among those arrivals. Here, it is assumed that the arrivals on one link form a Poisson process with the rate equal to the corresponding estimated flow rate.

Unlike the original ad hoc approach, this iterative ad hoc method requires LPs to share link information in a way that state changes are reflected to others as soon as possible. This is accomplished by imposing that an estimated flow rate over a given time period must be used to reconstruct the link during that particular time interval. Specifically, consider a value v as the flow rate over $[t, t + \Delta)$ on some link l. The LPs modeling link l as an incoming link have to use v (along with the assumption about the corresponding arrival process) to generate arrivals during $[t, t + \Delta)$.

This design guarantees that simulations do not progress backwards since the value over $[t, t + \Delta)$ from one LP affects, at earliest, the simulations of $[t, t + \Delta)$ carried out by other LPs. Or, in the terminology of distributed simulations, the "lookahead" value is zero. Zero lookahead commonly raises the concern of deadlocks, which can be eliminated by optimistic synchronization. However, livelocks are possible because LPs may fall into a loop within which they keep rolling back each other. Avoiding such livelocks requires careful design of local simulation models; this will be revisited in Section 3.6.

3.3 Information Aggregation

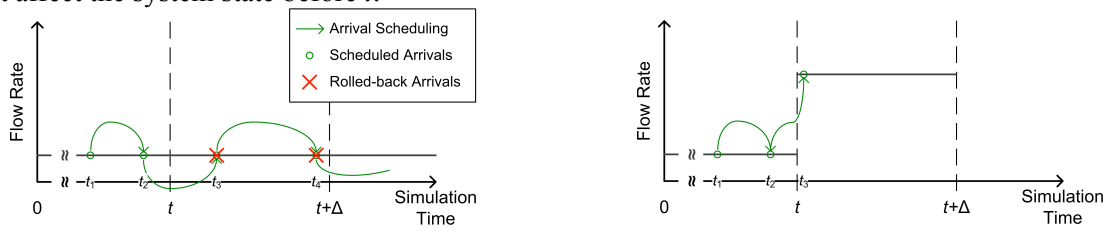
This iterative ad hoc method retains the original information aggregation mechanism—each state variable (i.e., the estimated flow rate of one link over a certain time interval) is affiliated with a data model so that aggregating various estimates is equivalent to randomly generating a sample out of the model. Data models are constructed using the KDE method with Gaussian kernels.

3.4 Optimistic Synchronization and Rollbacks

The optimistic synchronization in this iterative ad hoc method builds upon three principles: 1) intuitive practices, 2) simple implementation/maintenance, and 3) statistical validity. While the last one is the fundamental idea to determine the necessity of a rollback (referred to as the "rollback criterion"), the former two are pervasive throughout the design. For example, when a desired flow rate is unavailable, the requesting LP uses the most current rate of the same link rather than an arbitrary value, by assuming that the link state has not changed. Another example concerns the system state restoration for nonstationary Poisson processes; this will be revisited soon.

The rollback criterion involves a statistical test, which evaluates a used value against all the shared, estimated rates. Specifically, the confidence interval for the mean rate estimate serves as an acceptable range: $\bar{r} \pm t_{n-1,1-\alpha/2} \times S_r / \sqrt{n}$ where $\bar{r}$, S_r^2, and n are the sample mean, sample variance, and sample size, respectively. The significance level α is set to 0.005. The critical value $t_{n-1,1-\alpha/2}$ is based on the Student's t distribution with $n - 1$ degrees of freedom. If the used value falls outside the range, it is rejected and a rollback is triggered. Compared to the original design, this method introduces an additional parameter (i.e., the degrees of freedom) in order to adapt for various variations due to different sample sizes.

In response to rollbacks, LPs perform system state restoration. Since modeling input links involves nonstationary arrival processes, additional state information (other than flow rates) is needed. In typical discrete-event queueing simulations, processing a current arrival event includes scheduling a new arrival event, the timestamp of which relies on some "future information," i.e., in this design, future flow rates. If any pertaining future rate is later proved incorrect, a rollback is triggered; Figure 5 depicts such a situation where an LP is rolled back for using an underestimated flow rate during $[t, t + \Delta)$. Since resetting system state removes all the arrivals beyond t, an initial arrival has to be generated on every input link. The generating process has to consider the elapsed time from t_2 (when the last arrival occurred) to t as part of the interarrival time. Note that the new arrival must come after t because a rollback targeting at $[t, t + \Delta)$ cannot affect the system state before t.



