# Experiment Scheduling in Faster-than-Real-Time Simulation

Dimosthenis Anagnostopoulos
Dept. of Informatics
University of Athens
Panepistimiopolis, Athens, 15771, Greece
email: dimosthe@di.uoa.gr

## Abstract

*Faster-than-real-time simulation (FRTS) can be used for the performance evaluation of systems behavior in real time, providing significant capabilities for studying systems with a time-varying behavior. FRTS enables model validation through comparing simulation results with the corresponding system observations. However, experimentation proves to be rather demanding, as both delivering output results and ensuring their reliability must be accomplished within a predetermined time frame. Output analysis of system observations and model results and relevant timing issues are discussed. A method is introduced that determines whether it is possible to execute the "optimal" faster-than-real-time experiment, in which case multiple replications are scheduled for execution, or a compromise has to be made between the ability to predict for the long future and the degree of reliability achieved for predictions. FRTS experimental results are also presented to support the effectiveness of the proposed method.*

## 1. Introduction

A real-time simulator is a real-time system where some portions of the environment, or even portions of the real-time system itself, are realized by a simulation model [1]. Evaluating the performance of the system under study in real time is certainly not a trivial task, depending on various issues, such as the system speed (i.e. how often state changes occur, compared to world time) and nature, which determines crucial issues, such as the allowed degree of human interaction with the system. When attempting to reach conclusions for the system behavior in the near future, faster-than-real-time simulation (FRTS) is widely used. In this type of simulation, advancement of simulation time occurs faster than real world time.

Real time systems often have hard requirements for interacting with the human operator or other agents [2]. As making models run faster is the modeler's responsibility, a significant disadvantage is that timing problems are recognized during or even after testing. Researchers have thus pointed out that timing requirements should be addressed at the design phase [3]. Addressing timing problems at this phase, though, does not consider the variability in the time required to execute an experiment, which is caused by real conditions, such as arrival-process distribution parameters, which may be non-stationary. It does not also consider the need for achieving a specific degree of confidence for simulation results, which is calculated during the execution of the experiment. We thus argue that timing issues should also be addressed at the implementation (or execution) phase. The complexity of interacting in real time with the system, in order to conclude on the validity of the model and predictions, should also be addressed both at the methodological level and for domain-oriented applications, e.g. transportation, process-control systems and communication networks [4].

Timing requirements such as the following are considered in FRTS: the model must run faster than the system, system data must be obtained and processed in real time so that the current system state is always known, the model has to be adapted to the current system state without terminating the FRTS experiment, since the system under study may be a time-varying one, and dynamic (i.e. in real time) model modification should be enabled. In addition, model validity must always be ensured and, when simulation results are utilized to ensure validity, they must have the essential statistical properties (e.g. the appropriate sample size and statistical processing). Analysis of simulation results and system data must always be performed within the given time frames. Timing requirements have been addressed by methodological approaches, such as [5], which is employed in this paper. This conceptual FRTS methodology provides extensions to traditional simulation, focusing on real-time interaction and ensuring the validity of simulation results. It consists of the following phases: *modeling, experimentation and remodeling*. The modified FRTS executive is presented in figure 1. The system and the model are under monitoring during experimentation. Data depicting their consequent states are obtained within predetermined time intervals of equal length, called *auditing intervals*. In case the model state deviates from the corresponding system state, remodeling is invoked. This may occur due to system modifications, which can involve its input data, operation parameters and structure [5]. In these cases, remodeling adapts the model to the current system state. This is accomplished without terminating the real time experiment, since no recompilation is performed. When model modifications are completed, experimentation resumes. Remodeling can also be invoked when deviations (expressed through appropriate performance measures) are indicated between the system and the model due to the stochastic nature of simulation, even when the system parameters/ components have not been modified. In case simulation results (predictions for the near future) are considered to be valid, an additional phase, called *plan scheduling,* is invoked to take advantage of them [5]. However, as depicted in figure 1, plan scheduling activities are not considered as a part of the FRTS environment.

Experimentation encompasses monitoring, which is the activity where system and model states are obtained and stored while the model is executed, and auditing. Auditing examines if simulation results are reliable, based on system observations, as well as whether the system has been modified during the last observation. The dynamic system behavior may result in critical modifications in the system input data, operation parameters and structure. Structure variability, in particular, has been studied either at the methodological level [6], [7] or in domain-oriented research approaches, such as for computer networks [4]. To conclude about such modifications, specific attributes of both systems are put under monitoring. The corresponding variables are referred as *monitoring variables.* Monitoring variables should be considered at a conceptual level, not according to the conventional, single-valued definition of program variables, as later discussed. Auditing examines the monitoring variables corresponding to the same real time points (i.e. the current system state and simulation predictions for this time point) and concludes for the evolution of the system and the model [5].

