COTS Distributed Simulation: A Comparison of CMB and HLA Interoperability Approaches to Type I Interoperability Reference Model Problems

Simon J E Taylor (1), Stephen J Turner (2),
Navonil Mustafee (1), Henrik Ahlander (3) and Rassul Ayani (3)

(1) Centre for Applied Simulation Modelling
Department of Information Systems and Computing
Brunel University
Uxbridge, UK

(2) Parallel and Distributed Computing Centre
School of Computer Engineering
Nanyang Technological University
Singapore

(3) Laboratory for Electronics and Computer Systems (LECS)
Royal Institute of Technology (KTH)
Stockholm, Sweden

Abstract

Commercial-off-the-shelf (COTS) simulation packages (CSPs) are software used by many simulation modellers to build and experiment with models of various systems in domains such as manufacturing, health, logistics and commerce. COTS distributed simulation deals with the interoperation of CSPs and their models. Such interoperability has been classified into six interoperability reference models. As part of an on-going standardisation effort, this paper introduces the *COTS Simulation Package Emulator*, a proposed benchmark that can be used to investigate Type I interoperability problems in COTS distributed simulation. To demonstrate its use, two approaches to this form of interoperability are discussed, an implementation of the CMB conservative algorithm, an example of a so-called "light" approach, and an implementation of the HLA TAR algorithm, an example of a so-called "heavy" approach. Results from experimentation over four federation topologies are presented and it is shown the HLA approach out performs the CMB approach in almost all cases. The paper concludes that the CSPE benchmark is a valid basis from which the most efficient approach to Type I interoperability problems for COTS distributed simulation can be discovered.

1. Introduction

It is a well known fact that *commercial-off-the-shelf (COTS) simulation packages* (CSPs), such as Arena, Automod, EMPlant, Prodmodel, Simul8, Taylor and Witness, are software used by many simulation modellers to build and experiment with models of various systems in domains such as manufacturing, health, logistics and commerce. Swain (2001) reviews many of these. There is a growing body of research dedicated to the study of COTS distributed simulation, the interoperation of CSPs and their models (Taylor et al. 2003a). The High Level Architecture – COTS Simulation

Package Interoperation Forum (HLA-CSPIF) (www.cspif.com) was created in August 2002 in an attempt to unify this research. For convenience, the use of interoperation techniques to create a distributed simulation consisting of CSPs we will call COTS distributed simulation. One of the outputs of the Forum is the classification of some of the interoperability requirements of COTS distributed simulation on the basis of *interoperability reference models* (Taylor 2003). These are:

- Type I - Asynchronous Entity Passing
- Type II - Synchronous Entity Passing (The Bounded Buffer Problem)
- Type III - Shared Resources
- Type IV - Shared Events
- Type V - Shared Data Structure
- Type VI - Shared Conveyor

There have been various attempts to interoperate models and the COTS simulation packages in which they have been developed (Boer, et al. 2002; Gan and Turner 2000; Hibino, et al. 2002; Lendermann, et al. 2001; McLean and Riddick 2000; Sudra, et al. 2000; and Taylor, et al. 2002). Most of these approaches deal with interoperability problem of transferring an *entity*, or similar representation of temporary model state, between models and their CSPs. This problem is described by the Type I interoperability reference model *Asynchronous Entity Passing*, i.e. entities are passed between distributed models as timestamped event messages with no other synchronisation (the Type II interoperability reference model *Synchronous Entity Passing* describes the interoperability requirements of passing entities to models that receive them to a bounded buffer).

It is interesting to note that of the cited approaches to interoperability, roughly half use software (a runtime infrastructure (RTI)) compliant with the High Level Architecture (HLA) (IEEE 1516-2000 2000) and half do not. The motivations for its use (and non-use) appear to be finely balanced. For example, a major factor in using an RTI compliant with the HLA is that it is software based on a standard. The development of an interoperability approach and associated software based on a standard, at least in theory, infers widespread usability of that approach and its software. However, the non-HLA camp cites arguments such as cost and performance as significant factors against the use of HLA-complaint software. They argue that although it was once possible to obtain "free" versions of an RTI, today one cannot. Any interoperability solution based on the HLA will therefore add a significant cost factor to the solution in that a RTI must be purchased. This is possibly a false argument as any other approach to interoperability will ultimately *also* add cost as those involved attempt to recoup their investment. More convincing perhaps is the argument of performance. Over the past few years a perception has grown that the HLA is too "heavy", i.e. a HLA-compliant RTI is perceived to be geared to supporting the communication of huge volumes of information in support of large, real-time distributed simulations. It is argued that the communication needs of substantially smaller COTS distributed simulation is "light" and therefore does not need much of the RTI software. From this a view has appeared that immediately discounts HLA approaches as being too cumbersome for the needs of COTS distributed simulation.