(a) Before System State Restoration (b) After System State Restoration

Figure 5: Arrival Scheduling during Rollbacks

The above issue is simplified in ad hoc queueing network simulations because the arrivals on input links are modeled as nonstationary Poisson processes. Two well-known methods for generating nonstationary Poisson arrivals are the thinning method (Lewis and Shedler 1979) and the inversion method (Çınlar 1975). The thinning method, an acceptance-rejection algorithm, requires the upper bound on the flow rate function, which is generally unavailable. Also, this method may be inefficient when the acceptance rate is low. On the other hand, the inversion method generates the arrivals times $\{t_i\}$ using a sequence of Poisson arrival times at rate 1 $\{t_i'\}$ and the expectation function of the rate function $\Lambda(t) = \int_0^t \lambda(y)dy$, as follows:

$$1)\ u \sim \mathrm{U}(0, 1),\ 2)\ t_i' = t_{i-1}' - \ln(u),\ \text{and}\ 3)\ t_i = \Lambda^{-1}(t_i').$$

We adopt this method because it is practical and easy to implement with one additional variable for t_{i-1}' and an array data structure for Λ^{-1} (as the rate functions in ad hoc queueing network simulations are step functions).

3.5 Naming—"Iterative" Ad Hoc Queueing Simulation Method

The term "iterative" comes from the iterative methods in computational mathematics. These iterative methods solve the problems that are formulated into the fixed-value problem $f(x) = x$, where f is a function. To find a solution of such a problem, the typical procedure of an iterative method starts with an arbitrary x_0, which is used in f to obtain $f(x_0)$; then, the value $f(x_0)$ is set to x_1. This procedure of $x_{n+1} = f(x_n)$ is repeated until the sequence $\{x_i\}$ converges. The definition of convergence varies; it can be that the difference between the last two numbers in the sequence are either zero or within a designated scale of error.

A similar phenomenon of iteration can be observed in this iterative ad hoc approach. Figure 6 depicts an example queueing network partitioned into two parts, each containing one node. The LPs modeling node 0 generate the state information for link B and request that of link A; by contrast, those modeling node 1 use the information for link B to produce that for link A. The relationship between the desired and the shared information can be captured by functions: F_0 for the LPs simulating node 0 and F_1 for node 1. Consider $A_{[t,\,t+\Delta)}$ as the state of link A with respect to the time period $[t,\, t + \Delta)$, and similarly $B_{[t,\,t+\Delta)}$ for link B; the relations can be written down as $B_{[t,\,t+\Delta)} = F_0(A_{[t,\,t+\Delta)})$ and $A_{[t,\,t+\Delta)} = F_1(B_{[t,\,t+\Delta)})$. Combining these two yields $A_{[t,\,t+\Delta)} = F_1(F_0(A_{[t,\,t+\Delta)}))$, which has the form $f(x) = x$ where f is $F_1 \circ F_0$.

3.6 Avoidance of Potential Livelocks

To prevent livelocks in this iterative ad hoc method, it is essential to comprehensively understand a physical system before building local simulation models and designing the ad hoc components, especially the rollback criteria. A livelock situation arises when two or more LPs are involved in a loop in which their shared values keep invalidating each other's simulation inputs. That is, these LPs roll back each other successively so that, from a global view, the entire simulation execution does not show forward progress. Although the zero-lookahead feature allows LPs to "bring back" others to the same simulation time, this feature cannot be blamed for livelocks. Instead, the causes are incorrect simulation models and unrealistic rollback criteria. The former one is analogous to the classical livelock problem where the application/model itself is not appropriately defined; we will not delve into this well-known issue. Instead, here we focus on the latter, which is specific to the ad hoc approach.

A feasible, legitimate rollback criterion must take into consideration the characteristics of state variables, such as randomness. For example, for a state variable with possible values ranging across some continuous space (i.e., a continuous stochastic variable), its affiliated rollback criterion needs to be flexible with regard to evaluating the "correctness" of a value. A value should be considered correct if it is within a certain range. This idea applies to discrete stochastic variables as well. However, the consequence would be more severe for continuous random variables because the probability of a continuous random variable equal to any arbitrary value is zero. This implies that, given an LP that has used a value v for some input link, another LP modeling the link is highly improbable to generate v as a state measure for the link. Hence, the LP using v is rolled back. The following example illustrates such a situation based on the queueing network in Figure 6.

