Facilitating the Analysis of a UK National Blood Service Supply Chain Using Distributed Simulation

Navonil Mustafee [1§], Simon J.E. Taylor[1], Korina Katsaliaki[2], Sally Brailsford[3]

[1]School of Information Systems, Computing and Mathematics
Brunel University
Uxbridge UB8 3PH, UK

[2] Middlesex University Business School
Middlesex University
London, N11 1QS, UK

[3]School of Management
University of Southampton
Southampton, SO17 1BJ, UK

[§]Corresponding author (contact details)
School of Information Systems, Computing & Mathematics
Brunel University, Uxbridge, Middlesex UB8 3PH
Tel: +44 (0)1895 267674
Fax: +44 (0)1895 251686
Email: navonil.mustafee@brunel.ac.uk

Manuscript:
Submitted 01 Aug 2007
Revision request 11 Mar 2008
Revision submitted 24 Jul 2008
Accepted 22 Oct 2008

Abstract

In an attempt to investigate blood unit ordering policies, researchers have created a discrete-event model of the UK National Blood Service (NBS) supply chain in the Southampton area of the UK. The model has been created using Simul8, a commercial-off-the-shelf discrete-event simulation package (CSP). However, as more hospitals were added to the model, it was

discovered that the length of time needed to perform a single simulation severely increased. It has been claimed that distributed simulation, a technique that uses the resources of many computers to execute a simulation model, can reduce simulation runtime. Further, an emerging standardized approach exists that supports distributed simulation with CSPs. These CSP Interoperability (CSPI) standards are compatible with the IEEE 1516 standard *The High Level Architecture*, the defacto interoperability standard for distributed simulation. To investigate if distributed simulation can reduce the execution time of NBS supply chain simulation, this paper presents experiences of creating a distributed version of the CSP Simul8 according to the CSPI/HLA standards. It shows that the distributed version of the simulation does indeed run faster when the model reaches a certain size. Further, we argue that understanding the relationship of model features is key to performance. This is illustrated by experimentation with two different protocols implementations (using Time Advance Request (TAR) and Next Event Request (NER)). Our contribution is therefore the demonstration that distributed simulation is a useful technique in the timely execution of supply chains of this type and that careful analysis of model features can further increase performance.

Keywords – Discrete-event simulation, supply chain simulation, commercial simulation packages, distributed simulation, high level architecture, standards.

1. Introduction

The fact that, in the UK at least, relatively few people donate blood has placed considerable pressure on those managing the supply of blood to hospitals. Katsaliaki and Brailsford [1]

describe efforts that have been made in the UK to model and simulate such a supply chain between the National Blood Service supply centre and hospitals in the Southampton area of the UK. The model was built using the commercial off the shelf discrete-event simulation package (CSP) Simul8. These CSPs are widely used in industry by Operations (Operational) Researchers and Management Scientists to facilitate model building and simulation [2, 3]. Unfortunately, it was discovered that as the model grew in size with the number of hospitals being modeled, the simulation runtime increased drastically. For example, it took around 36 hours to simulate a supply chain consisting of one blood supply centre and four hospitals for one year on a PC (1.7 GHz processor with 1GB RAM). With around 16 hospitals in the Southampton area, this presented a major obstacle to their research.

It has been claimed that distributed simulation is a technique that, in addition to interoperating geographically remote models, promoting data hiding and model reusability, can use the resources of many computers to execute a simulation model, can reduce simulation runtime [4]. Further, led by researchers at Brunel University, an emerging standardized approach exists that supports distributed simulation with CSPs such as Simul8 [5]. These CSP Interoperability (CSPI) standards are compatible with the IEEE 1516 standard *The High Level Architecture*, the defacto interoperability standard for distributed simulation. This paper therefore describes a joint attempt by Southampton and Brunel researchers to investigate the degree to which distributed simulation can reduce the execution time and/or increase the practical size of the NBS supply chain simulation by creating a distributed version of the CSP Simul8 according to the CSPI/HLA standards. As will be seen, the contribution of this paper is therefore the demonstration that distributed simulation using the CSPI/HLA standard is a useful technique in

the timely execution of supply chains of this type and that careful analysis of model features can further increase performance.

This paper is structured as follows. Section 2 gives a background to the NBS supply chain and briefly describes the conventional NBS model. Distributed simulation and the CSPI/HLA standards are introduced in section 3. The distributed approach to the NBS simulation is presented in Section 4. Section 5 presents experimentation and results. Section 6 discusses the findings and Section 7 draws the paper to a close.

2. Modeling the National Blood Service Supply Chain: A Brief Overview

The UK National Blood Service (NBS) is a part of the National Health Service Blood and Transplant (NHSBT) organization. NHSBT was formed on 1st October 2005 by the merger of the National Blood Authority (NBA) (which manages the NBS, Bio Products Laboratory and the International Blood Group Reference Laboratory) and UK Transplant [6]. The NBS is responsible for collecting blood through voluntary donations, testing the blood for ABO and Rhesus grouping and infectious diseases such as HIV, processing the blood into around 120 different products (of which the main three are Red Blood Cells, plasma and platelets), storing the stockpile and transferring excess stock between different NBS centers, and finally issuing the different blood products to the hospitals as per their needs. The NBS infrastructure consists of 15 Process, Testing and Issuing (PTI) centers which together serve 316 hospitals across England and North Wales. Each PTI Centre thus serves around 20 hospitals. Our NBS simulation has been modeled with inputs from the Southampton PTI Centre.

Blood products are stored in PTI Centres until they are requested by the hospitals served by that Centre. A hospital places an order for blood products when its inventory falls below a predetermined order point, or when rare products not held in stock are requested for particular patients. Hospitals normally receive their orders daily and the blood remains in the hospital bank until it is cross-matched (tested for compatibility) for a named patient. It is then placed in "assigned inventory" for that patient for a fixed time after the operation. If it is not used, it is returned to "unassigned inventory" and can be cross-matched again for another patient. On average a unit will be cross-matched four times before it is used or outdated. In practice, however, only half of the cross-matched blood is actually transfused. This clearly represents a huge potential for savings since the cost of a single unit of RBC is around £132.

In the original study, researchers modeled the NBS Southampton PTI and included only RBC and platelets, which together comprise 85% of issues and are the chief source of wastage and shortages. The model was originally built using the CSP Simul8 and is described in Katsaliaki and Brailsford [1]. We present a brief overview of the model to give orientation to the work in this paper. There are two main categories of entities in the model; items and orders. Items are the individual blood units (RBC and platelets) delivered from the NBS Centre to the hospitals in a one-way direction, since returns of products are not allowed. Orders are placed by the hospital blood bank managers to the NBS Centre for blood products. Requests are matched with items according to their characteristics (attributes) as in a Kanban system and delivered as appropriate.

While the model runs, data such as the day and time of placing an order, the requested blood product (RBC or platelets), the amount by blood group, etc. are reported to an Excel file. The model advances time in simulated minutes but the hospitals' blood bank stock for placing orders to the NBS PTI is checked only every simulated hour. Likewise, the blood stocks which are ready to be delivered from the NBS PTI centre to the hospital(s) are also checked only once every simulated hour. Additionally, blood products are perishable in nature and it is important to keep an account of their remaining shelf-life. The shelf-life of a blood product is therefore decreased by the minute. Thus Simul8 schedules an event for each unit of RBC or platelet present in the system, at every simulated minute, which brings down the shelf life of the blood product by 1 minute.

The model contains the processes of the NBS PTI Centre, from collection of whole blood to delivery of blood products, and the processes of a single medium-volume hospital. The model captures physicians' requests for blood and the processes whereby the hospital blood bank checks its stock levels and places orders. The order entities and item entities are represented as information flow (hospital orders) and material flow (blood products) respectively. A single supply centre and hospital are shown in Fig. 1. Fig. 2. shows a simplified diagram showing the relationship between four hospitals and one supply centre. This figure also shows that these are running in a single CSP on a single PC. This is our dilemma. As stated in the introduction, this simulation took 36 hours to simulate one year's production. Let us now consider if distributed simulation can provide a faster alternative.

3. Distributed Simulation and COTS Simulation Package Interoperability

To provide context in this paper, distributed simulation can be defined as the distribution of the execution of a single run of a simulation program across multiple processors [4, 7]. To provide a standard approach to distributed simulation, in the year 2000 the IEEE 1516 standard *The High Level Architecture* (HLA) [8] was published (and updated in 2006). In the HLA, a distributed simulation is called a *federation*, and each individual simulator (in our case the combination of a CSP and its model) is referred to as a *federate*. A HLA *Runtime Infrastructure* (RTI) provides facilities to enable federates to interact with one another, as well as to control and manage the simulation. The HLA is composed of four parts: a set of compliance rules [9], the Object Model Template (OMT) [10], the Federate Interface Specification (FIS) [11], and the Federate Development Process (FEDEP) [12]. The FIS is an application interface standard for distributed simulation middleware which defines how federates interact within the federation, and is implemented by an RTI (i.e. federates communicate with one another via an RTI). There are several RTIs available. The OMT provides a common presentation format for HLA federates. The HLA is therefore a general standard for distributed simulation.