As part of an on-going community discussion, this paper attempts to compare a "light" approach against a "heavy" approach to interoperation for COTS distributed simulation. Our "light" approach is an implementation of the well-known Chandy-Misra-Bryant (CMB) conservative time management protocol (Chandy and Misra 1978). Our "heavy" approach uses the DMSO RTI 1.3 NG version 5 and is based on the time advance request (TAR) (Fujimoto 2000). We make the comparison on the basis of the *CSP Emulator* (CSPE) described in the next section.

The rest of the paper is structured as follows. Section 3 and 4 discusses our "light" and "heavy" approaches respectively. Section 5 presents results taken from experimentation over four different interoperability scenarios using CSPE. Section 6 discusses our findings. Section 7 concludes the paper.

2. The COTS Simulation Package Emulator

The introduction to this work cited many good attempts to interoperate CSPs. A wide variety of techniques are used to interface between interoperability middleware and CSPs. A problem with this is that it is therefore difficult to make a comparison of efficient interoperability solutions as the latency of the CSP interface can mask an otherwise good approach (for example, some CSP interfaces only allow incremental time advance and prevent more effective approaches to be adopted (Taylor, et al. 2003b review four different approaches to this).

The CSP Emulator (CSPE) was created to provide a benchmark through which alternative approaches to COTS distributed simulation can be compared. It exhibits the computational characteristics of a CSP without a visual interface. It uses a three-phase simulation executive to perform the discrete event simulation of a simple model (Schriber and Brunner (2002) provide a discussion of this and other simulation world views). In the investigation of Type I interoperability reference model problems, CSPE performs the simulation of a pipeline model shown in figure 1. As can be seen, the model consists of set of FIFO queue-workstation pairs with an entry and exit point. This was chosen as the simplest model that allows various experimental factors relevant to the distributed simulation of Type I interoperability reference model problems to be controlled while reflecting a realistic simulation modelling environment (i.e. other, more complex Type I models could be created but with little benefit to the investigation). There are three experimental factors in CSPE. These are *lookahead*, *workload* (reflected by the ratio of arriving entities to the number of internal events that will be generated as a result of an arrival, i.e. the number of workstations) and *entities generated* (the number of entities generated by CSPE during experimentation). The simulation and real time taken for a workstation to process an entity, as well as the inter-arrival time of entities, can also be controlled. It is assumed in CSPE that queues are unbounded.