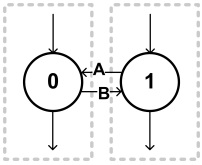


Figure 6: 2-Node Open Queueing Network

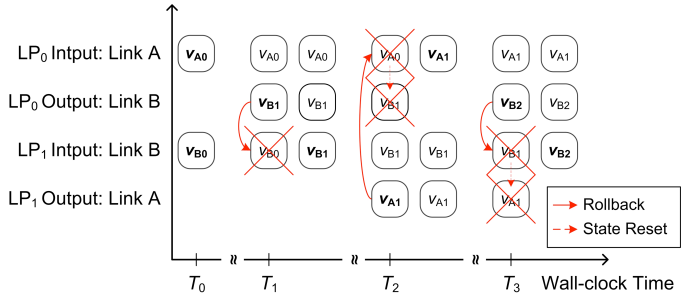


Figure 7: An Example of a Livelock Situation in Modeling the 2-Node Open Queueing Network in Figure 6

The example concerns a simulation of the queueing network in Figure 6 with LP_0 modeling the left part (i.e., node 0 and link B) and LP_1 covering the right one. LP_0 requires the state information of link A as input to its local simulation model, and this information is shared by LP_1. Similarly, LP_1 relies on LP_0 for link B. The link states are measured by the flow rates estimated over Δ seconds, and the LPs must use the exact value their corresponding LP generates. Considering a time interval $[t, t + \Delta)$, the process used by the two LPs to reach an agreement on the rates is depicted in Figure 7; clearly, they fail and fall into a livelock. The detailed process is as follows:

1. At wall-clock time T_0, both LP_0 and LP_1 have not generated the flow rates for their corresponding links with respect to $[t, t + \Delta)$. As a consequence, they adopt arbitrary values: LP_0 applies v_{A0} for link A and LP_1 uses v_{B0} for link B.
2. At time T_1, LP_0 produces the estimated flow rate of link B, v_{B1}, which rolls back LP_1.
3. Then at T_2, LP_1 observes the flow rate of link A being v_{A1} after using v_{B1} for link B. As a consequence, LP_0 is rolled back and its shared state information about link B, v_{B1}, is revoked.
4. At time T_3, a similar situation occurs: LP_0 derives a new value for link B, v_{B2}, due to the usage of v_{A1}. Rolling back LP_1 results in the simulation ending up in a situation resembling that at T_1.

This loop of rollbacks is anticipated to continue because the flow-rate estimates are continuous stochastic variables; it is highly unlikely that $v_{Ai} = v_{Ai+1}$ (nor $v_{Bi} = v_{Bi+1}$) for any integer $i \geq 0$.

4 EXPERIMENTS AND RESULTS

This section evaluates the iterative ad hoc queueing simulation method with three experiments. The first two (Sections 4.2 and 4.3) revisit those in Sections 2.1 and 2.2 but use the new iterative ad hoc method. The third one in Section 4.4 loads the previous tandem network with heavy traffic. All three are to show that the method is effective in terms of identifying system transients. The metrics for evaluation include the flow rates of input links and the mean queue lengths at nodes. The results are compared with the sequential counterparts.

4.1 Welch's *t* Test

The following experiments use Welch's t test to statistically assess the (null) hypothesis that the mean values of two samples are equal. This test modifies the well-known Student's t-test to consider that two samples may have unequal variances. Its statistic t and the degrees of freedom v are defined by Equation (1) where $\overline{X}_i$, S_i^2, and N_i are the i-th sample mean, sample variance, and sample size, respectively. Given a significance level α (e.g., 0.01), the hypothesis is rejected if the derived p-value is below α; the p-value for a two-tailed test based on Student's t-distribution is $1 - (F_v(|t|) - F_v(-|t|))$ where F_v denotes the respective distribution function.

$$t = \left(\overline{X}_1 - \overline{X}_2\right) \Bigg/ \sqrt{\frac{S_1^2}{N_1} + \frac{S_2^2}{N_2}} \text{ and } v = \left(\frac{S_1^2}{N_1} + \frac{S_2^2}{N_2}\right)^2 \Bigg/ \left(\frac{S_1^4}{N_1^2(N_1 - 1)} + \frac{S_2^4}{N_2^2(N_2 - 1)}\right) \quad (1)$$

The two samples in the subsequent hypothesis tests are 1) the data from the ad hoc runs and 2) those from the sequential counterparts. Regardless of the metric, the size of the ad hoc sample (i.e., N_1) is always 10, same as the number of replicate ad hoc runs, and that of the sequential sample (i.e., N_2) is 100. This part of the configuration is the same as that described in Section 2.