The problem of creating distributed simulations consisting of COTS simulation packages (CSPs) using the HLA was first addressed in Straßburger, et al. [13]. CSPs are standalone "black box" packages that expose simple interfaces that are used to control the package and to access the model stored within the package. The main problem is therefore the manner in which the HLA RTI software is interfaced to the CSP. Early HLA associated work either addressed the use of distributed simulation and CSPs within wider contexts or represented individual research projects. For example, the IMS MISSION project attempted to use distributed simulation and

CSPs within large decision support environments in manufacturing supply chains [14, 15, 16, 17, 18, 19]. Individual research projects developed different, but incompatible approaches to the use of the HLA in support of distributed simulation with CSPs [20, 21, 22, 23].

Building on the lessons learnt from this work, as described in Taylor, et al. [5] a standardization movement specifically addressing the problems of HLA-based distributed simulation and CSPs began in 2002. This has led to the development of a suite of CSP Interoperability (CSPI) standards under development by the Simulation Interoperability Standards Organization's CSPI Product Development Group (CSPI PDG). The CSPI PDG's standards are intended to provide guidance on how the specific requirements of HLA-based distributed simulation with CSPs can be supported. This is accomplished by providing a set of Interoperability Reference Models (IRM) that describe different distributed simulation requirements, a set of Data Exchange Representations that are used to define the format of exchanged data, and a set of Interoperability Frameworks that specify the architecture and protocol used to integrate the CSP with the HLA RTI and exchange data in a time synchronized manner. Currently, there are six IRMs that describe the distributed simulation requirements for a range of scenarios, one Data Exchange Representation (entities) and one Interoperability Framework to support entity exchange (IRM Type I, *Asynchronous Entity Transfer*) [24, 25]. Recent work on CSPI standards include Wang, et al. [26] who study possible implementations of the Type II IRM Synchronous Entity Passing, Taylor, et al. [27] and Gan, et al [21] investigate the use of CSP distributed simulation in engine manufacturing and semiconductor manufacturing respectively. Also, the use of these standards within a semiconductor manufacturing decision support environment is discussed by

Lendermann et al. [28]. We now discuss how these have been used to guide the development of the NBS distributed simulation.

4. Creating the NBS Distributed Simulation

A. Overview

The NBS Distributed Simulation was created using the Defense Modeling and Simulation Offices (DMSO) RTI [29] and extended our earlier work on Simul8/DMSO RTI integration [30] using techniques based on the CSPI standards. The NBS distributed simulation fits the profile of the Type A.1 Interoperability Reference Model (Entity Transfer, T1<T2), i.e. dividing the model into federates results in individual, distributed models of the Southampton NBS PTI and hospitals that interact by exchanging entities. These models run in separate copies of Simul8. Together these form a federation that interact by timestamped messages that represent the interaction of one model part with another (e.g., when an entity leaves one part of a model and arrives at another). Entities are represented using the CSPI Entity Transfer Specification and exchanged using HLA interactions. The complete model, constituted of distributed federates, form our NBS supply chain federation (as shown in Fig. 3).

In this investigation, the interaction between the models/Simul8 and the HLA-DMSO RTI is via an Excel file. For example, entities representing orders are written into the file by Simul8 during the execution of hospital models. The HLA-DMSO RTI, augmented by the CSPI standards, then correctly transfers this information to the NBS model by means of HLA interactions. The

incoming orders from each hospital are collected into their corresponding queues in the NBS model and the orders are matched with the available stock of blood. The resulting matched units are written into the Excel spreadsheet in the NBS federate. This information is then sent to the different hospital models again through HLA interactions. The interactions between the hospitals and the NBS center are sent every 60 minutes of simulated time, provided orders/delivers exist. Thus, although the discrete-event simulation generates orders and deliveries as the model progresses in time, these are only released at specific time-steps. It is to be noted here that this time-stepped information exchange behavior occurs as a result of the blood ordering and delivery policies in place with NBS.

As with most distributed simulations, the NBS models being executed on different computers need a mechanism to synchronize their simulation time and to ensure current ordering of events. In a standalone simulation the event list consists only of events generated internally by the "single" running instance of the model. We term these events as *internal events*. In a distributed simulation each federate also receives events from other federates. We term these events as *external events*. The event list of a distributed federate, therefore, has to correctly time order both the internal events and the external events and execute them without any error. However, due to (1) latencies in the network, (2) different processing requirements for different models, and (3) different hardware configurations of machines, it is not possible to guarantee when external events will arrive at a federate. In order to ensure that in a federation events arrive at each federate in the causally correct order, we utilize the time management services of the HLA [31]. The HLA defines different time management schemes, including *conservative time synchronization* (the federate only advances simulation time when RTI guarantees that it will not

receive any external events in its past) and *optimistic time synchronization* (the federate advances time without any constraint but must be prepared to rollback to a previous state when it receives an event in its past). In this paper we have experimented with the more widely used conservative approach.

The HLA defines several variants of the conservative time synchronization mechanism that can be invoked by a federate to request time advance from RTI. The two most commonly used are Next Event Request (NER) and Time Advance Request (TAR). We have implemented both NER and TAR versions for our Simul8-DMSO RTI integration work to investigate implications to performance. Literature suggests that for a discrete-event simulation, like our NBS simulation, using NER is more appropriate [31, 32]. As will be discussed, however, our simulation performs better using TAR.

B. Simul8- DMSO RTI Integration

Simul8 is a CSP that supports discrete-event simulation and that enables users to rapidly construct accurate, flexible and robust simulations using an easy-to-use visual interface [33]. It includes an internal programming language called "Visual Logic" and provides a Windows COM [34] interface that can be used from within any COM-compliant language to interface Simul8 [35]. For our Simul8-DMSO RTI case study we have developed a CSP Controller Middleware (CCM) that interacts with both Simul8 Professional Edition and the DMSO RTI 1.3-NG to realize a Simul8-based distributed simulation. CCM utilizes the Simul8 Professional COM interface to access the Simul8 simulation engine and its interaction with DMSO RTI is

through the services defined in the HLA interface specification. Before discussion the CCM architecture further, we present below a brief overview of the HLA-defined service groups being used by the CCM.

The HLA interface specification organizes the communication between federates and the RTI into six different service groups [36]. Following the interoperability approach suggested by the CSPI PDG, for our Simul8/DMSO RTI integration we require HLA-defined services defined under the groups:

- Federation Management: RTI Calls for creation and deletion of federation; joining and resigning of federates from the federation; and creation and realization of synchronization points.
- Declaration Management: Calls pertaining to publication and subscription of interactions.
- Object Management: Calls that relate to sending and receiving interactions.
- Time Management: RTI calls required to enable time constraint and time regulation and also to advance the federate simulation clock.

As noted earlier the CCM middleware performs two specific tasks; it communicates with Simul8 through its COM interface and it interacts with DMSO RTI using the HLA interface specification. Each of these two tasks is performed by two distinct components of the CSP controller middleware: the *Simul8 adapter* and the *DMSO RTI adapter*. The communication between these adapters is via Java Native Interface and Jacob technologies [37, 38]. The CCM has a separate implementation for TAR and NER. The architecture of the CSP Controller

Middleware is shown in Fig. 4 and the message exchange protocol followed by both these variants of CCM is given in fig 5 and Fig. 6.

The Simul8 adapter defines methods like *OpenSim(modelFile)*, *RunSimulation(time)*, *getBloodOrdersFromHospital(hospital)* and *introduceEntitiesToHospital(hospital, bloodUnit)* that are invoked by the DMSO RTI adapter to open a Simul8 *modelFile*, run the model to the *time* specified, get blood orders from *hospital* and to introduce entities into the *hospital* respectively. These methods encapsulate both the application logic and the Simul8 COM method calls. For example, method *getBloodOrdersFromHospital(hospital)* has application logic that reads *hospital* order details being output by Simul8 into an Excel file and method *introduceEntitiesToHospital(hospital, bloodUnit)* invokes Simul8 COM method *ExecVL* to set various *bloodUnit* parameters into the running *hospital* model and to schedule events. The Simul8 adapter also calls methods defined in the DMSO RTI adapter like *tellSimulationTimeEnd(time)* and *sendOrderToNBS(hospital, bloodOrder)* to convey to the DMSO RTI adapter that Simul8 has completed processing a model till a defined "safe" *time* (see discussion below) and to transfer the *bloodOrder* collected from the *hospital*. The DMSO RTI adapter methods contain application logic and invoke HLA defined service calls. For example, the method *tellSimulationTimeEnd(time)* has application logic which sets the logical time of the federation to the *time* returned by the method call and *sendOrderToNBS(hospital, bloodOrder)* invokes HLA defined method *sendInteraction* to pass the *bloodOrder* details from respective *hospital* federates to the NBS PTI federate in the form of HLA interactions. It is worthwhile here to mention that it is the RTI adapter that has separate federate logic for NER and TAR implementations (referred subsequently as CCM-NER and CCM-TAR).