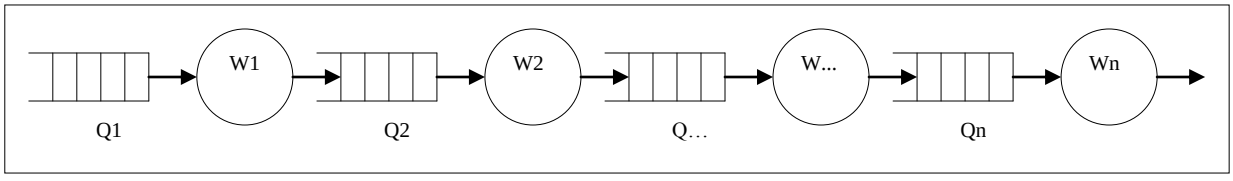


Figure 1: FIFO Pipeline Model used by CSPE

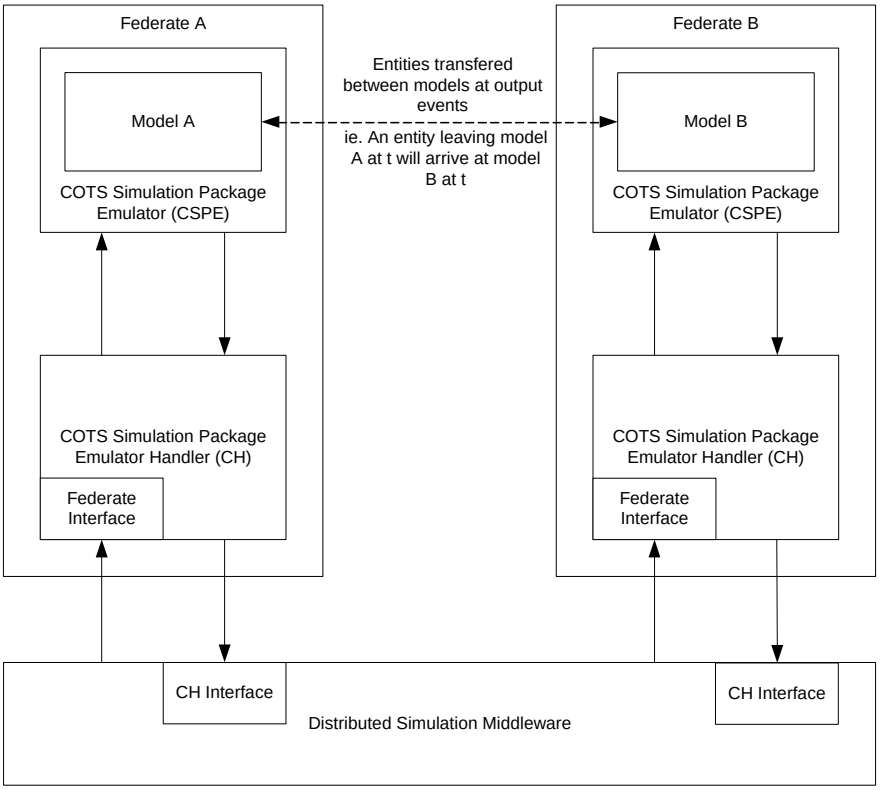


Figure 2: COTS Simulation Package Emulator Relationships (Type I)

The simulation of the model contained in CSPE is controlled via a simple API defined in the CSPE-Handler (CH). Through this API, CH can advance CSPE's simulation time and send and receive entities to and from CSPE. CH also interfaces CSPE to distributed simulation middleware via an interface determined by the form of the middleware. The interface between CH and CSPE is implemented through sockets. These relationships are shown in figure 2. The API is described as follows.

- *start()* is used to signal CSPE to start the simulation.
- *advance(time)* instructs CSPE to advance the simulation until *time*.
- *advance(time, entity)* instructs the CSPE to advance the simulation until *time* and introduce the entity *entity* into the model at the entry point.
- *output(time)* is sent to CSPE-Handler from CSPE for several reasons depending on the needs of the middleware. CSPE-Handler then passes this to the middleware as appropriate.
- *output(time, entity)* is sent to CSPE-Handler when an entity *entity* leaves the CSPE model exit point at *time* (effectively the time at which the last machine processes the entity). CSPE-Handler then passes this to the middleware as appropriate.
- *terminate()* This method is called by CSPE-Handler when it receives simulation completion notification from CSPE. What this method will do is specific to the middleware.

We now present our two approaches.

3. CSPE-CMB

To investigate a so-called "light" approach we implemented the CMB algorithm and linked it to the CH to form CSPE-CMB. In our implementation, null messages are sent in the following two cases.

- After execution of every event and time advance caused by a null message.
- Whenever an incoming link from a federate is empty. In this case the CSPE-CMB middleware sends null messages to other federates in order to resolve possible deadlock.

To satisfy the known topology needs of CMB, the links between each federate (LP), each federate must therefore know beforehand the other federates with which it will interact during a simulation run. CSPE-CMB meets this condition by reading federate topology information from a *Federate Definition File*. Link queues are set up in the CH. The distributed simulation middleware in this case is just TCP/IP, with messages passed via sockets and IP addresses connecting the CHs.

Under CMB, CSPE passes two kinds of messages to the CH. These are null messages, with timestamp equal to the time of that CSPE has just advanced to (CH currently adds lookahead – derived from the minimum timestamp increment of a federate) and event messages that are sent to other federates. In terms of CSPE interface messages, null messages are represented by *output(time)* and event messages are represented by *output(time, entity)*.

In order to guarantee that messages are sent in increasing timestamp order CSPE-CMB implements a buffer for event messages in CH. Thus, event messages are not sent immediately to other federates but are held in the buffer. Null messages are sent immediately. When the timestamp of a null message (the current simulation time + lookahead) equals or exceeds the timestamp of an event message in the buffer, only then is the external message sent. If there is more than a single event message that meets this condition, then all of them are sent before sending the null message. If the "equals" condition is met, the null message is not sent.

All messages received by a CH are placed in appropriate link queues. When the CMB algorithm identifies that the next message to be processed is a null message, *advance(time)* is used to order CSPE to advance to that time (processing internal events as appropriate). If the next message to be processed is an event message, then *advance(time, entity)* is used to order CSPE to advance to that time and to introduce the new entity (again processing internal event messages as appropriate).

4. CSPE-HLA

In the investigation of our "heavy" HLA-based approach, we developed a new variant of CSPE called CSPE-HLA. To represent the transfer of entities from one CSPE federate to another, CSPE-HLA uses *interactions*. Our justification of this is that it has been shown by experimentation that for the RTI-1.3 NG version 5 interactions have less latency than the other communication options. We base our implementation on the Entity Transfer Specification, version 1.1.1 that has been developed by the HLA-CSPIF to standardize the transfer of entities for the Type I interoperability