4.2 Experiment 1: Partially-Bidirectional Tandem Network

This experiment concerns the network in Figure 2(a) with all the configurations from Section 2.1. The simulation results based on the iterative ad hoc method are shown in Figure 8. Regardless of the Δ value, the ad hoc simulations perform well on capturing the estimated flow rates and mean queue lengths. The revealed choppiness in estimates is anticipated due to the randomness in the simulation models. Further-

more, it is expected that a large Δ may weaken the real-time response as the changes are averaged out across the entire update periods. However, such issue is insignificant in this experiment.

Figure 9 plots the *p*-values from Welch's *t* tests on the same two measurements of interest. These results reaffirm the good performance of the iterative ad hoc method because the vast majority of the *p*-values are above the critical level $\alpha = 0.01$. Some exceptions include those at simulation times 3:54, 4:16, 4:28, and 4:32; they are rather insignificant because 1) the one at time 3:54 occurs just prior to the arrival rate increase at time 4:00 and 2) they are sparse across the entire simulation.

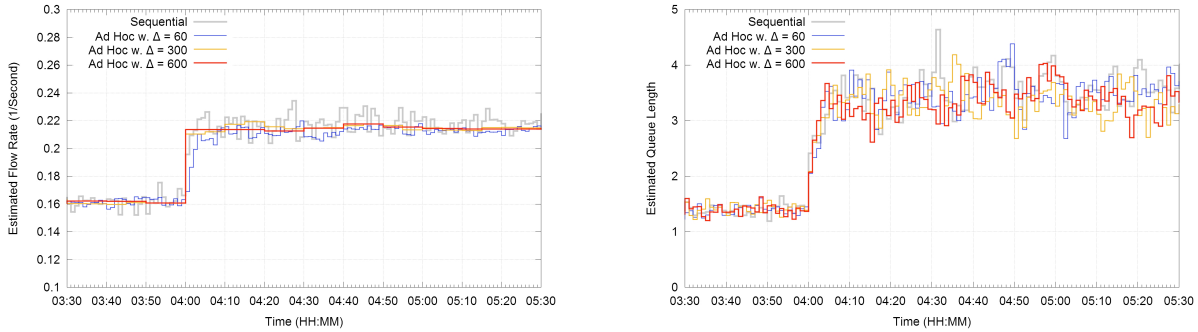
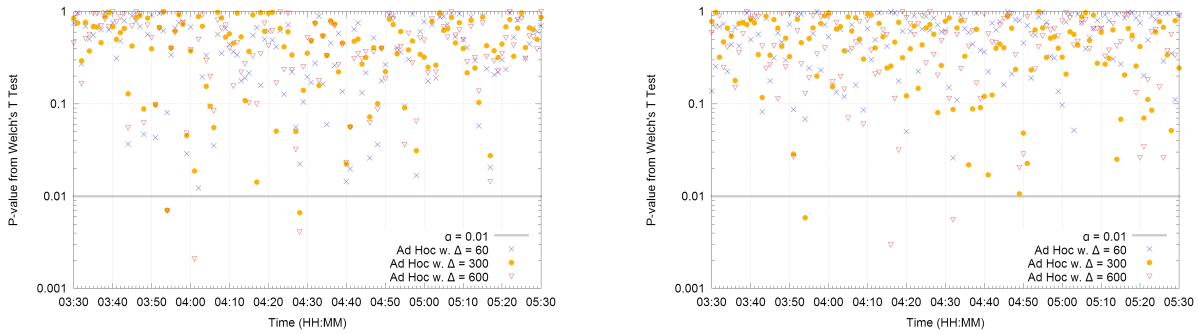


Figure 8: Estimated Arrival Rate across Link 10 and Mean Queue Length at Node 4 in Experiment 1



(a) Estimated Arrival Rate across Link 10

(b) Estimated Mean Queue Length at Node 4

Figure 9: *P*-Values from Welch's *t* Tests in Experiment 1

4.3 Experiment 2: Completely-Bidirectional Tandem Network with Moderate Traffic

This experiment extends that in Section 2.2 by replacing the original ad hoc method with the new one. The network of interest is in Figure 2(b) and the results are in Figure 10. Here (and afterwards), the flow-rate metric is skipped as it leads to the same conclusion as the queue-length metric—that is, the iterative ad hoc method performs well. Nevertheless, again, sporadic hypothesis rejections occur but they are not major, as those in the previous experiment.