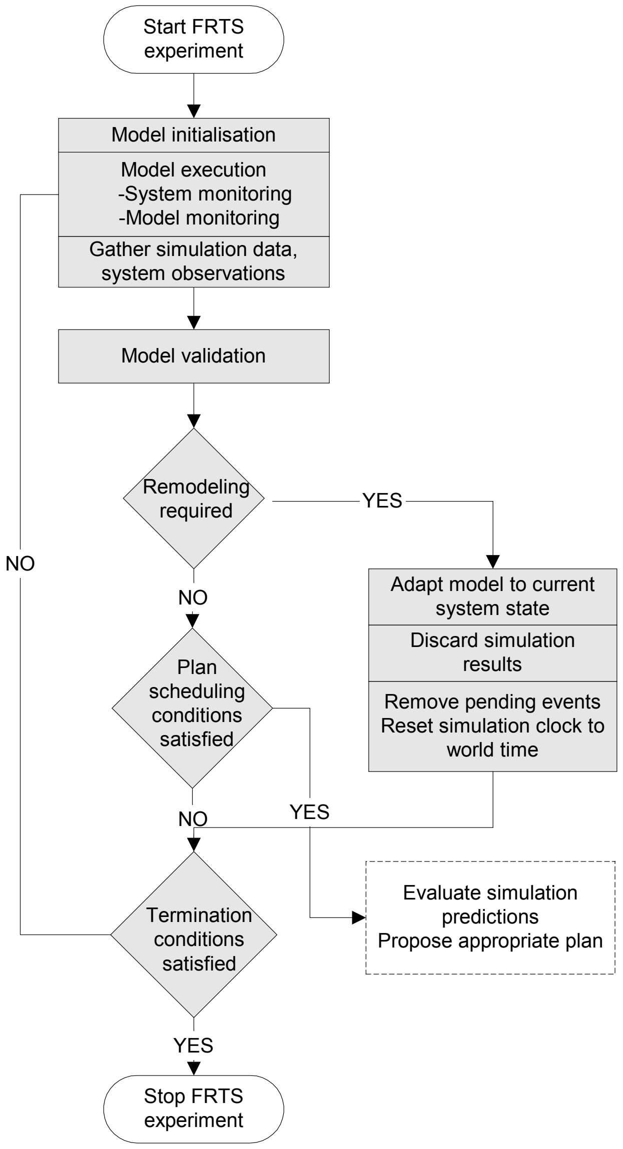

Start FRTS experiment
Model initialisation
Model execution
-System monitoring
-Model monitoring
Gather simulation data, system observations
Model validation
Remodeling required
YES
NO
Adapt model to current system state
Discard simulation results
Remove pending events
Reset simulation clock to world time
NO
Plan scheduling conditions satisfied
NO
YES
Evaluate simulation predictions
Propose appropriate plan
Termination conditions satisfied
YES
Stop FRTS experiment

**Figure 1: FRTS executive**

An example is presented in figure 2, where the evolution of the system and the model is depicted at the two horizontal axes. Real time points are noted as $t_i$. The states of the system and the model at point $t_i$ are noted as $R_i$ and $S_i$. When at $t_x$ the model predicts the system state at $t_n$ (simulation time is equal to $t_n$) we use the notation $Sim(t_x)= t_n$. States $S_x$ and $R_n$ are thus compared during auditing at time point $t_n$.

In this paper, we address timing issues of FRTS, that is, we state the problems encountered and propose a method for executing experiments conforming to the real-time requirements. The proposed experiment scheduling method is independent of the execution environment, as it applies to both sequential and parallel execution. However, in the latter case, processing units must be of the same hardware and have the same load, as latter discussed. The term FRTS (or FRTS experiment) denotes the entire simulation process, involving model execution, monitoring, auditing and remodeling activities, performed in consecutive auditing intervals. FRTS experiments are terminated when termination conditions are fulfilled (e.g. if FRTS cannot be achieved or world time reaches a predetermined value). A single experiment is executed within each auditing interval and consists of one or multiple replications. We consider the case of terminating simulations (there is no relationship with the abovementioned termination conditions), which is more common yet more difficult to carry out, as discussed in section 2. Thus, each single experiment is a terminating simulation, imposing that multiple replications must be completed within the given time frame to reach predictions. A discussion concerning output analysis issues encountered in FRTS resides in section 2. Timing issues as well as a method for scheduling experiments and dealing with them are presented in section 3. Section 4 includes experimental results and conclusions reside in section 5.

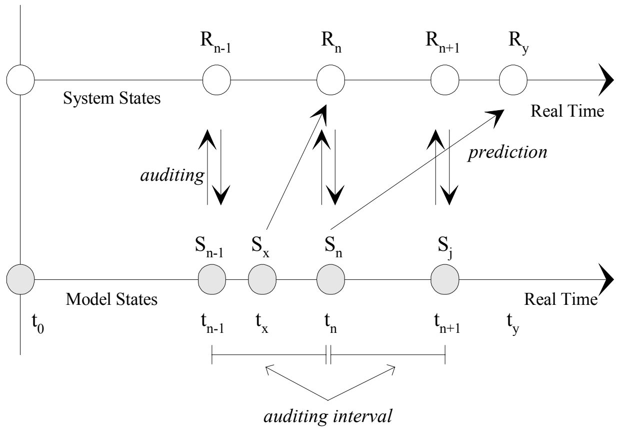


**Figure 2: Experimentation in FRTS**

## 2. Output analysis

In this section, we discuss whether FRTS should be considered as a terminating or non-terminating simulation and how simulation data and system observations are compared (that is, when system and model data are obtained and how this comparison should be performed). In general, the options available in designing and analyzing simulation experiments depend on the type of simulation at hand, which may be either terminating or non-terminating, depending on whether there is an obvious way to determine run length. A terminating simulation is one for which there is a "natural" event that specifies the length of each run (replication), whereas a non-terminating simulation is one for which there is no such event [8]. A performance measure for such a simulation is a steady state parameter if it is a characteristic of the steady state distribution of some output process. The authors argue that experiments in FRTS should be widely considered as terminating simulations due to the following:

1. Stochastic processes for more real systems do not have steady-state distributions, since the characteristics of the system change over time [8].
2. Observations from a particular replication are clearly not IID.
3. Experiment termination, which occurs when simulation predictions are produced for specific future time points, can be considered as the terminating event.