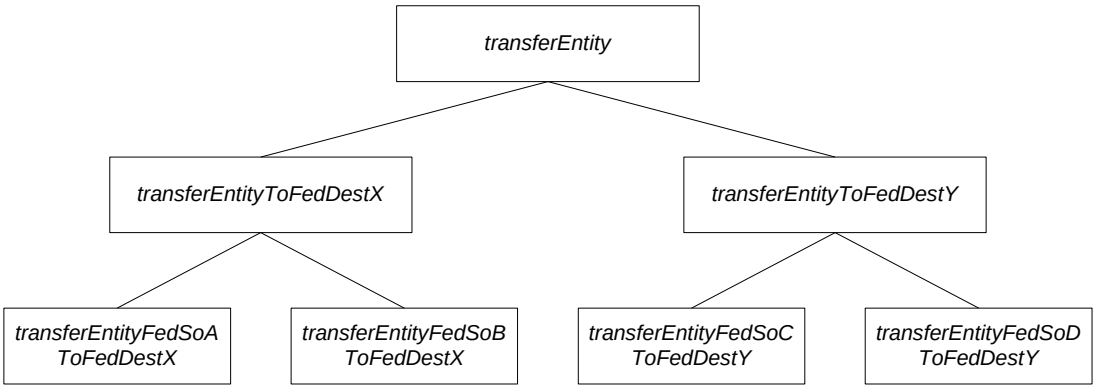


Figure 3: Interaction Class Hierarchy

reference model (Taylor, Turner and Low 2004). Figure 3 shows the interaction class hierarchy. For example, for a CSPE-HLA named *fedA* to interact with another named *fedB*, the interaction *transferEntityfedATofedB* would be used. *fedB* would subscribe to all interactions with itself by subribing to *transferEntityTofedB* and *fedA* would publish the interaction class *transferEntityfedATofedB* to send entities to *fedB.* As with CSPE-CMB, CSPE-HLA uses the *Federate Definition File* to specify what other federates a federate is connected to (as well as the lookahead. The interaction classes are derived from this file. To use an interaction class, the CH of CSPE-HLA calls *getInteractionClassHandle(name)*, where *name* is the name of the interaction class, to receive a handle to the interaction class. CSPE-HLA stores all handles to classes it publishes in a hash table with the class names as keys for fast access since they are needed every time an entity is sent.

To publish and subscribe to an interaction class a federate uses the methods *publishInteractionClass(handle)* and *subscribeInteractionClass(handle)*. To send and receive objects the methods *sendInteraction(handle, parameters, time, tag)* and *receiveInteraction(handle, parameters, time, tag, eventRetractionHandle)* used. *handle* is the interaction class's handle, *parameters* are the parameters of the interaction class (in this case only the dummy *message* is used), time is the timestamp of the object, *tag* is used for user-specified messages (not used in CSPE-HLA) and *eventRetractionHandle* is a unique identity for each event message in the federation (used in optimistic simulations for the retraction of objects, not used in CSPE-HLA).

The HLA has two main options available for conservative time advancement in a distributed simulation composed of CSPEs. These use either *timeAdvanceRequest()* (TAR) or *nextEventRequest()* (NER). As each approach gave similar performance results, we limit our discussion of the implementation of CSPE-HLA to TAR. This promises that a federate calling this method will not generate any timestamped events with a timestamp lower than the requested time + lookahead. In our approach, the CH first uses *queryMinNextEventTime()* to request the minimum next safe event time from the RTI to allow its CSPE to advance to. When this call returns the next safe event time *safetime*, the CH orders CSPE to advance until *safetime*. CSPE does this, executing internal events as it does so. This continues until the next event to be processed by CSPE is greater than *safetime*. If the next event is not equal to *safetime*, CSPE will advance to *safetime* as it is safe to do so. For each internal event that is processed, CSPE outputs *output(time)* and/or *output(time, entity)*. When *time* equals *safetime*, CH knows that CSPE has advanced as far as it can. When this occurs, CH

uses *timeAdvanceRequest(time)* to inform the RTI that its CSPE has reached the correct safe time as determined by *queryMinNextEventTime()*. CH then uses *receiveInteraction(entity,time)* to receive any new event messages sent from other federates. These are buffered until *timeAdvanceGrant()* is asserted (i.e. all safe messages have been delivered). CH introduces these to CSPE with *advance(time,entity)*. In the case of a single entity, when CSPE receives this, it will process the entity at *time*, i.e. it will treat this as a bound event and schedule new events and test conditional events as demanded by the B and C phases of the TPA (this occurs repeatedly for multiple entities). The time advancement cycle continues by calling *queryMinNextEventTime()* once again. Finally, if *output(time, entity)* is received by CH, it is converted into the appropriate interaction and passed to the RTI with *sendInteraction(entity,time).*