To introduce CCM-NER and CCM-TAR protocols (fig 5 and fig 6 respectively), let us first present a discussion of the HLA NER and TAR and how these are used to advance the simulation time of a federate. Both NER and TAR service calls, defined by the HLA standard and implemented by the RTI, are invoked with a time component which represents the logical time the federate wishes to move to. Depending on whether NER or TAR is called by the simulating federate, the time granted to it by the RTI can be different. NER will grant the federate a time that is either less than or equal to the requested time depending on whether external events are present and their timestamps (if external events are present). If an external event exists for the federate with timestamp less than the requested time then the time granted by RTI will be equal to the timestamp of the external event. If no external events exist or an external event with timestamp equal to the requested time is received, then the RTI will grant the federate the requested time. TAR, on the other hand, will grant the simulation federate a time that is exactly equal to the time requested by a federate.

CCM-NER invokes the HLA defined NER method call (*nextEventRequest[timeRequested]*) and CCM-TAR invokes the HLA defined TAR method call (*timeAdvanceRequest[timeRequested]*). Both these service calls have a time argument (*timeRequested*) that specifies the simulation time to which the federate wants to move to. The CCM-NER requests a time from the RTI that is equal to its current logical time + 60 (*timeRequested=logicaltime+60*) or a time that is equal to its previously requested time (*timeRequested=timePreviouslyRequested*) depending on whether the RTI had granted the *timeRequested* by the federate in the preceding NER call or it had granted a time less than the *timeRequested*. CCM-TAR, on the other hand, requests a time from

the RTI that is always equal to its current logical time + 60 (*timeRequested=logicaltime+60*). The NBS PTI center and the hospitals exchange information at every 60 units of simulation time and therefore both CCM-NER (incase *timeRequested* had been granted in preceding HLA NER call) and CCM-TAR request a time equal to the current logical time of the federate + 60 simulation units. The difference with regards to *timeRequested* by CCM-NER and CCM-TAR protocols is because they implement two different HLA synchronization strategies, viz. NER and TAR.

In case of both CCM-TAR and CCM-NER the new time granted to the federate by the RTI is conveyed using the HLA TIME ADVANCE GRANT callback (*timeAdvanceGrant[timeGranted]*). This callback, invoked by the RTI on the federate RTI adapter, carries the time (*timeGranted*) that has been granted by the RTI and is a guarantee that there will be no external events from the rest of the federation before this time. This new "safe" time is conveyed by the RTI adapter to the Simul8 adapter (*newSimulationTime[timeGranted]*) and the simulating federate processes the Simul8 model to this time. This may, in turn, generate other internal or external events. Subsequently, the logical time of the federate becomes equal to this new time (*logicalTime=timeGranted*) and the process of requesting time advancement using NER or TAR starts all over again.

Now we look at how external events are sent across federates in our NBS simulation. As has been said earlier we use HLA interactions to achieve this. When a federate generates an external event the Simul8 adapter of CCM conveys this to the DMSO RTI adapter, which in turn invokes the HLA defined service SEND INTERACTION (*sendInteraction**). Each interaction contains a

time stamp and associated data. These interactions are sent to the RTI to be delivered to the respective federates in the causally correct order. On the receiving end, the RTI delivers the interactions to the DMSO RTI adapter though the RTI callback RECEIVE INTERACTION (*receiveInteraction**). The DMSO RTI adapter of the CCM then forwards the received data to the Simul8 adapter for introduction into the model. The data being exchanged in the federation relate to blood orders and deliveries. In both sendInteraction* and *receiveInteraction**, the superscript "*" indicates that multiple interactions can be sent or received.

5. Experiments and Results

To investigate the performance of NBS distributed simulation using NER time management service (implemented by CCM-NER), NBS distributed simulation using TAR service call (implemented by CCM-TAR), and NBS standalone simulation, we conducted experiments with four different scenarios. Each scenario represented one NBS PTI centre serving one, two, three or four hospitals respectively. The name of the scenario reflected the number of hospitals that the NBS PTI catered for. For example, scenario 2Hospital would mean that 2 hospitals were being served by one NBS center. In the distributed cases, scenario 2Hospital became three separate Simul8 models, each modeling either the NBS PTI center, Hospital1 or Hospital2 and ran on three separate computers. In the standalone case, scenario 2Hospital meant that a single Simul8 model, running on a single PC, modeled the behavior of the NBS center and two hospitals.

Experiments were conducted on Dell Inspiron laptop computers running Microsoft Windows XP operating system with 1.73GHz processors and 1GB RAM with a medium specification desktop

PC to host the RTI rtiexec software. These computers were connected through a 100Mbps CISCO switch and the RTI process (rtiexec.exe) was started on one of the computers. The results of the execution times for each of the scenarios were based on the average of 5 runs. In terms of variance, overall out of 144 results (three scenarios, four experiments and results taken for each month) there were 23 results with a standard deviation of more than 5% of the mean execution time. Apart from month 1 of the standalone 1Hospital experiment (5.84% of the mean), the results with a standard deviation greater than 5% of the mean were for all months of the standalone 4Hospital experiment (ranging from 9.42% to 53.42%, mean 22.71%). The 4Hospital experiments for NER and TAR had 4 results each over 5% in the range 5.01% to 9.56%. Apart from the standalone 4Hospital experiment, a variance of less than 5% was acceptable and due to operating system and networking effects. The variance of the standalone 4Hospital experiment was high but expected due to the large amount of swapping between RAM and virtual memory (see later).

Fig. 7 shows the execution time in seconds for both standalone and distributed approaches for all the four scenarios. The results show that the conventional model with one hospital took approximately 14 minutes to run for a whole simulated year. The run time rose to 78 minutes when the model ran with two hospitals and to approximately 17.5 hours with three hospitals. The addition of the fourth hospital increased the execution time to 35.8 hours. The NER version of the distributed model with one NBS supply centre and one hospital ran in approximately 8.4 hours, with two hospitals in 9.8 hours, with three hospitals in 12.7 hours and with four hospitals in 16.5 hours. The execution time for the TAR version of the distributed model was 7.2, 7.8, 10.3 and 15.5 hours for the 1Hospital, 2Hospital, 3Hospital and 4Hospital scenarios respectively.

Fig. 8 compares the time taken to execute the three versions of the NBS simulation (standalone, distributed-NER [implemented by CCM-NER] and distributed-TAR [implemented by CCM-TAR]), for every consecutive month of the year (1month to 12months) and for each of the four scenarios (1Hospital, 2Hospital, 3Hospital and 4Hospital). The results obtained from scenarios 1Hospital and 2Hospital show that the conventional (standalone) version executes much faster compared to its distributed counterparts. In case of scenarios 3Hospital and 4Hospital both the distributed versions outperform the execution of the standalone model. However, in all the 4 scenarios the TAR-based simulation executes faster compared to the NER-based simulation. There also appears to be an exponential increase of the runtime in the conventional version while increasing the number of hospitals in the model. This is quite a contrast to the substantially smaller and smoother rise in the runtime in case of both NER and TAR versions of the distributed model.

Our experimental methodology generates results by using 5 replications per data point to reduce variance caused by operating system and networking effects. This is in line with approaches in literature. In similar distributed simulation research performed by well known researchers in the field, for example, [39] and [40] do not use any replications at all and just use the results from single runs, and [41] use 5 replications. However, this is an excellent opportunity to improve our analysis. While literature does not report variances, to add to our results we have calculated the standard deviation for all our results. In terms of variance, overall out of 144 results (three scenarios, four experiments and results taken for each month) there were 23 results with a standard deviation of more than 5% of the mean execution time. Apart from month 1 of the

standalone 1Hospital experiment (5.84% of the mean), the results with a standard deviation greater than 5% of the mean were for all months of the standalone 4Hospital experiment (ranging from 9.42% to 53.42%, mean 22.71%). The 4Hospital experiments for NER and TAR had 4 results each over 5% in the range 5.01% to 9.56%. Apart from the standalone 4Hospital experiment, a variance of less than 5% was acceptable and due to operating system and networking effects. The variance of the standalone 4Hospital experiment was high but expected due to the large amount of swapping between RAM and virtual memory.

6. Discussion

From the results we observe:

- for scenarios 1Hospital and 2Hospital the standalone NBS simulation executes faster than its distributed counterparts and for scenarios 3Hospital and 4Hospital the distributed versions out perform the conventional simulation.
- comparing the performance of the distributed versions we see that for each consecutive month of the year and for each of the four scenarios (except month 2 and 7 in scenario 4Hospital), the simulation using TAR time management executes between 3.5-23.9% faster than its NER counterpart (see table 1).
- the average performance gain by using TAR over NER for scenarios 1Hospital, 2Hospital, 3Hospital and 4Hospital is approximately 13.7%, 21%, 19% and 6% respectively.