Considering FRTS as a terminating simulation imposes that we make *n* independent replications, each one terminated by a "natural" event that is scheduled for execution when simulation time reaches the predicted time points. Beginning with the same initial conditions, replications produce *n* observation sequences. Suppose that $MV_1$, $MV_2$, ... $MV_k$ are the monitoring variables used for the purposes of a FRTS experiment. Each variable $MV_i$ is practically distinguished into two separate values $MV_i.r$ and $MV_i.s$ for the system and the model, respectively. $MV_i.r$ is calculated as a function of either a single-valued variate (performance measure or system parameter) or multiple observations $R_{i1}, R_{i2}, \ldots$ from the system, in which case $MV_i.r = f_i(R_{i1}, R_{i2}, \ldots)$. $MV_i.s$ can also be calculated as a function of either a single-valued variate or an output stochastic process. $MV_i.s$ can thus be a function of *n* stochastic processes:

$$S_{i11}, S_{i12}, S_{i13}, S_{i14}, \ldots, S_{i1k_1}$$
$$S_{i21}, S_{i22}, S_{i23}, S_{i24}, \ldots, S_{i2k_2}$$
$$S_{in1}, S_{in2}, S_{in3}, S_{in4}, \ldots, S_{i1k_n}$$

In FRTS, the number of observations per replication is not the same, since simulation ends at a specific simulation time, no matter the current status of the system entities (e.g. how many customers have been serviced in a GI/G/s system). Replication results are thus extracted from $k_1, k_2, \ldots, k_n$ observations. Considering that the output process of each replication *j* produces a single statistical sample $S_{ij}$, then:

$$S_{ij} = g(S_{ij1}, S_{ij2}, \ldots, S_{ijk_j})$$
$$MV_i.s = sum(S_{i1}, S_{i2}, \ldots, S_{in})/n$$

The issue of comparing system observations and simulation results has been thoroughly examined in the literature and various methods have been proposed depending on the nature of the problem. Law and Kelton provide an excellent review [8]. A confidence interval approach based on independent data proves to be a well-suited solution. In our case, specific conditions that must be taken under consideration when selecting an appropriate statistical method are that there are *n* independent sets of data from the model but only one from the system and that data from the system are not correlated with data from the model.

We focus on determining model and system data to be compared. Suppose there are no previous predictions, as in the case when remodeling has just been performed, and simulation execution starts at point $t_n$. Since comparisons between model and system data are performed during auditing, it is reasonable to determine the predicted time points using the auditing interval (*AudInt*) as a time unit. When the model runs faster than the system, it reaches conclusions for *p* intervals ahead within the auditing interval $[t_n, t_{n+1}]$ (figure 3). In this example, $Sim(t_x) = t_n$, $Sim(t_y) = t_{n+1}$, $Sim(t_z) = t_{n+2}$.

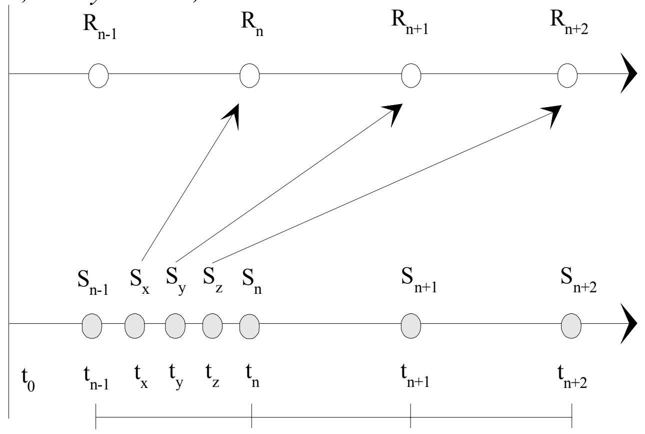
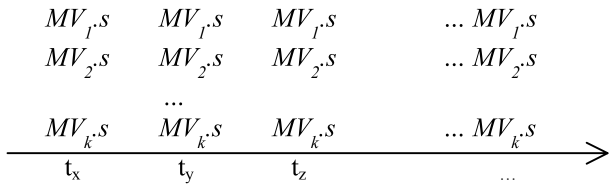


**Figure 3: Reaching conclusions for *p* intervals ahead**

To perform comparisons, model results are obtained at the corresponding real time points. Considering that *n* independent replications have to be made, model output $MV_i.s$ has to be calculated at $t_x$, $t_y$ and $t_z$. Note that $t_x$, $t_y$ and $t_z$ are the time points where all replications must be completed for the corresponding predicted intervals, i.e. at $t_x$, the "slowest" replication produces its output for $R_n$. There have to be *p* sets of output results $MV_1.s, MV_2.s, ..., MV_k.s$ produced within a single auditing interval, as presented in figure 4. The discussion for calculating the values of $MV_i.s$ thus concerns each of the future states predicted within the intervals $[t_n, t_x]$, $[t_x, t_y]$, $[t_y, t_z]$, etc. In fact, typically, we could also refer to *p* simulations, where *n* replications are made for each of them. In the next section, we discuss determining *n* for each specific replication. Except from the first interval (i.e. $[t_n, t_x]$), where the initial model state has to be identical to the corresponding system state, in all other cases, the model state at the end point of the previous interval serves as the initial state for the next one.



$MV_1.s$ $MV_1.s$ $MV_1.s$ ... $MV_1.s$
$MV_2.s$ $MV_2.s$ $MV_2.s$ ... $MV_2.s$
...
$MV_k.s$ $MV_k.s$ $MV_k.s$ ... $MV_k.s$
$t_x$
$t_y$
$t_z$
...