5. Experimentation

In this section we present our experimentation and results performed with the two variants of CSPE. Four different federation topologies were used (pipeline, local feedback, fully interconnected and producer-consumer) with three different experiments (variable external/internal event ratio, variable workload and variable lookahead). The federate topologies were chosen to reflect possible actual COTS distributed simulation. It has been observed that actual or proposed distributed simulations of industrial problem tend to have more than just simple connections, i.e. entities can be passed between federates in a fixed but arbitrary relationship. However, the first of our topologies, the *pipeline* (figure 4) is derived from the fact that the most simple (theoretical) manufacturing model can be a simple series of work processes. Entities are generated in source federate A and then passed in one direction through all federates until they finally are removed after been processed in federate F (sink). To investigate the effect of a more closely coupled relationship, our second topology *local feedback* (figure 5) reflects the class of models where entities represent, for example, rejected parts or confirmations of delivery. In our work, we assume that the entity represents a *batch*. In the case where there is more than one output, as with federate B, the output is chosen in a round robin manner. If the output follows the "backbone" of the pipeline, the entity is passed on as normal. If, however, a feedback loop is selected, we assume that part of the batch that the entity represents is faulty. The entity therefore splits into two – one entity carries on along the pipeline and the other is returned. The number of entities generated by the source federate A is adjusted to keep the number of entities generated constant. This allows us to analyse of the effect of extra coupling in the model (rather that the fairly obvious result of spiralling workloads).

Our third topology, *fully interconnected* (figure 6) reflects the case where there is local feedback and all federates can produce and consume entities. Entities are generated and passed around in a round robin manner. In all cases apart from where local feedback occurs, a received entity is consumed. If local feedback occurs alternatively, for example entities sent from federate B to federate E are alternatively consumed or split and returned after processing in federate E's model. This topology is included to represent a distribution network that is typical of some supply chains. Finally, the fourth topology model reflects a real-world problem where several producer models feed parts into a single consumer model (Taylor et al. 2002). This is termed *producer-consumer* topology (figure 7).

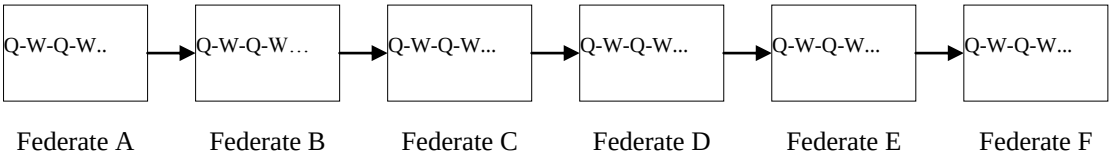


Figure 4: Pipeline Topology

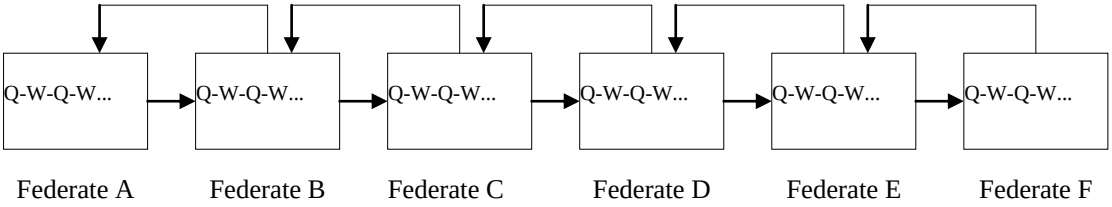


Figure 5: Local Feedback Topology

For each topology, three experiments were carried out: variable external/internal event ratio, variable workload and variable lookahead. For the first of these, *variable external/internal event ratio*, the ratio of external and internal events can be important as represents the volume of events that can be processed relative to the volume of event messages present in the distributed simulation. This is implemented by varying the number of machines in each copy of CSPE. Tables 1-3 show the experiments carried out. Note that the lookahead is equal to the setup and processing time for a workstation, as it the number of entities processed by each federate. For *variable workload*, the event ratio and lookahead are fixed so allowing us to investigate the scalability of our approaches as the amount of entities to be processed increases. Our final experiment allows us to investigate the effect of increasing lookahead on our approaches.