Let us now consider the implications of these observations.

A. Comparing Standalone and Distributed Implementations

By applying the principles of distributed simulation and the HLA the time taken to run the NBS simulation was reduced significantly when the model became larger. When compared with the conventional NBS model, both the distributed versions recorded a negative performance improvement for scenarios 1Hospital and 2Hospital. In percentages, the NER version of the model was running slower by approximately 3498% and 653% for scenarios 1Hospital and 2Hospital respectively when compared with the standalone version. The TAR version faired slightly better by running 3004% and 495% slower for the same scenarios. As more complicated models were introduced in scenarios 3Hospital and 4Hospital the distributed models ran faster recording a performance increase of 28% and 54% for the NER variant and 41% and 58% for the TAR variant of the distributed NBS model respectively, when compared with the conventional standalone NBS model.

These findings indicate that for the conventional method an expansion in model size will be accompanied by a near exponential increase in the total runtime. On the other hand, for the distributed methods an increase in the number of hospitals (and therefore of computers) will be followed by a much smaller increase in total runtime. Therefore, if more than two hospitals are added to any model, the distributed method would be a better platform in which to develop and run the simulation experiments. Overall, the distinctive trend that the two methods follow concerning runtimes seems to be continuous; in other words the more hospitals we add to the model, the more the differences in the runtimes between the two methods favour the distributed approach. The increase in runtime appears to be primarily due to a large event list caused by a

combination of the volume of entities and the "counting down" of the shelf life of blood products in minutes. The large event list in turn causes swapping between RAM and virtual memory which further causes long runtimes. Our results suggest that the distributed approach allows the processing and memory demands made by large event lists to be shared over several computers. Note that eliminating the "counting down" model feature with a different approach to blood product shelf life would most likely give an increase in performance. However, this would invalidate the model.

It may be argued that a machine with more processing power and with more RAM (compared to our 1.73GHz processor and 1GB RAM laptops) could execute the standalone 3Hospital and 4Hospital scenarios of NBS model much faster, such that it outperforms its distributed 3Hospital and 4Hospital counterparts. Thus the negative performance improvement recorded by using the distributed models, as against using the conventional standalone models, for scenario 1Hospital and 2Hospital may also occur in scenarios 3Hospital and 4Hospital through use of better hardware. This would make the distributed simulation infeasible.

Although there is some merit to this argument we would like to raise two specific points to show the feasibility of using the distributed approach. Firstly, having more CPUs and more memory does always appear to equate to faster performance. This is especially true in case of machines having multiple CPUs (Dual-Core and Quad-Core processors) or machines that have CPUs with Hyper-Threading Technology (HTT) enabled. HTT is a new CPU technology and more elaboration is necessary for further discussion later in this section. HTT makes a single physical processor appear as two logical processors, wherein the physical execution resources are shared

and the architecture state is duplicated for the two logical processors [42]. The operating system treats a hyper-threaded CPU as two processors instead of one and a program can schedule processes or threads on both the logical processors and the CPU will execute them simultaneously, as if there were two physical processors present in the system.

One important factor that determines that a program executes faster on a higher configuration machine is that the program itself has been implemented to make the best possible use of all the available hardware in the system. Thus, it differs according to package implementation. To test whether Simul8 gains from an even higher configuration machine we tested the 4Hospital scenario on a PC having 2GB RAM and 3.2GHz Hyper-threaded Pentium 4 CPU. The time taken to run the simulation was around 38 hours. Thus, the execution time was not reduced by using more hardware. One interesting observation that we made was that the CPU utilization was about 50%. Monitoring the program execution using Windows Task Manager we made another observation. Most of the processing was done on one "logical" processor (the P4 CPU was hyper-threaded). Further discussions with Simul8 confirmed these observations. Most of the processing in Simul8 takes place on one main thread that makes use of one "logical" processor (in case HTT is enabled) or one "physical" processor (in case of Dual-Core and Quad-Core machines). Thus, our experience shows that in order for a CSP to utilize additional hardware effectively, the CSP vendor may have to modify the program itself. A distributed approach to CSP simulation may alleviate the need for such technology-specific changes.

Finally, it might be worth considering the effect of running the 4Hospital standalone NBS simulation on a CPU which is not HTT enabled. Would Simul8 have performed better if the

main processing thread had access to all the 3.2GHz processing power? It is generally the case that an application running on a non-HTT enabled CPU would execute faster than a HTT enabled CPU, but this also depends on whether the CPU can multi-task between different application programs. If Simul8 is programmed to utilize all the available CPU and memory (without relinquishing the control of the processor from time to time) then the package might indeed perform better. However, the current Simul8 implementation still means that it would not be able to utilize multiple physical processors. Multiple processors in a system are a reality that program developers have to face sooner than later for the following reason. Moore's law states that the number of transistors on a chip, or transistor density, doubles every 24 months. However, as transistor size decreases and transistor density and computing power increases, the heat generated by the chip becomes a major problem and multi-core processors become important [43]. Consequently, the major chip manufacturers are now looking at doubling CPU performance by increasing the number of CPU cores, as against doubling the clock-speed of a single CPU. Until the time a CSP is implemented to utilize multiple CPU-cores, distributed simulation of very large and complex models will remain feasible. Furthermore, we need to investigate the performance gains which can be expected by implementing multiple-processor friendly CSPs. Issues such as the division of the execution of a single instance of the simulation executive onto two processors, distributing the event list over multiple CPUs, etc. can be difficult and may require some synchronization of its own. As is the case with distributed simulation, to achieve this synchronization some overheads may be generated, Thus, whether standalone, multiple-processor CSP implementation outperforms distributed, single-processor CSP implementation, or vice-versa, is a question which requires further investigation.

Our second argument on the feasibility of distributed simulation for modeling large CSP-based supply chain models is that it can provide an alternative to single computer CSP simulation, in cases where the model to be simulated is so large and complex that its execution cannot be completed in acceptable time even on the fastest machine available for commercial purchase. In such cases, self-federating an existing CSP simulation by dividing the model between multiple computers can help reduce run time.

B. Comparing NER and TAR

The distributed simulation using TAR time management service call performs better because the discrete-event NBS simulation is modeled to exchange information at constant intervals of simulation time (the NBS PTI center and the hospitals exchange information at every 60 units of simulation time). Thus, it is possible to treat the NBS simulation as a time-stepped simulation in the distributed sense and use TAR to request RTI for a time advance equal to current logical time + 60 units of simulation time.

Using NER time management introduces the overhead of an extra *NextEventRequest* service call being made by a federate (and the resultant invocation of *TimeAdvanceGrant* callback by the RTI) whenever an interaction is received. Figs. 5 and 6 outline the protocols followed by NER and TAR versions of the CSP controller middleware (CCM) respectively.

The CCM-NER protocol represented in Fig. 5 shows that when a time-constrained federate (a federate that receives timestamped messages from other federates) and time-regulating federate

(a federate that sends timestamped messages to other federates) is in time granted state (see Fig. 9), the DMSO RTI Adapter of the CCM requests time advance (*timeRequested*) equal to either, (1) its *logicaltime + 60*, or (2) its previous time request (*timePreviouslyRequested*). (1) is used if the federate had received a *timeAdvanceGrant* equal to *timeRequested* during the preceding time advancing state. In short, if *timeGranted = timePreviouslyRequested* then *timeRequested* for the next NER call will be *logicaltime + 60*. This happens when no time stamped order (TSO) interactions are received by the federate during the time advancing stage. However, if an interaction is received then *timeGranted* by RTI will be equal to the timestamp of the interaction and *timeGranted* will be less than *timePreviouslyRequested*. As the simulation executes in equal timesteps, viz, 60, 120, 180, therefore *timeRequested* for the next NER call will be *timePreviouslyRequested* (but which was not granted by RTI). Since the *logicaltime* of the federate will be equal to *timeGranted* by RTI through the *timeAdvaceGrant* callback, we can also say that (a) if *logicaltime = timePreviouslyRequested* then *timeRequested* for the next NER call will be *logicaltime + 60*, and (b) *logicaltime < timePreviouslyRequested* then *timeRequested* for the next NER call will be *timePreviouslyRequested*.

As previously discussed, the CCM-TAR protocol represented in Fig. 6 is different because the DMSO RTI Adapter of the CCM always requests a time equal to its *logicaltime + 60* when invoking the next TAR request, irrespective of whether the federate has received an interaction in the preceding time advancing state. In this case the *timeGranted* returned by RTI through the *timeAdvaceGrant* callback will always be equal to *timePreviouslyRequested*. Any TSO interactions are delivered to the federate before the *timeAdvanceGrant* callback. Thus, using TAR time management mechanism in the NBS distributed simulation saves one redundant

message exchange between the federate and the RTI whenever the federate receives an interaction.