**Figure 4: Output results produced for each predicted interval**

We have previously discussed the terminating event of each of the *n* replications. In fact, there have to be *p* terminating events for each "complete" (i.e. covering all predicted intervals) replication. The terminating event is the one examining when at time point $t_j$ simulation time reaches the end point of the predicted interval, that is:

$$Sim(t_j) = t_n + i*AudInt, i \in N^*, 1 \le i \le p$$

The above conclusions are reached on the basis that a fixed number or replications is required to reach predictions for each of the future system states, which is rather a common practice. In this way, reliability is ensured with the same amount of experimental results for each of the predicted states. The conclusions reached concerning the execution of an experiment within a specific auditing interval are used in the following to determine an effective way for scheduling and executing *n* replications, with respect to the timing requirements imposed.

## 3. Timing issues and proposed method

Suppose the current time is $t_n$ and we want to reach predictions for $t_k$ within the current auditing interval. Then at some future point $t_j$, condition $Sim(t_j) = t_k$ must be fulfilled. This introduces the issue of determining the prediction interval length. It is required that $t_j < t_k$, but the question is how much $t_k - t_j$ should be. An obvious answer is that "predictability" must be dictated by the requirements imposed to simulation (e.g. in case of a control processes) and by the constraints imposed by the nature of the system (e.g. how fast the system evolves). On the one hand, it is evident that having (reliable) predictions for long intervals ahead would be desirable. However, the degree of reliability cannot be ensured and usually tends to decrease for very long predictions, suffering from the fact that the characteristics of real systems tend to change over time. On the other hand, timing restrictions for experimentation are increased when predicting long ahead, especially in cases where the system evolves nearly as fast as simulation.

To discuss in depth timing issues, a specification of the overall control scheme of FRTS is provided in figure 5, in terms of a real-time data flow diagram (DFD) focusing on the control process and the time-consuming activities [9]. Suppose that simulation reaches conclusions for at least $p$ auditing intervals ahead of real time. If current time is $t_n$ and $Sim(t_n) = t_n$ (i.e. simulation is re-initiated) then, at some future point $t_j$, the following conditions must be fulfilled:

$$Sim(t_j) = t_n + p*AudInt$$
$$t_j < t_n + AudInt$$

To achieve this, simulation activities that definitely must be accomplished before $t_j$, as depicted in figure 5, are: model initialization, system monitoring and model monitoring. The process that accepts experiment parameters from the user, as discussed in paragraph 2.1, is accomplished prior to the initiation of the FRTS experiment. System monitoring is executed continuously and concurrently with model execution during the auditing interval. Thus, timing requirements involve model initialization and model monitoring (i.e. model execution and storing output results), which consume time equal to $T_{Init}$ and $T_{Exec}$, respectively.

Auditing, on the other hand, does not add any time overhead to the experiment conducted within the current interval, since it is executed only when the auditing interval has elapsed. There are, however, requirements for auditing duration, as model execution is temporarily paused, so that conclusions for model reliability are reached with minimum time overhead before resuming experimentation. The same stands for remodeling, as it is executed after auditing and only when necessary. Evidently, model execution is also paused during remodeling. Remodeling discards all previous simulation results and the FRTS experiment is re-initiated from the current real time point. We examine the detailed execution of monitoring, auditing and remodeling activities during sequential intervals using figure 6.

Since $Sim(t_x) = t_n$ and $Sim(t_y) = t_{n+1}$, auditing compares $S_x$ with $R_n$ at $t_n$. Assuming that auditing indicates that the corresponding model and system states, expressed through monitoring variables *MV.s* and *MV.r*, do not coincide, predictions (i.e. $S_y$) are discarded and remodeling is activated to restore consistency. The time required is denoted as $T_{Audit}$ and $T_{Remodel}$. The following are noted:

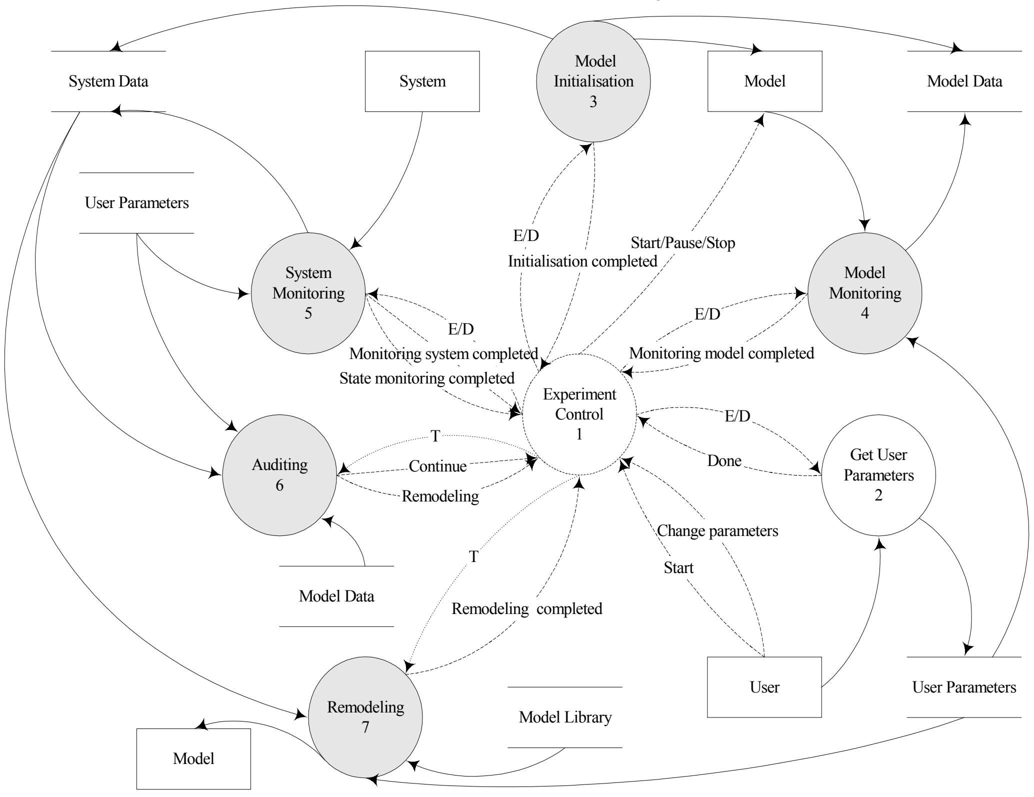


**Figure 5: DFD specification of FRTS control**

1. System monitoring process is re-initiated after the completing of the previous auditing interval without waiting for the completion of auditing and monitoring activities. In this way, we avoid ignoring significant system changes during the execution of these activities and we ensure that auditing is always performed at the expected time point (i.e. $t_n + i{*}AudInt$).
2. Model monitoring/ execution is actually performed during a shorter interval than *AudInt* (i.e. reduced by $T_{Audit}$ and $T_{Remodel}$). The duration of both auditing and remodeling activities should thus be considered when determining the available time frame for model execution. However, $T_{Audit}$ and $T_{Remodel}$ affect the next interval – not the one producing the data auditing and remodeling examine.

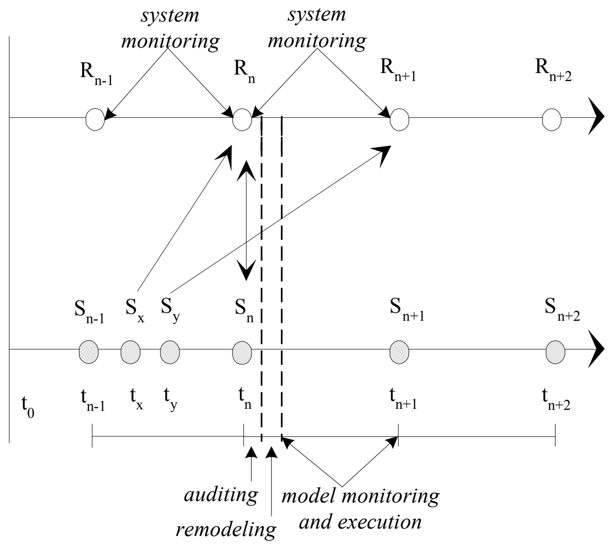


**Figure 6: Execution of monitoring, auditing and remodeling within the auditing interval**

Formulating the timing requirements, we consider the worst-case scenario, which is described as follows:

1. Both auditing and remodeling have been performed, consuming time equal to $T_{Audit}$ and $T_{Remodel}$. Simulation time must thus be re-initiated from the starting point of the current auditing interval.
2. Simulation must reach predictions for *p* auditing intervals ahead of this time point.

The essential condition for accomplishing FRTS within each auditing interval is the following.

$$T_{Audit} + T_{Remodel} + T_{Init} + T_{Exec} \leq AudInt \qquad (1)$$

$T_{Init}$ is practically equal to 0. $T_{Exec}$ is the time required to reach predictions for the *p* intervals ahead. As $T_{Audit}$ and $T_{Remodel}$ concern the previous auditing interval, they are already known when a new experiment is initiated within the current interval. We need to perform *n* independent replications. Suppose that $T_i$ is the time required to execute replication *i*. We define $b_i = T_i \,/\, PredictedInterval$, which expresses how many times is the system slower than simulation in each replication. Then,

$$T_{Exec} = \sum_{i=1}^{n} T_i = \sum_{i=1}^{n} b_i(pAudInt) = pAudInt \sum_{i=1}^{n} b_i$$

$$T_{Audit} + T_{Remodel} + pAudInt \sum_{i=1}^{n} b_i \leq AudInt$$

$$\sum_{i=1}^{n} b_i \leq \frac{AudInt - T_{Audit} - T_{Remodel}}{pAudInt} \qquad (2)$$

Condition (2) is the one determining if it is possible to perform FRTS on a software and hardware platform offering the capability to run each replication $1 \,/\, E(b_i)$ times faster than the system ($E(b_i)$ is the expected value of $b_i$ ). Based on this, the following method is proposed for designing the execution of multiple replications within a single auditing interval. This method combines theoretical and experimental results to examine whether it is possible to execute *n* replications, each one producing results for *p* auditing intervals ahead of the starting point, within the given time frame. A number of initial replications ($n_o$) is used to estimate $\mu=E(b_i)$.

1. Make $n_o$ replications of the simulation and gather statistics (note that $n_o$ should be large enough to provide a good estimation for $\mu$).
2. Estimate $\mu$ and $\sigma^2=Var(b_i)$ as follows: $\mu = \dfrac{1}{no}\displaystyle\sum_{i=1}^{n_o} b_i$ ,

   $$s^2 = \frac{1}{n_o - 1}\sum_{i=1}^{n_o}[b_i - E(b_i)]^2 .$$

3. Suppose that $B_n = \displaystyle\sum_{i=1}^{n} b_i$ . Assume there is distribution function for $b_i$. We know that $b_1$, $b_2$, … $b_n$ are IID, thus $s^2$ is an estimate of $\sigma^2$. Using the central limit theorem, $(B_n - n\mu)/\sigma\sqrt{n} \sim N(0,1)$. We want the probability that *n* replications can be executed to be equal to *a* (e.g. 0.9). Thus, $P[B_n \leq \dfrac{AudInt - T_{Audit} - T_{Remodel}}{pAudInt}] \geq a.$

   Suppose that $\lambda = \dfrac{AudInt - T_{Audit} - T_{Remodel}}{pAudInt}$ .