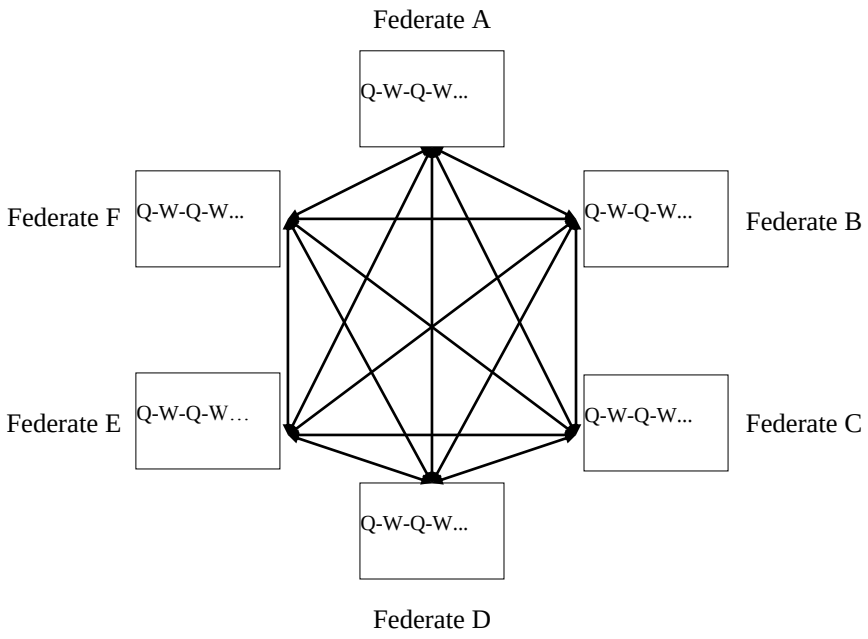


Figure 6: Fully Interconnected Topology

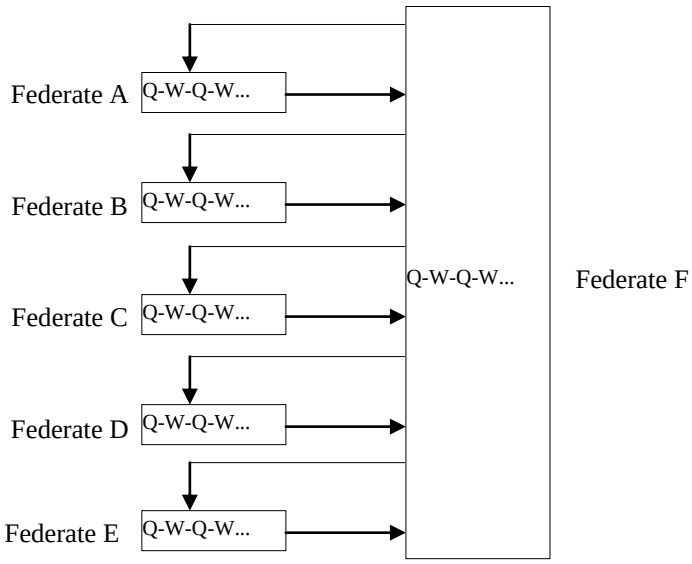


Figure 7: Producer-consumer topology

5.1 Results

Our performance tests were carried out on six computers connected through an isolated 10 Mbit local area network. Six computers ran a single CSPE federate. In the case of CSPE-HLA, a seventh computer was used to run the RTI Executive (RTI 1.3-NG version 5). Each of the six federate computers was an Intel Pentium III 650 MHz with 256 MB RAM running either Windows 2000 or Windows XP. The RTI executive computer ran at 950 MHz. An automatic test harness was developed to run the experiments. Each test was run three times with the result being taken as an average (no significant variance due to the isolated local area network). Figures 8-11 show the results for the pipeline, local feedback, fully interconnected and producer-consumer models respectively.

As can be seen, in variance of the *external/internal event ratio*, with a fixed workload of 1000 entities and lookahead of 10, in all cases apart from the pipeline model, for the CMB approach, execution time decreases slightly as the external event density decreases there is little effect on the magnitude of federation execution time. The opposite is true for the HLA approach, as the external event density decreases, execution time slightly increases. The observation here is therefore as the effective granularity of internal event processing increases CMB performs slightly better and HLA performs slightly worse. However, the most obvious result from these experiments is that HLA approach is far faster than the CMB approach. At an external event density of 0.05 the HLA performs better by a factor of 5.58 in the local feedback model, 6.53 in the fully connected model and 4.78 in the producer consumer model. In the pipeline model as external event density decreases, the effect on both CMB and HLA is more significant as both perform significantly worse. However, the most interesting result is that in this case CMB out performs HLA (at an event density of 0.05 by a factor of 1.27).

For variable workload, with a fixed event ratio of 0.2 and a lookahead of 10, an increasing volume of entities passed through the simulation, effects both the CMB and HLA approaches in a similar manner. For all models, as the volume of entities increases both the CMB and HLA approaches take more time to complete their work.

In all cases apart from the pipeline the HLA approach out performs the CMB approach. In processing 1000 entities the HLA approach performs better by a factor of 9.18 in the local feedback model, 12.30 in the fully connected model and 7.58 in the producer consumer model. Again, in the pipeline model the relationship is reversed with HLA performs marginally worse than the CMB approach by a factor of 1.63.