C. Analyzing Performance Gains Achieved by Using TAR over NER

To further examine the performance gain achieved by using TAR over NER and to investigate its gradual drop (from approx. 21% in scenario 2Hospital to approx. 6% in scenario 4Hospital) we focus on the interactions being sent across the NBS federation. As has been said earlier, the discrete-event NBS model can be perceived as a time-stepped simulation because the exchange of information between federates take place every 60 units of simulation time. The orders generated in the hospitals between two distinct time steps (say, 60 and 120) are kept in buffer and only released to the NBS PTI model in the subsequent time step (120 in this case). Similar is the case with NBS PTI model. The successfully match blood units are kept ready for delivery but not released to the hospitals until the next time step. In HLA-based simulation, a time-regulating federate in a time granted state can send interactions with any timestamp at least equal to its logical time + its *lookahead*. A lookahead value, expressed in terms of simulation time units, places a restriction on the time-regulating federate; if the federate is at a logical time t and has a lookahead value l, the RTI will not allow it to send timestamped messages with time less than $t+l$ [31]. The NBS models operate with a look ahead of 1 unit of simulation time. Thus, at time 120 the hospitals send interactions to NBS PTI with a time stamp of 121. These interactions carry order information specifying the requirement of blood. Similarly, the NBS PTI delivers interactions to the different hospitals at time 121 to inform the respective hospitals of the quantity of blood delivered along with a host of attributes.

The timestamp of the interactions received by a federate in time advancing stage are important. To find out why, let us extend our previous example and suppose that at logical time 120, hospital1, hospital2 and hospital3 send requests for blood. The timestamp of the interactions being sent to NBS PTI will be 121. The NBS PTI receives all the interactions in the time advancing stage when it requests the RTI to advance its simulation time to 180. The messages that the federate exchanges with RTI to reach logical time 180 will depend upon the time management service being used.

1. TAR: RTI delivers all three TSO interactions through *receiveInteraction* callback and then grants time 180 through *timeAdvanceGrant* callback. The logical time of the federate is therefore 180.
2. NER: RTI delivers the three TSO interactions to the NBS PTI federate using *receiveInteraction* callback. The RTI will then grant time 121 through *timeAdvanceGrant* and the federate will reach time granted state. The federate will then request time 180 from the RTI and in this occasion the time advance will be granted to 180. This is because communication between federates can only take place at constant intervals of time. At time 120, the set of orders were already released by the hospitals with a timestamp 121. If orders are generated between 120 and 180 they would be released when the hospitals are in the time granted state at logical time 180. The timestamp of the interaction for the next set of orders will be 181.

The above discussion shows that a NER federate in the NBS simulation generates a maximum of one extra pair of federate-RTI communication (when compared to a TAR federate) for every 60 units of simulation time, irrespective of the number of interactions it receives. In the example above, the NBS PTI federate received three interactions with timestamp 121 but generated only one extra NER call and received subsequent callback. The NBS simulation was run for 524160 simulated minutes. Therefore, the total number of extra federate-RTI communication that could be generated is 8736 (524160 / 60) for each NER federate. The actual number is much less since orders are not placed every hour by the hospitals and the NBS PTI delivers blood at pre-defined times (except for emergency cases).

From the discussions above it seems likely that the drop of average performance gain by using TAR over NER (from approx. 21% in case of scenario 2Hospital to approx. 19% in scenario 3Hospital and again to approx. 6% in scenario 4Hospital) cannot be attributed to an increased number of extra federate-RTI communications taking place as the number of hospitals are increased. As we have shown above, when the number of hospitals increase from 3 to 4, for example, the NBS PTI federate may receive a maximum of 4 interactions (one from each hospital placing an order). However, since the time stamps of the interactions received will be the same therefore the NER generates only one extra pair of federate-RTI communication in the form of one NER call and the subsequent callback received from RTI.

It seems likely that the drop in performance is because the NBS PTI model grows more complicated as it starts serving more hospitals. The process of finding a match between hospital orders and present blood stocks itself is complicated. As the number of hospitals increase this

process has to be repeated for orders for each hospital. The time gained by applying TAR time management mechanism is primarily because of the reduction of messages between federates. But as the NBS PTI model becomes more complex it takes longer to execute it and this slowly erodes the time gained through reduction of messages brought about through the application of TAR. A solution to this could be to divide the NBS PTI centre into two or more separate models. However, this would require revalidation of the model.

7. Conclusion and Future Work

This paper has described an investigation into using conventional and distributed approaches to simulating the supply chain of blood from the UK Southampton National Blood Service Centre to hospitals in this area. It has further applied two different conservative time management approaches to the distributed version of the model to investigate performance.

Using multiple sets of experiment results we have shown that a Simul8-DMSO RTI distributed simulation, will run faster than its standalone counterpart when the model has reached sufficient size. Thus, for our NBS model, distributed simulation appears to offer a viable alternative to conventional simulation by sharing the processing and memory requirements of the simulation across multiple computers. As we have used two specific software applications for our study (Simul8 and DMSO RTI 1.3NG), it is difficult for us to generalize our findings to encompass the entire range of CSPs and RTIs available today. However, it does add a success to the growing body of literature in this area.

We have also argued that the selection of an appropriate conservative time advance mechanism (NER or TAR) in HLA-based distributed simulation should be made not only based on the internal characteristics of the simulation, but consideration should also be given to the characteristics of the message flow between models. As has been shown in the case of NBS distributed simulation, a HLA federation comprising of discrete-event simulation federates (i.e., each federate simulates a discrete-event model), designed to exchange messages only at constant intervals of time, can be considered as a time-stepped simulation in the distributed sense. Thus, using TAR time management service call is more appropriate in this case as compared to using NER.

The HLA-CSP interoperability proposed by the authors can potentially be used with other "black box" simulation packages that expose package functionality through APIs, COM objects etc. However, it is not practically possible to investigate individual CSPs and to find out whether it offers appropriate function calls (for example, open simulation model, start and stop simulation, extract variable from queues, etc.) that would allow execution of a HLA-based simulation specific to a particular CSP. However, the authors have included a table (table 2) which lists example discrete-event simulation CSPs that expose package functionality, and thus have the potential to execute HLA-based simulations.

The limitation of this work is that the discussions are based solely on the NBS simulation models, which by design have very high event density and model-to-model synchronization takes place at constant intervals of time. This work complements research done by Boon Ping, et al. [32] that also shows the importance of model features on performance. Both these features were perfectly suited for implementing a distributed simulation solution with the NBS models. In

future, the authors would like to investigate the applicability of distributed simulation on models which may exhibit other defining features*!*

In future, we would also like to experiment with the full 16 hospital scenario. From our experiments with the different versions of the conventional model, we know that an expansion in model size will be accompanied by a severe increase in the total runtime. This is because the number of events in the event list increases as the scope of one conventional NBS simulation is augmented to model the behavior of more and more hospitals. As the event list continues to grow in size, the processing and memory constraints of a single computer means that it is no longer able to effectively simulate the model. Running a 16 hospital simulation over multiple computers (such as the NBS distributed simulation), where each computer will simulate a model incorporating more than one hospital (such as the NBS conventional simulation), will split the processing and memory requirement associated with executing the 16 hospital simulation. However, the number of computers to be used in order to effectively execute the full problem will need to take into consideration the following.

(a) The processing requirements of the NBS PTI Centre. As the NBS PTI will be responsible for all 16 hospitals, thus it can be expected that the event list will be quite large. It may be required that the processing of the NBS PTI centre is also distributed so that it can efficiently manage the demand and supply of blood units for all the hospitals.

(b) Since the hospitals in the full 16 hospital scenario can be small, medium or large, the event list will differ in size. Thus, the number of hospitals that can be effectively simulated using one

instance of Simul8 will differ according to the size of the hospital. Some load balancing between models may therefore be necessary (for example, we may decide to model two small hospitals and one large hospital in one instance of Simul8; three small hospitals and one medium hospital in another instance; and so forth).

In the immediate future, we are investigating the performance of the NBS distributed supply chain simulation with other RTIs to determine the sensitivity of our results to changes in RTI technology. Future work will involve simulating larger NBS supply chain models and end user requirements for making this technology easily used by OR/MS practitioners.

Acknowledgements

We would like to thank Dr. Mark Elder, founder and CEO of Simul8 Corporation, for granting us the use of multiple Simul8 professional licenses and for the insightful discussions. Thanks are also due to the Simul8 technical support team.

References

[1] K. Katsaliaki and S. Brailsford, “Using Simulation to Improve the Blood Supply Chain,” *Journal of the Operational Research Society* advance online publication, 19 April 2006, doi: 10.1057/palgrave.jors.2602195.

[2] T. W. Tewoldeberhan, A. Verbraeck, E. C. Valentin, and G. Bardonnet, “An Evaluation and Selection Methodology for Discrete-Event Simulation Software,” in *Proc. 2002 Winter Simulation Conference*, 2002, pp. 67-75.

[3] J. J. Swain, “Simulation Reloaded: Sixth biennial survey of discrete-event software tools,” *OR/MS Today*, vol. 30, no. 4, pp. 46-57, 2003.