   Then, $P[B_n \leq \lambda] = P[(B_n - n\mu)/s\sqrt{n} \leq (\lambda - n\mu)/s\sqrt{n}\,]$ $= P[Z \leq (\lambda - n\mu)/s\sqrt{n}\,] \geq a$ . Suppose that *k* is a value for which $P[Z \leq k] = a$. Thus, to perform $n$ replications within the time frame given with probability $a$, the following condition must be fulfilled:

$$(\lambda - n\mu)/s\sqrt{n} \geq k \qquad (3)$$

4. If condition (3) can be fulfilled, execute the rest $n\text{-}n_O$ concurrent replications. When results from all replication are produced for each interval, update statistics (using also the results from the $n_O$ replications) and store results. If the time consumed during simulation exceeds *AudInt*, immediately terminate simulation.
5. If condition (3) cannot be fulfilled, a decision has to be made concerning decreasing the number of remaining replications ($n\text{-}n_O$) so that $(\lambda - n\mu)/s\sqrt{n} \geq k$ or decreasing the number of predicted intervals.
6. Execute the remaining replications concurrently. When results from all replications are produced for each interval, update statistics (using also the results from the $n_O$ replications) and store results. Meanwhile, if the time consumed exceeds *AudInt*, immediately terminate simulation.

Using this method, we cannot be certain that results from all replications will be produced within the given time frame - in the general case, this would not be possible due to the stochastic nature of simulation. In any experiment that simulation exceeds the time frame, this specific experiment has to be immediately terminated. To ensure that results for the most immediate intervals are produced with an acceptable degree of reliability, which depends on the number of replications, concurrent execution of all replications is suggested. In this way, reliability of results for the most immediate intervals, for which we are mostly concerned as they are soon to be compared with system data, is not endangered. On the other hand, if some replications were not to produce results in the case of a sequential execution, this would have an impact on reliability, and potentially cause remodeling. This must be avoided, since all predictions would then have to be discarded and remodeling time would have to be consumed. After all, if predictions for "remote" intervals were not obtained, they can always be produced within the next auditing interval. The authors argue that this principle should also be adopted when considering whether to decrease the number of remaining replications or the number of predicted intervals. Thus, to fulfill $(\lambda - n\mu)/s\sqrt{n} \geq k$, the value for $p$ must be chosen so that:

$$p \leq \frac{AudInt - T_{Audit} - T_{Remodel}}{AudInt(ks\sqrt{n} + n\mu)} \qquad (4)$$

Based on the above, the proposed scheme for executing $n$ independent replications is depicted in figure 7.

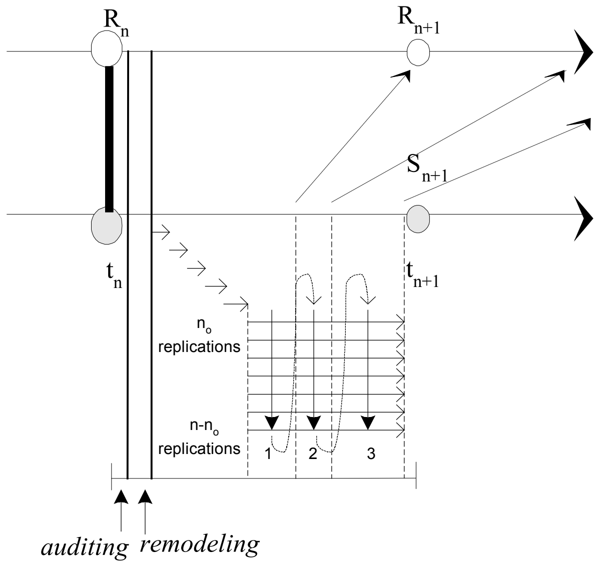


**Figure 7: Proposed scheme for executing *n* replications**

## 4. Experimental results

The proposed experiment scheduling method is independent of the execution environment, as it applies to both sequential and parallel execution. Parallel simulation is most suitable for executing the remaining $n\text{-}n_O$ replications, to ensure that results for the most immediate intervals are produced with an acceptable degree of reliability (step 6). However, processing units must be of the same hardware and have the same load, in order to run each replication $1 / E(b_i)$ times faster than the system. Furthermore, when executing independent, sequential simulations on different processors (also known as *replicated trials* approach), a drawback is that each processor must contain sufficient memory to hold the entire simulation [10]. In the case of large-scale systems, $T_{Audit}$ and $T_{Remodel}$ could be considerably increased, but this should not be a problem as they refer to the previous auditing interval and are thus known when a new experiment is initiated in the current interval. A potential problem could emerge due to unreliable communications, especially in a LAN or WAN computing environment, preventing the delivery of some replication results. Reaching results on the basis of the rest $n_O\text{–}1$ or *n-1 replications* (provided that these are available) could be an acceptable solution to this problem.