In terms of variable lookahead, with a fixed event ratio of 0.2 and workload of 1000 entities, as lookahead increases the effect on the CMB approach is to reduce the overall execution time in all models apart from the pipeline. In this case, the effect of increasing lookahead is negligible. The effect on the HLA approach is reversed; in all models apart from the pipeline the effect of increasing lookahead is negligible. In the pipeline model the HLA approach is significantly affected by increasing lookahead as execution time decreases with larger values of lookahead. Overall HLA out performs CMB in all models apart from the pipeline. In the pipeline model this relationship is reversed with CMB out performing HLA. In our results the smallest value of lookahead is 2 and the largest is 10 (the maximum possible value of lookahead - equal to the length of simulation time taken for a workstation to complete its task). At lookahead value 2, the HLA out performs CMB by a factor of a factor of 17.45 in the local feedback model, 22.20 in the fully connected model and 14.29 in the producer consumer model. In the pipeline model, CMB out performs HLA by a more marginal factor of 2.84. At the maximum value of lookahead, the HLA out performs CMB by a factor of a factor of 9.67 in the local feedback model, 12.07 in the fully connected model and 7.75 in the producer consumer model. In the pipeline model, CMB out performs HLA again by a marginal factor of 1.63.

The above observations on our results can be summarised as follows.

- In all models apart from the pipeline model and in all experiments the HLA approach performs better than the CMB approach,

- In the pipeline model in all experiments the CMB approach performs marginally better than the HLA approach.

6. Discussion

Our results indicated that in almost all cases our heavy approach out performed the light approach. Let us consider why this is the case. In the CMB approach, a federate must stop execution until it has established a safe condition under which it may advance time. This means it cannot advance until messages are present in all input link queues. To prevent possible deadlock, the CMB approach uses timestamped null messages to allow federates which cannot process any event messages to inform other federates safe times to advance to. Generally, for a federate to advance time in the CMB approach, both null messages and event messages must be consumed. In the HLA approach time progression is essentially a cycle of requesting the minimum next safe event time from the RTI software, advancing to that time and then checking for new event messages. The calculation of the new safe times is depending on all federates performing the request. The actual calculation is based on what interactions have been sent between federates and the lookahead. In general we can therefore observe that time progression in the CMB approach can be limited by the availability

of information in each link queue, and in the HLA approach by the time taken for all federates to request from the RTI the next safe time to which to advance.

From this, our results can be explained as follows. In the case of the pipeline where the CMB approach performs marginally better than the HLA, the progression of the CMB federates is almost always via the processing of event messages as they appear on the input links. Null messages occasionally appear as a result of delayed processing (especially at the beginning of the run) but are insignificant in numbers. There is therefore very little wasted processing. In our HLA approach, as the next safe time cannot be calculated until all federates have requested this, and all messages have been delivered, there is a comparative delay between request and action. It is this delay which causes this HLA approach to perform worse than the CMB approach. In all other cases, the presence of a feedback loop means that the CMB federates cannot progress until null messages have been propagated across their input links. These must then be processed. In the HLA approach, the federates are just required to follow a simpler cycle to request permission to advance time. It is this simpler time advancement cycle that allows the HLA experiments to perform significantly better than the CMB approach.

7. Conclusions

As part of an on-going community discussion, this paper has attempted to compare a "light" CMB-based approach against a "heavy" HLA-based approach to interoperation for COTS distributed simulation. The COTS Simulation Package Emulator has been introduced as a benchmark for the comparison of different approaches. Experiments over four topologies have been presented and discussed, and it has been shown that for almost all cases the HLA-based approach out performs the CMB approach.

Rather than the unrealistic conclusion that the "heavy" HLA or "light" CMB approaches is best, the contribution of this paper is, from the perspective of COTS distributed simulation, is the foundation for the search for the best interoperability solution. The COTS Simulation Package Emulator is the first benchmark that has made possible an informed discussion between interoperability approaches. Our results have indeed shown that for all experiments except the pipeline, the HLA approach out performs the CMB. However, it is important to note that it is entirely possible to improve on our so-called approaches. For example, Fujimoto (2003) notes that both conservative and HLA RTI approaches have several different forms that might lead to better performance for CSP distributed simulation. For example, the CMB approach can be made more efficient through revisiting the exploitation of "distance" between federates (Ayani 1989; Cai and Turner 1990) and lookahead (Meyer and Bagrodia 1999). The HLA approach as presented is essentially a modification of the TAR-based conservative time-stepped behaviour. The alternative next event request (NER) and flush queue request (FQR) are the basis of conservative event-driven and optimistic protocols and could possibly form the basis of better performance. Indeed, other approaches such as the FAMAS backbone (Boer, et al 2002) may also yield better results. However, without the existence of the CSPE benchmark it would be difficult to make an informed comparison as to which approach is best.

In conclusion, we hope that the CSPE-based work presented here will eventually lead to consensus on the "best" performing approach to the Type I interoperability reference model problem. Forms of CSPE for optimistic protocols and the Type II interoperability reference model are currently under development.