4.4 Experiment 3: Completely-Bidirectional Tandem Network with Heavy Traffic

This experiment intends to show that the iterative ad hoc method is capable of promptly responding to sudden, massive state changes. The network of interest is depicted in Figure 2(b) and the system transients are introduced by increasing the external arrivals to each node as depicted in Figure 11. This makes node 3 saturated—an extremely busy server with its queue building up. Node 4 is also saturated with its steady-state traffic intensity reaching approximately 0.97. Since the queues grow rapidly, the system transient period is configured to last for only 30 minutes. This allows the ad hoc method to demonstrate that it can also capture state changes in the opposite direction (i.e., decrease in flow rates).

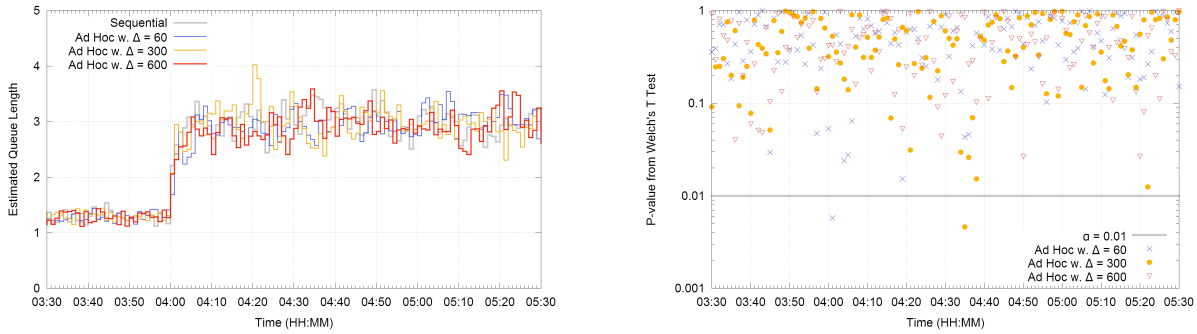


Figure 10: Point Estimates and *p*-Values on Estimated Mean Queue Length at Node 4 in Experiment 2

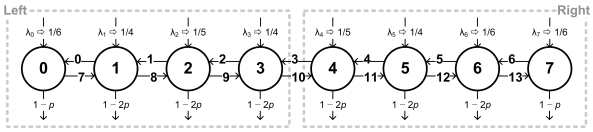


Figure 11: An 8-Node Completely-Bidirectional Tandem Network with Heavy Traffic

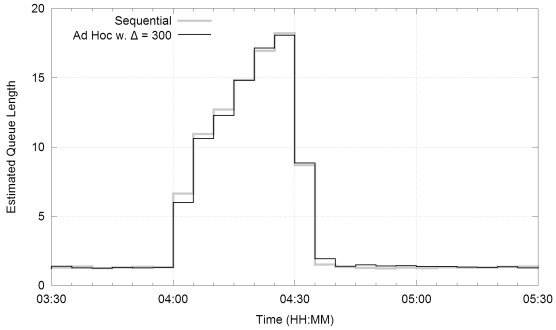


Figure 12: Estimated Mean Queue Length at Node 4 in Experiment 3

The simulation results affirm that the iterative ad hoc method is competent in this heavy-traffic scenario; see Figure 12 for the estimated mean queue length at node 4. For clarity, Δ is set to a moderate value (Δ = 300), leaving off the jagged results from small Δ. Further, the respective *t* tests indicate insignificant differences between the ad hoc and sequential simulations (the figure is not shown due to space constraints).

5 CONCLUSIONS

By extending the work by Huang et al. (2012), which studies the prediction accuracy of steady-state metrics, this paper demonstrated the competitiveness of ad hoc distributed simulation on modeling short-term system dynamics in opening queueing networks. First, we conducted two experiments with increases in flow rates during simulation executions to argue that the original ad hoc method in the previous work fails to immediately reflect system transients. The extent of the delayed response relates to the length of the update period Δ. The failures are due to the fact that the state updates shared by individual LPs are regarded as predictions of future system states, rather than reflections of the current state.