[4] R. M. Fujimoto, “Distributed Simulation Systems,” in *Proc. 2003 Winter Simulation Conference*, 2003, pp. 124-134.

[5] S. J. E. Taylor, X. Wang, S. J. Turner, and M.Y.H. Low. “Integrating Heterogeneous Distributed COTS Discrete-Event Simulation Packages: An Emerging Standards-based Approach,” *IEEE Transactions on Systems, Man and Cybernetics: Part A*, vol. 36, no. 1, pp. 109-122, 2006.

[6] NHS Blood and Transplant. NHSBT website, 2006. [Online]: *http://www.nhsbt.nhs.uk*. Last accessed on December 20, 2006.

[7] R. M. Fujimoto, Parallel and Distributed Simulation Systems. New York, NY: John Wiley & Sons Inc., 1999.

[8] IEEE 1516, “IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA),” New York, NY: Institute of Electrical and Electronics Engineers, 2000.

[9] IEEE 1516.0, “IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Rules,” New York, NY: Institute of Electrical and Electronics Engineers, 2000.

[10] IEEE 1516.2, “IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Object Model Template (OMT) Specification,” New York, NY: Institute of Electrical and Electronics Engineers, 2000.

[11] IEEE 1516.1, “IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification,” New York, NY: Institute of Electrical and Electronics Engineers, 2000.

[12] IEEE 1516.3, “IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Development Process (FEDEP),” New York, NY: Institute of Electrical and Electronics Engineers, 2003.

[13] S. Strassburger, T. Schulze, U. Klein, and J. O. Henriksen, "Internet-based Simulation using off-the-shelf Simulation Tools and HLA," in *Proc. 1998 Winter Simulation Conference*, 1998, pp. 1669-1676.

[14] H. Hibino, Y. Fukuda, Y. Yura, K. Mitsuyuki, and K. Kaneda, "Manufacturing adapter of distributed simulation systems using HLA," in *Proc. 2002 Winter Simulation Conference*, 2002, pp. 1099-1107.

[15] K. Mertins, M. Rabe, and F.W. Jäkel, "Neutral Template Libraries for Efficient Distributed Simulation within a Manufacturing System Engineering Platform," in *Proc. 2000 Winter Simulation Conference*, 2000, pp. 1549-1557.

[16] K. Mertins and M. Rabe, "Inter-Enterprise Planning of Manufacturing Systems Applying Simulation with IPR Protection," in *Proc. 5th International Conference on Design of Information Systems for Manufacturing (DIISM),* 2002, pp.149-156.

[17] C. McLean and F. Riddick, "The IMS MISSION architecture for distributed manufacturing simulation," in *Proc. 2000 Winter Simulation Conference,* 2000, pp. 1539-1548.

[18] M. Rabe and F.W. Jäkel, "Non military use of HLA within distributed manufacturing scenarios," in *Proc. Simulation und Visualisierung*, 2001, pp. 141-150.

[19] M. Rabe and F.W. Jäkel, “On standardization requirements for distributed simulation,” in *Proc. Production and Logistics. Building the Knowledge Economy*, Twente, The Netherlands, 2003, pp. 399-406.

[20] S. Straßburger, “Distributed Simulation Based on the High Level Architecture in Civilian Application Domains,” Ghent, Belgium: Society for Computer Simulation International, 2001.

[21] B. P. Gan, P. Lendermann, M. Y. H. Low, S. J. Turner, X. Wang and S. J. E. Taylor, “Interoperating Autosched AP using the High Level Architecture,” in *Proc. 2005 Winter Simulation Conference*, 2005, pp. 394-401.

[22] R. Revetria, P. E. J. N. Blomjous, and S. P. A. Van Houten, “An HLA Federation for Evaluating Multi-Drop Strategies in Logistics,” in *Proc. 15th European Simulation Symposium and Exhibition Conference*, 2003, pp. 450-455.

[23] A. Borshchev, Y. Karpov, and V. Kharitonov, “Distributed simulation of hybrid systems with AnyLogic and HLA,” *Future Generation Computer Systems*, vol. 18, no. 6, pp. 829–839, 2002.

[24] S. J. E. Taylor, S. J. Turner, M. Y. H. Low, X. Wang, S. Strassburger, and J. Ladbrook, "Developing Interoperability Standards for Distributed Simulation and COTS Simulation Packages with the CSPI PDG," in *Proc. 2006 Winter Simulation Conference*, 2006, pp. 1101-1110.

[25] S. Strassburger, "The Road to COTS-Interoperability: From Generic HLA-Interfaces Towards Plug-and-Play Capabilities," in *Proc. 2006 Winter Simulation Conference*, 2006, pp. 1111-1118.

[26] X. Wang, S. J. Turner, M. Y. H. Low, and S. J. E. Taylor, "COTS Simulation Package (CSP) Interoperability – A Solution to Synchronous Entity Passing," in *Proc. 20th Workshop on Principles of Advanced and Distributed Simulation*, 2006, pp. 201-210.

[27] S. J. E. Taylor, L. Bohli, X. Wang, S. J. Turner, S.J, and J. Ladbrook, "Investigating Distributed Simulation at the Ford Motor Company," in *Proc. 9th International Symposium on Distributed Simulation and Real-Time Applications*, 2005, pp. 139-147.

[28] P. Lendermann, M. Y. H. Low, B. P. Gan, N. Julka, C. L. Peng, S. J. Turner, W. Cai, X. Wang, L. H. Lee, T. Hung, S. J. E. Taylor, and L. F. McGinnis, " (2005) An Integrated and Adaptive Decision-Support Framework for High-Tech Manufacturing and Service Networks," in Proc. *2005 Winter Simulation Conference*, 2005. pp. 2052-2062.

[29] N. Mustafee, S. J. E. Taylor, K. Katsaliaki, and S. Brailsford, "Distributed Simulation with COTS Simulation Packages: A Case Study in Health Care Supply Chain Simulation," in *Proc. 2006 Winter Simulation Conference*, 2006. pp. 1136-1142.

[30] N. Mustafee and S. J. E. Taylor, “Investigating Distributed Simulation with COTS Simulation Packages: Experiences with Simul8 and the HLA,” in *Proc. 2006 Operational Research Society Simulation Workshop (SW06)*, UK, 2006. pp. 33-42.

[31] F. Kuhl, R. Weatherly, and J. Dahmann, Creating Computer Simulation Systems: An Introduction to the High Level Architecture. Upper Saddle River, NJ: Prentice Hall PTR, 1999.

[32] B. P. Gan, M. Y. H. Low, X. Wang, and S. J. Turner, "Using manufacturing process flow for time synchronization in HLA-based simulation," In *Proc. 9th International Symposium on Distributed Simulation and Real-Time Applications*, 2005, pp. 148- 157.

[33] K.H. Concannon, K.I. Hunter, and J. Tremble, "SIMUL8-planner simulation-based planning and scheduling," in *Proc. 2003 Winter Simulation Conference*, 2003, 1488-1493.

[34] N.D. Gray, J. Hotchkiss, S. LaForge, A. Shalit, and T. Weinberg, "Modern languages and Microsoft's component object model," *Communications of the ACM*, vol. 41, no. 5, pp.55-65, 1998.

[35] Simul8 Corporation, “Simul8: Manual and Simulation Guide,” Glasgow, UK: Simul8 Corporation, 2003.

[36] US Department of Defense Modeling and Simulation Office, “High Level Architecture Run-Time Infrastructure RTI 1.3-Next Generation Programmer’s Guide,” US DoD M&S Office, 1999.

[37] Sun Microsystems Limited. Java Native Interface, 1995-2000. [Online]: *http://java.sun.com/j2se/1.3/docs/guide/jni/*. Last accessed on December 21, 2006.

[38] D. Alder. The Jacob Project: A Java–COM Bridge, Version 1.8, 1999-2004. [Online]: *http://danadler.com/jacob/*. Last accessed on December 21, 2006.

[39] Riley, G.F., Ammar, M.H., Fujimoto, R.M, Park, A., Perumalla, K. and Xu, D. (2004). A Federated Approach to Distributed Network Simulation, ACM Transactions on Modeling and Computer Simulation, 14, 2, 116-148.

[40] Lendermann, P., Julka, N., Gan, B.P., Chen, D., McGinnis, L.F. and McGinnis, J.P. (2003). Distributed Supply Chain Simulation as a Decision Support Tool for the Semiconductor Industry, SIMULATION, 79, 126-138.

[41] Cai, W., Turner, S.J., Lee, B-S and Zhou J. (2005). An Alternative Time Management Mechanism for Distributed Simulations, ACM Transactions on Modeling and Computer Simulation, 15, 2, 109-137.

[42] D. T. Marr, F. Binns, D. L. Hill, G. Hinton, D.A. Koufaty, J.A. Miller, and M. Upton, "Hyper-Threading Technology Architecture and Microarchitecture," *Intel Technology Journal*, vol. 6, no. 1, pp. 4-15, 2002.