Applying the proposed method also requires that monitoring capabilities be provided over $T_i$, that is, the time required to execute replication $i$. Calculating $T_i$ is rather trivial and, as numerous output variates are calculated within each replication, $T_i$ can be considered as an additional output variate, which does not impose any limitation to the proposed method. We simulated the following network queuing models, considering a probability of 95% for being able to execute *n* replications within the auditing interval (*a= 0.95*). The average delay in the queue (*AvgDelay*) is used as an output variate.

QNM1. a M/Gamma/s queuing network (using Kendall's notation in classical queuing theory), that is, with Exponential(b) interarrival times, Gamma(α,β) service times and *s* servers, for s=64, 128, 256, with various values for b and α, while β = 2.0

QNM2. a queuing network with *s* stations (multiple-server/multiple-queue system) where jobs are randomly routed to stations, with Exponential(b) interarrival times, Gamma(α,β) service times, for s=64, 128, 256.

We conducted both sequential and distributed FRTS experiments. In sequential simulation, the execution environment was a Sun Ultra 5 with 1 CPU and 640Mbyte running Solaris 8. In distributed simulation, the environment was a set of 5 Sun Ultra 5 with 1 CPU and128Mbyte running Solaris 8, under conditions of equal load. Auditing and prediction intervals are equal to 5.0sec/15.0sec and 10.0sec/30.0sec, respectively, so that predictions had to be reached for 3 auditing intervals ahead (*p=3*). Experiments were conducted according to the following algorithm:

| | |
|---|---|
| 1. | Execute 100 repetitions of the experiment |
| 1.1 | -Sequential: Execute $n_o$ replications and calculate statistics<br>-Distributed: Execute at least $n_o$ replications, equally distributed among processors, and calculate statistics |
| 1.2 | Calculate n so that condition (3) is fulfilled with probability a |
| 1.3 | -Sequential: Execute the remaining $n-n_o$ concurrent replications and calculate statistics<br>-Distributed: Execute at least $n-n_o$ equally distributed replications and calculate statistics |
| 1.4 | Determine if the experiment is successful (i.e. if the time required to execute n replications is less than the auditing interval) |
| 1.5 | Calculate the precision increase (*pri*) in results after n replications compared to the corresponding precision after $n_o$ replications (*pri*) |
| 2. | Calculate the percentage of successful experiments (*srate*) and the average precision increase |

After each completion of $n_o$ and $n\text{-}n_o$ replications, we build a 90% confidence interval for *avgD*. We use the average value of the confidence-interval half-length ($\delta(n,a)= t_{n-1,1-\frac{a}{2}}\sqrt{\frac{S^2(n)}{n}}$) divided by the point estimate $\overline{avgD}(n)$ as a measure of the precision of the confidence interval [8]. The overall algorithm was executed 3 consecutive times, to provide a better estimate of the percentage of successes and compare it against probability *a*. Results for sequential and distributed simulation successful experiments, for various combinations of interarrival times and service times, are given in table 1 for NQM1 and in table 2 for NQM2. In most cases, the results obtained are better than expected ($\alpha$), indicating that *n* replications were accomplished as required. The increased success ratio can be due to the low variance in the replication time duration.

Executing *n* replications within the given time frame, the average increase in the precision of simulation results is depicted in table 3. The precision obtained after $n_o$ and *n* replications is equal to $\delta(n_o,a)/\overline{avgD}(n_o)$ and $\delta(n,a)/\overline{avgD}(n)$, respectively. Precision is increased when $pri<1$, where $pri=(\delta(n,a)/\overline{avgD}(n))/(\delta(n_o,a)/\overline{avgD}(n_o))$. Precision increase results with distributed simulation are presented in table 3 for NQM1. As *avg(pri)* ranges from 0.616 to 0.846, an increase of 0.154 – 0.384 is obtained, contributing significantly to the reliability of predictions.

## 5. Conclusions

We argued that reaching conclusions in FRTS for a specific number of intervals ahead should be considered as a terminating simulation, which imposed that multiple replications have to be made. Timing issues for performing a large number of replications within the given time frame were examined. The method introduced facilitates designing and carrying out experiments consisting of *n* independent replications, where *n* is dynamically determined at runtime, as the duration of replications cannot be efficiently calculated at the design phase. Adjusting the proposed method for achieving a specified precision for simulation results is an open issue for further research.

**Acknowledgements**:
The author thanks the anonymous referees for their comments on this paper.

| s | interarrival and service times (b, α) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 64 | 0.01, 0.608 | 0.01, 0.624 | 0.01, 0.640 | 0.02, 1.216 | 0.02, 1.248 | 0.02, 1.280 | 0.03, 1.824 | 0.03, 1.872 | 0.03, 1.92 |
| seq. | 0.973 | 0.956 | 0.99 | 0.95 | 0.953 | 0.986 | 0.99 | 0.966 | 0.993 |
| distr. | 0.98 | 0.966 | 0.976 | 0.966 | 0.97 | 0.956 | 0.976 | 0.953 | 0.96 |
| 128 | 0.01, 1.216 | 0.01, 1.248 | 0.01, 1.280 | 0.02, 2.432 | 0.02, 2.496 | 0.02, 2.56 | 0.03, 3.648 | 0.03, 3.744 | 0.03, 3.84 |
| seq. | 0.986 | 0.983 | 0.986 | 0.996 | 0.973 | 0.986 | 0.99 | 0.983 | 0.97 |
| distr. | 0.966 | 0.99 | 0.98 | 0.976 | 0.966 | 0.97 | 0.98 | 0.966 | 0.973 |
| 256 | 0.01, 2.432 | 0.01, 2.432 | 0.01, 2.496 | 0.02, 4.864 | 0.02, 4.992 | 0.02, 5.12 | 0.03, 7.296 | 0.03, 3.488 | 0.03, 7.68 |
| seq. | 0.98 | 0.99 | 0.956 | 0.98 | 0.976 | 0.993 | 0.966 | 0.95 | 0.973 |
| distr. | 0.967 | 0.98 | 0.96 | 0.986 | 0.95 | 0.99 | 0.953 | 0.96 | 0.966 |