To address the delayed-response issue, we proposed an iterative ad hoc method along with detailed discussions on the design of the relevant components as well as the potential livelock issue arising in certain simulation models. The new method was empirically evaluated by three experiments that demonstrated the potential of the proposed iterative ad hoc approach to capture system dynamics. The evaluation was based on point estimation, which includes not only the direct comparison of the averaged statistics but also Welch's *t* test to provide more compelling and comprehensive statistical evidence.

ACKNOWLEDGMENTS

This research was supported by NSF Grant EFRI-0735991 and AFOSR Grant FA9550-13-1-0100.

REFERENCES

Allen, G. 2007. "Building a Dynamic Data Driven Application System for Hurricane Forecasting." In *Computational Science – ICCS 2007*, Edited by Y. Shi, G. D. v. Albada, J. Dongarra, and P. M. A. Sloot, 1034–1041. Berlin: Springer Berlin Heidelberg.

Chaturvedi, A., A. Mellema, S. Filatyev, and J. Gore. 2006. "DDDAS for Fire and Agent Evacuation Modeling of the Rhode Island Nightclub Fire." In *Computational Science – ICCS 2006*, Edited by V. N. Alexandrov, G. D. v. Albada, P. M. A. Sloot, and J. Dongarra, 433–439. Berlin: Springer Berlin Heidelberg.

Çınlar, E. 1975. *Introduction to Stochastic Processes*. Englewood Cliffs, NJ: Printice-Hall.

Cortial, J., C. Farhat, L. J. Guibas, and M. Rajashekhar. 2007. "Compressed Sensing and Time-Parallel Reduced-Order Modeling for Structural Health Monitoring Using a DDDAS." In *Computational Science – ICCS 2007*, Edited by Y. Shi, G. D. v. Albada, J. Dongarra, and P. M. A. Sloot, 1171–1179. Berlin: Springer Berlin Heidelberg.

Darema, F. 2004. "Dynamic Data Driven Applications Systems: A New Paradigm for Application Simulations and Measurements." In *Computational Science – ICCS 2004*, Edited by M. Bubak, G. D. v. Albada, P. M. A. Sloot, and J. Dongarra, 662–669. Berlin: Springer Berlin Heidelberg.

Davis, W. J. 1998. "On-Line Simulation: Need and Evolving Research Requirements." In *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, Edited by J. Banks, 465–518. New York: Wiley.

Douglas, C. C., R. A. Lodder, J. D. Beezley, J. Mandel, R. E. Ewing, Y. Efendiev, G. Qin, M. Iskandarani, J. Coen, A. Vodacek, M. Kritz, and G. Haase. 2006. "DDDAS Approaches to Wildland Fire Modeling and Contaminant Tracking." In *Proceedings of the 2006 Winter Simulation Conference*, 2117–2124.

Farhat, C., J. G. Michopoulos, F. K. Chang, L. J. Guibas, and A. J. Lew. 2006. "Towards a Dynamic Data Driven System for Structural and Material Health Monitoring." In *Computational Science – ICCS 2006*, Edited by V. N. Alexandrov, G. D. v. Albada, P. M. A. Sloot, and J. Dongarra, 456–464. Berlin: Springer Berlin Heidelberg.

Fujimoto, R., D. Lunceford, E. Page, and A. M. Uhrmacher. 2002. "Grand Challenges for Modeling and Simulation." Dagstuhl Seminar Report 350, Schloss Dagstuhl, Dagstuhl, Germany.

Fujimoto, R., M. Hunter, J. Sirichoke, M. Palekar, H. Kim, and W. Suh. 2007. "Ad Hoc Distributed Simulations." In *Proceedings of the 21st International Workshop on Principles of Advanced and Distributed Simulation*, 15–24.

Huang, Y.-L., C. Alexopoulos, M. Hunter, and R. Fujimoto. 2012. "Ad Hoc Distributed Simulation Methodology for Open Queueing Networks." *Simulation* 88(7): 784–800.

Hunter, M., H. K. Kim, W. Suh, R. Fujimoto, J. Sirichoke, and M. Palekar. 2009. "Ad Hoc Distributed Dynamic Data-Driven Simulations of Surface Transportation Systems." *Simulation* 85(4): 243–255.

Hunter, M., J. Sirichoke, R. Fujimoto, and Y.-L. Huang. 2009. "Embedded Ad Hoc Distributed Simulation for Transportation System Monitoring and Control." In *Proceedings of the 2009 INFORMS Simulation Society Research Workshop*, 15–19.