[43] Q. Kelly, V. Turner, and J. Yang, "The Next Evolution in Enterprise Computing: The Convergence of Multicore x86 Processing and 64-Bit Operating Systems," IDC Whitepaper (#05C4442), 2005. [Online]:http://multicore.amd.com/GLOBAL/WhitePapers/IDC_WhitePaper_Convergence_en.pdf. Last accessed on December 19, 2006.

[44] Mustafee, N. (2007). A grid computing framework for commercial simulation packages. *PhD thesis*. School of Information Systems, Computing and Mathematics, Brunel University, UK.

List of Figures and Tables with captions

Table 1. Percentage Performance Increase of TAR over NER

	Scenario 1Hospital	Scenario 2Hospital	Scenario 3Hospital	Scenario 4Hospital
1 month	16.84	23.01	20.19	5.62
2 months	13.12	21.13	17.68	-7.89
3 months	12.83	21.88	19.28	6.46
4 months	15.19	22.48	18.97	11.73
5 months	14.33	20.92	19.81	8.56
6 months	13.12	19.40	17.07	6.82
7 months	11.76	20.40	18.98	-2.28
8 months	13.59	20.32	18.99	8.98
9 months	15.17	21.35	18.22	15.28
10 months	14.15	21.61	17.69	8.87
11 months	13.86	20.21	23.86	7.35
12 months	10.94	19.72	17.40	3.54
Mean	13.74	21.04	19.01	6.09

Table 2. CSPs that expose package functionality, adapted from [44]

Software	Vendor
AnyLogic	XJ Technologies
Arena	Rockwell Automation
eM-Plant	UGS
Enterprise Dynamics Studio	Incontrol Enterprise Dynamics
Extend Industry, Extend OR and Extend Suite	Imagine That, Inc.
Simcad Pro	CreateASoft, Inc.
Simprocess	CACI Products Company
Simul8 Standard and Professional Editions	Simul8 Corp
Witness	Lanner

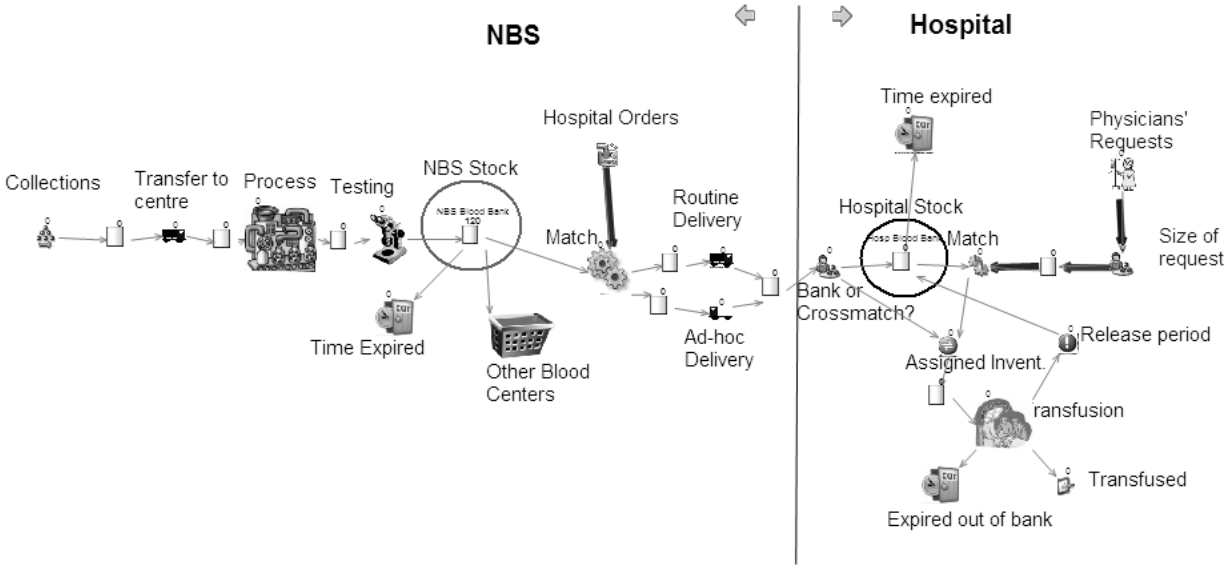


Fig 1. The National Blood Service Supply Chain Model (Simplified)

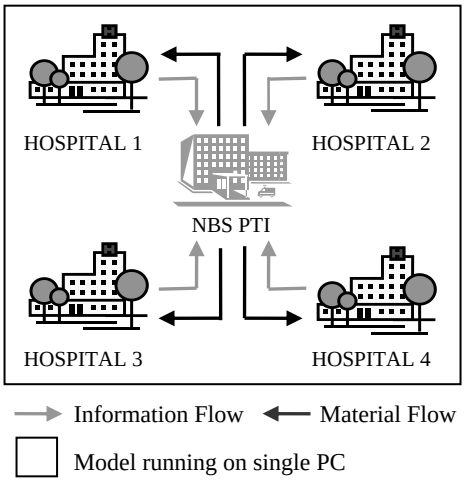


Fig 2. Conventional simulation approach with NBS PTI and four hospitals

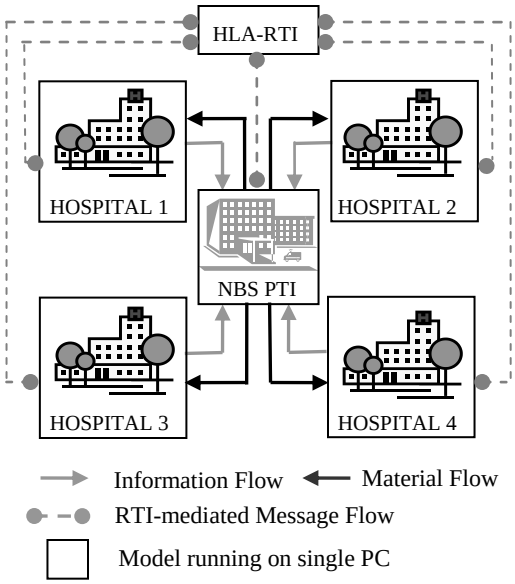


Fig 3. NBS Distributed Simulation with NBS PTI and Four Hospitals

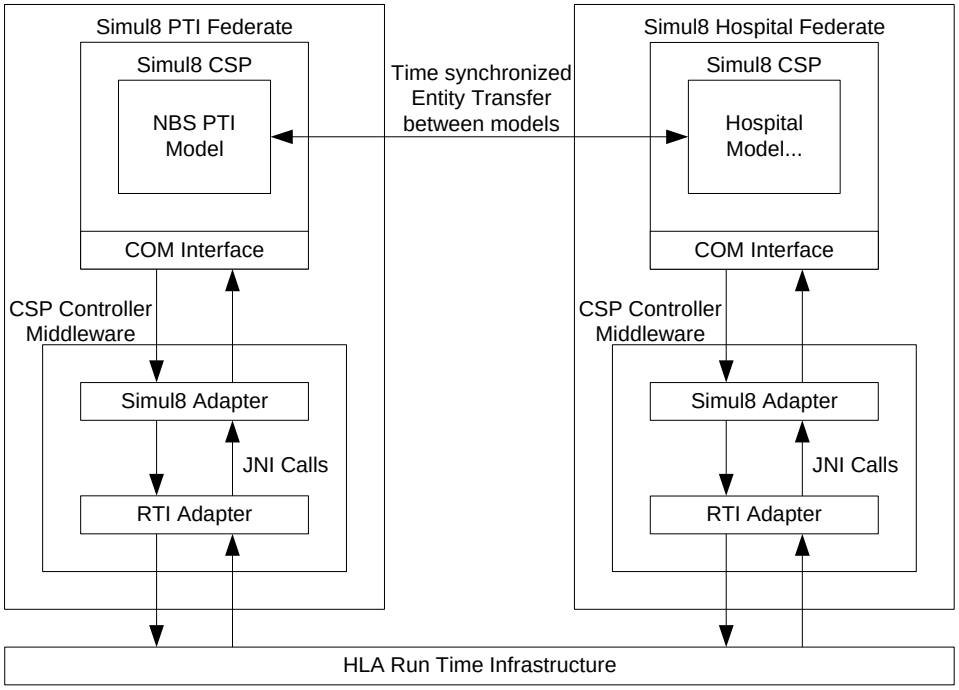
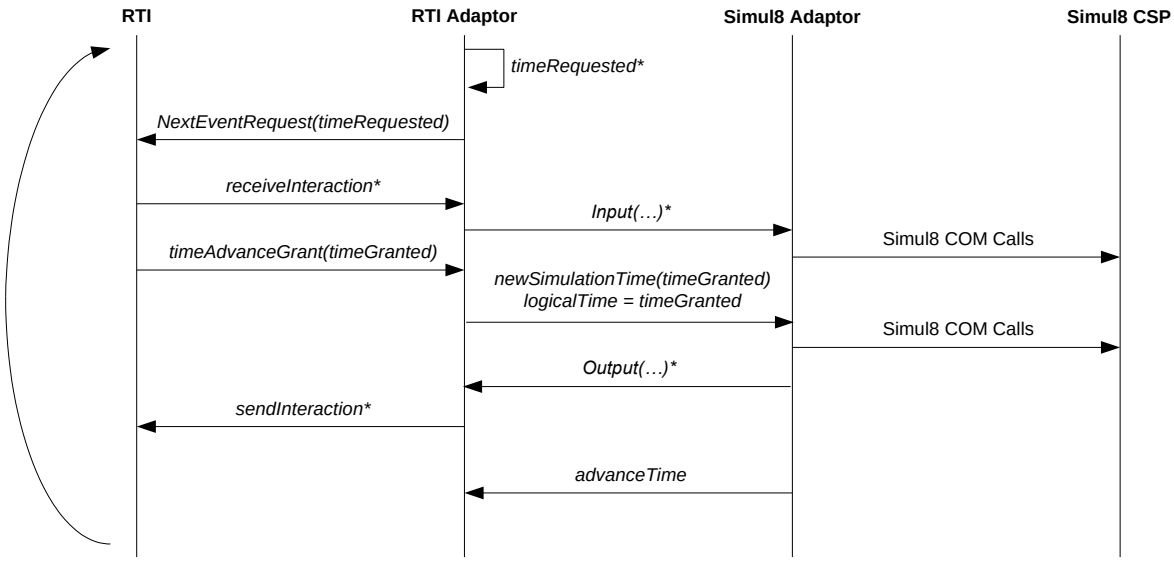