**Table 1: *avg(srate)* in sequential and distributed simulation (NQM1)**

| s | interarrival and service times (b, α) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 64 | 0.01, 0.512 | 0.01, 0.576 | 0.01, 0.640 | 0.02, 1.024 | 0.02, 1.152 | 0.02, 1.28 | 0.03, 1.536 | 0.03, 1.728 | 0.03, 1.92 |
| seq. | 0.99 | 0.996 | 0.97 | 0.986 | 0.97 | 0.966 | 0.98 | 0.993 | 0.986 |
| distr. | 0.973 | 0.973 | 0.99 | 0.963 | 0.963 | 0.976 | 0.963 | 0.976 | 0.97 |
| 128 | 0.01, 1.024 | 0.01, 1.152 | 0.01, 1.28 | 0.02, 2.048 | 0.02, 2.304 | 0.02, 2.56 | 0.03, 3.072 | 0.03, 3.456 | 0.03, 3.84 |
| seq. | 0.99 | 0.976 | 0.993 | 0.98 | 0.956 | 0.97 | 0.963 | 0.976 | 0.963 |
| distr. | 0.966 | 0.96 | 0.99 | 0.973 | 0.96 | 0.99 | 0.99 | 0.976 | 0.98 |
| 256 | 0.01, 2.948 | 0.01, 2.304 | 0.01, 2.56 | 0.02, 4.096 | 0.02, 4.608 | 0.02, 5.12 | 0.03, 6.144 | 0.03, 6.192 | 0.03, 7.68 |
| seq. | 0.99 | 0.986 | 0.976 | 0.993 | 0.99 | 0.98 | 0.963 | 0.966 | 0.956 |
| distr. | 0.973 | 0.953 | 0.99 | 0.963 | 0.956 | 0.97 | 0.953 | 0.97 | 0.95 |

**Table 2: *avg(srate)* in sequential and distributed simulation (NQM2)**

| s | interarrival and service times (b, α) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 64 | 0.01, 0.608 | 0.01, 0.624 | 0.01, 0.640 | 0.02, 1.216 | 0.02, 1.248 | 0.02, 1.280 | 0.03, 1.824 | 0.03, 1.872 | 0.03, 1.92 |
| *avg(pri)* | 0.713 | 0.720 | 0.706 | 0.663 | 0.658 | 0.671 | 0.616 | 0.616 | 0.622 |
| 128 | 0.01, 1.216 | 0.01, 1.248 | 0.01, 1.280 | 0.02, 2.432 | 0.02, 2.496 | 0.02, 2.56 | 0.03, 3.648 | 0.03, 3.744 | 0.03, 3.84 |
| *avg(pri)* | 0.789 | 0.768 | 0.773 | 0.684 | 0.692 | 0.702 | 0.644 | 0.647 | 0.656 |
| 256 | 0.01, 2.432 | 0.01, 2.432 | 0.01, 2.496 | 0.02, 4.864 | 0.02, 4.992 | 0.02, 5.12 | 0.03, 7.296 | 0.03, 3.488 | 0.03, 7.68 |
| *avg(pri)* | 0.820 | 0.846 | 0.843 | 0.697 | 0.702 | 0.741 | 0.648 | 0.673 | 0.674 |

**Table 3: Average precision increase after the execution of $n$-$n_0$ replications (NQM1)**

## References

1. Ghosh K., K. Panesar, R. M. Fujimoto, and K. Schwan. "PORTS: A Parallel, Optimistic, Real-Time Simulator," in *Proceedings of 1994 Workshop on Parallel and Distributed Simulation,* IEEE Computer Press, 1994, pp.24-31
2. Cubert R., P. Fishwick, "OOPM: An Object-Oriented Multimodeling and Simulation Application Framework, *Simulation*, vol. 70, no. 6, 1998, pp. 379-395
3. Burns A., A. Wellings, "Hrt-Hood: A structured design method for hard real time systems", *Real Time Systems*, vol. 6, 1994, pp. 73-114
4. Anagnostopoulos D., M. Nikolaidou, "An Object-Oriented Modeling Approach for Dynamic Computer Network Simulation", *International Journal of Modeling and Simulation*, vol. 21, no. 4, 2001
5. Anagnostopoulos D., M. Nikolaidou, P. Georgiadis, "A Conceptual Methodology for Conducting Faster-Than-Real-Time Experiments", *SCS Transactions on Computer Simulation*, vol. 16, no. 2, 1999, pp. 70-77
6. Barros F. J., "Modeling Formalisms for Dynamic Structure Systems", *ACM Transactions on Modeling and Computer Simulation - TOMACS*, vol. 7, no. 4, 1997, pp. 501-515
7. Zeigler B. P., H. Praehofer, T. Kim, *Theory of Modeling and Simulation (second edition)*, Academic Press, 2000
8. Law A.M., W.D. Kelton, *Simulation Modeling and Analysis (third edition),* McGraw-Hill, 2000
9. Goldsmith S., "A Practical Guide To Real-Time Systems Development", Prentice Hall, 1993
10. Fujimoto R., "Parallel Discrete-Event Simulation", Communications of the ACM, vol. 33, no. 10, 1990, pp. 30-52