Kamrani, F. and R. Ayani. 2007. "Using On-Line Simulation for Adaptive Path Planning of UAVs." In *Proceedings of the 11th IEEE International Symposium on Distributed Simulation and Real-Time Applications*, 167–174.

Lee, E. A. 2008. "Cyber Physical Systems: Design Challenges." In *Proceedings of the 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing*, 363–369.

Lewis, P. A. W., and G. S. Shedler. 1979. "Simulation of Nonhomogeneous Poisson Processes by Thinning." *Naval Research Logistics Quarterly* 26(3): 403–413.

Low, M. Y. H., K. W. Lye, P. Lendermann, S. J. Turner, R. T. W. Chim, and S. H. Leo. 2005. "An Agent-Based Approach for Managing Symbiotic Simulation of Semiconductor Assembly and Test

Operation." In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, 85–92.

Mandel, J., L. S. Bennethum, M. Chen, J. L. Coen, C. C. Douglas, L. P. Franca, C. J. Johns, M. Kim, A. V. Knyazev, R. Kremens, V. Kulkarni, G. Qin, A. Vodacek, J. Wu, W. Zhao, and A. Zornes. 2005. "Towards a Dynamic Data Driven Application System for Wildfire Simulation." In *Computational Science – ICCS 2005*, Edited by V. S. Sunderam, G. D. v. Albada, P. M. A. Sloot, and J. J. Dongarra, 632–639. Berlin: Springer Berlin Heidelberg.

Ye, T., H. T. Kaur, S. Kalyanaraman, and M. Yuksel. 2008. "Large-Scale Network Parameter Configuration Using an On-Line Simulation Framework." *IEEE/ACM Transactions on Networking* 16(4): 777–790.

AUTHOR BIOGRAPHIES

YA-LIN HUANG received her Ph.D. in Computer Science from the Georgia Institute of Technology in 2013, and her M.S. in Network Engineering and B.S. in Computer Science & Information Engineering from National Chiao Tung University in 2007 and 2005, respectively. Her Ph.D. thesis work focused on distributed simulations, discrete-event simulations, and simulation methodologies. Prior to this, she had four-year research experience on Internet telephony, primarily on the computer network interoperability among IPv4, IPv6, and NAT networks. Her e-mail is huangyl@cc.gatech.edu.

CHRISTOS ALEXOPOULOS is a Professor in the H. Milton Stewart School of Industrial and Systems Engineering at the Georgia Institute of Technology. His research interests are in the areas of simulation, statistics, and optimization of stochastic systems. He is a member of INFORMS and an active participant in the Winter Simulation Conference, having been proceedings co-editor in 1995, associate program chair in 2006, and a member of the Board of Directors since 2008. He is also an area editor of the ACM Transactions on Modeling and Computer Simulation. His e-mail address is christos@isye.gatech.edu, and his Web page is www.isye.gatech.edu/~christos.

MICHAEL HUNTER is an Associate Professor at the School of Civil and Environmental Engineering at the Georgia Institute of Technology. His primary teaching and research interests are in transportation operations, design and safety, specializing in signal control, traffic simulation, and real-time arterial corridor operations. Dr. Hunter obtained his Ph.D. and M.S. in Civil Engineering from the University of Texas at Austin in 2003 and 1994. He obtained his B.S. in Civil Engineering from Rensselaer Polytechnic University in 1992. He has also worked in private industry conducting traffic impact studies, signal timing projects, freeway operation evaluations, toll plaza analyses, etc.

RICHARD FUJIMOTO is a Regents' Professor and the Chair of the School of Computational Science and Engineering at the Georgia Institute of Technology. He received the Ph.D. and M.S. degrees from the University of California (Berkeley) in 1980 and 1983 (Computer Science and Electrical Engineering) and B.S. degrees from the University of Illinois (Urbana) in 1977 and 1978 (Computer Science and Computer Engineering). His research is concerned with the execution of discrete-event simulation programs on parallel and distributed computing platforms ranging from mobile systems to supercomputers. This work has spanned several application areas including transportation systems, telecommunication networks, multiprocessor systems, and defense systems. He led the working group that was responsible for defining the time management services for the Department of Defense High Level Architecture (HLA) effort (IEEE Standard 1516). His e-mail is fujimoto@cc.gatech.edu.