Fig 4. CSP Controller Middleware Architecture



NOTE: timeRequested = logicaltime+ 60 (if, *logicaltime = timePreviouslyRequested*) OR
timeRequested = timePreviouslyRequested (if, *logicaltime < timePreviouslyRequested*)

Fig 5. CCM-Next Event Request (NER) Protocol

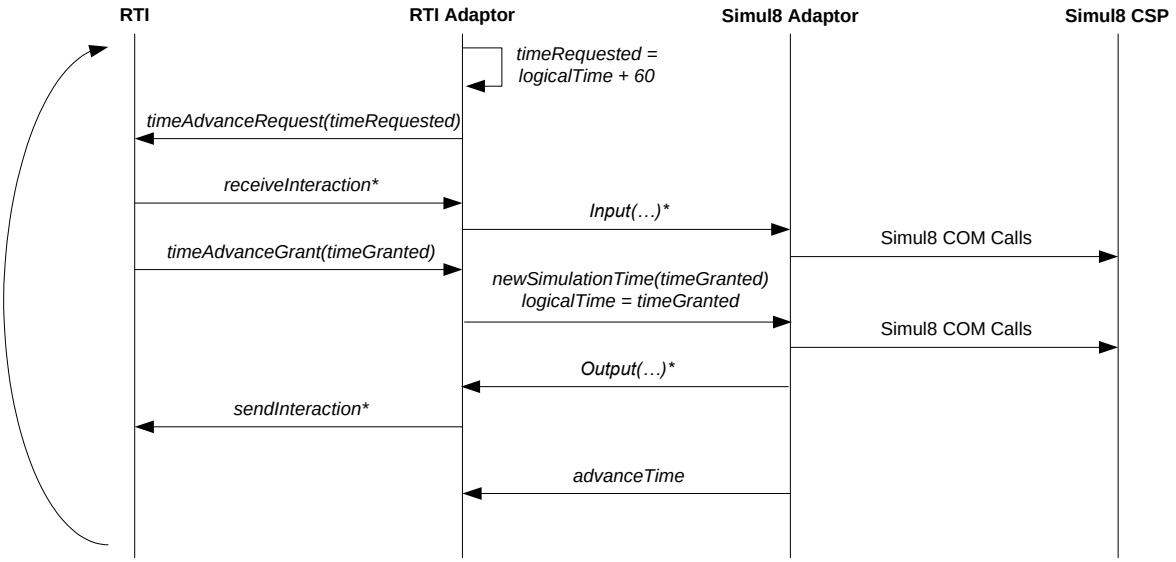


Fig 6. CCM-Time Advance Request (TAR) Protocol

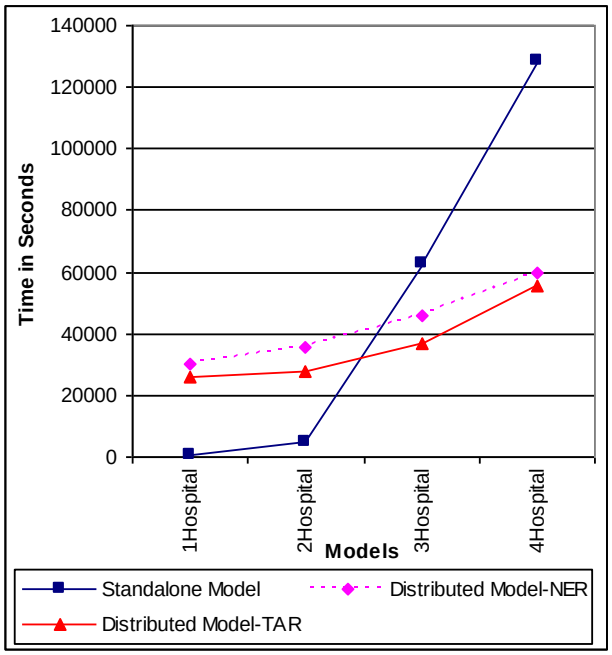


Fig 7: Execution time of NBS Distributed Simulation Using NER and TAR vs Standalone Simulation

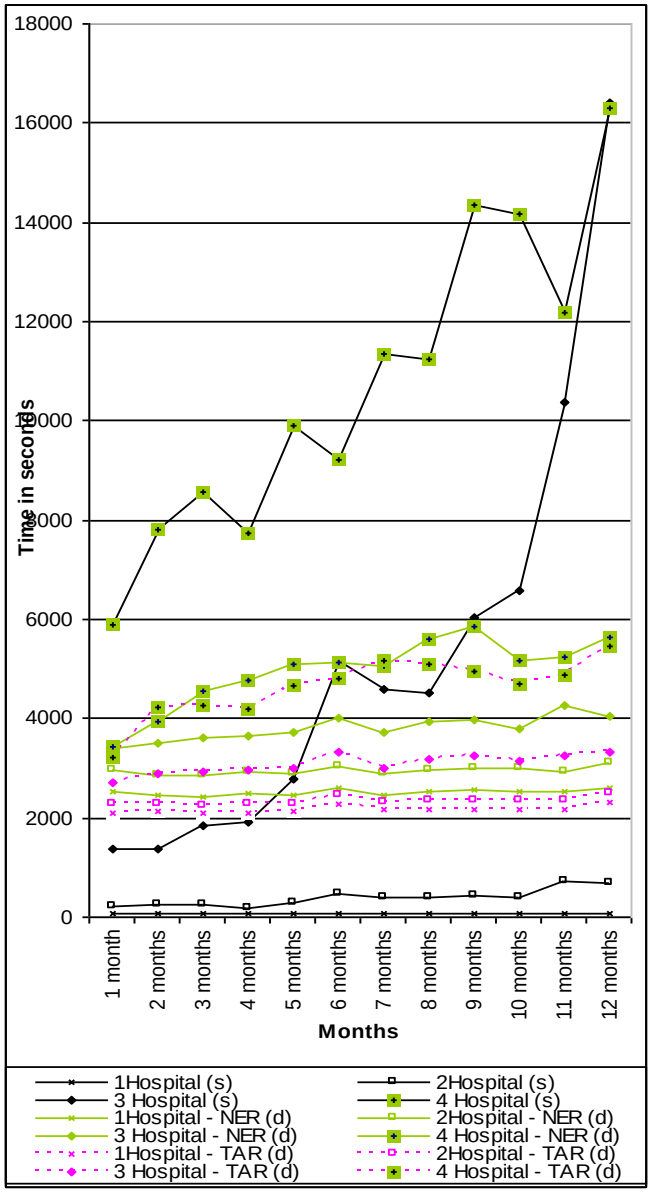


Fig 8. Monthly Execution Time of NBS Distributed Simulation Using TAR and NER Vs Standalone Simulation

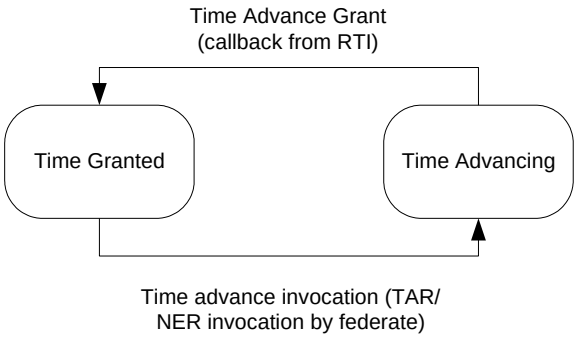


Fig 9. Time Management States of a Federate (Adapted from [31])