References

Ayani, R. (1989). A Parallel Simulation Scheme Based on the Distance Between Objects. *Proceedings of the SCS Multiconference on Distributed Simulation*, Society for Computer Simulation. 21: 113-118.

Boer, C. A., A. Verbraeck, and H.P.M. Veeke. (2002) The Possible Role of a Backbone Architecture in Real-Time Control and Emulation. In *Proceedings of the 2002 Winter Simulation Conference*, E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes (eds.) Association for Computing Machinery Press, New York, NY. 1675-1682.

Cai, W. and S. J. Turner (1990). An Algorithm for Distributed Discrete-Event Simulation -- the "Carrier Null Message" Approach. *Proceedings of the SCS Multiconference on Distributed Simulation*, SCS Simulation Series. 22: 3-8.

Chandy, K. M. and J. Misra. (1978) "Distributed Simulation: A Case Study in Design and Verification of Distributed Programs." *IEEE Transactions on Software Engineering* SE-5(5): 440-452.

Fujimoto, R. M. (2000) *Parallel and Distributed Simulation Systems*, Wiley Interscience.

Gan, B.P., and S.J. Turner. (2000) An asynchronous protocol for virtual factory simulation on shared memory multiprocessor systems. *Journal of Operational Research Society,* 51: 413-422.

Hibino, H, Y. Fukuda, Y. Yura, K. Mitsuyuki and K. Kaneda. (2002) Manufacturing Adapter of Distributed Simulation Systems Using HLA. In *Proceedings of the 2002 Winter Simulation Conference*, E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes (eds.) Association for Computing Machinery Press, New York, NY. 1099-1107.

IEEE Std 1516-2000 (2000) *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)*, Institute of Electrical and Electronics Engineers, New York, NY.

Lendermann, P., B.P. Gan and L.F. McGinnis. (2001) Distributed Simulation with Incorporated APS Procedures for High-Fidelity Supply Chain Optimization. In *Proceedings of the 2001 Winter Simulation Conference,* B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, (eds.) Association for Computing Machinery Press, New York, NY. 1138-1145.

Meyer, R. A. and R. L. Bagrodia (1999). Path Lookahead: A Data Flow View of PDES Models. *Proceedings of the 13th Workshop on Parallel and Distributed Simulation*. IEEE Press 12-19.

McLean, C. and F. Riddick. (2000) The IMS MISSION Architecture for Distributed Manufacturing Simulation. In *Proceedings of the 2000 Winter Simulation Conference*. J.A. Joines, R.R. Barton, K. Kang, and P.A. Fishwick (eds.). Association for Computing Machinery Press, New York, NY.1539-1548

Schriber, T.J. and D.T. Brunner. 2002. Inside discrete-event simulation software: How it works and why it matters. In *Proceedings of the 2002 Winter Simulation Conference,* E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes (eds.) Association for Computing Machinery Press, New York, NY. 97-107.

Sudra, R., S.J.E. Taylor, and T. Janahan. (2000) Distributed Supply Chain Simulation in GRIDS'. In *Proceedings of the 2000 Winter Simulation Conference*. J.A. Joines, R.R. Barton, K. Kang, and P.A. Fishwick (eds.). Association for Computing Machinery Press, New York, NY. 356-361.

Swain, J. J. (2001) Power tools for visualization and decision making: 2001 simulation software survey. *OR/MS Today* 28(1): 52-63.

Taylor, S.J.E. (2003) HLA-CSPIF: The High Level Architecture – COTS Simulation Package Interoperation Forum. In *Proceedings of the Fall 2003 Simulation Interoperability Workshop*. Simulation Interoperability Standards Organisation, Institute for Simulation and Training. Florida. 03F-SIW-126.

Taylor, S.J.E., R. Sudra, T. Janahan, G. Tan and J. Ladbrook. (2002) GRIDS-SCS: An Infrastructure for Distributed Supply Chain Simulation. *SIMULATION*. 78(5): 312-320.

Taylor, S.J.E., B.P. Gan, S. Strassburger, A. Verbraeck. (2003a) HLA-CSPIF Technical Panel on Distributed Simulation. In *Proceedings of the 2003 Winter Simulation Conference*. Association for Computing Machinery Press, New York, NY. 881-887.

Taylor, S.J.E., J. Ladbrook and J. Sharpe. (2003b) Time Management Issues in COTS Distributed Simulation: A Case Study. In *Proceedings of the 2003 Winter Simulation Conference*. Association for Computing Machinery Press, New York, NY. 838-846.

Taylor, S.J.E., S.J. Turner and M. Y.-H. Low. (2004) A Proposal for an Entity Transfer Specification for COTS Simulation Package Interoperation. In *Proceedings of the European 2004 Simulation Interoperability Workshop*. Simulation Interoperability Standards Organisation, Institute for Simulation and Training. Florida. 03E-SIW-126.

Table 1: Variable External/Internal Ratio

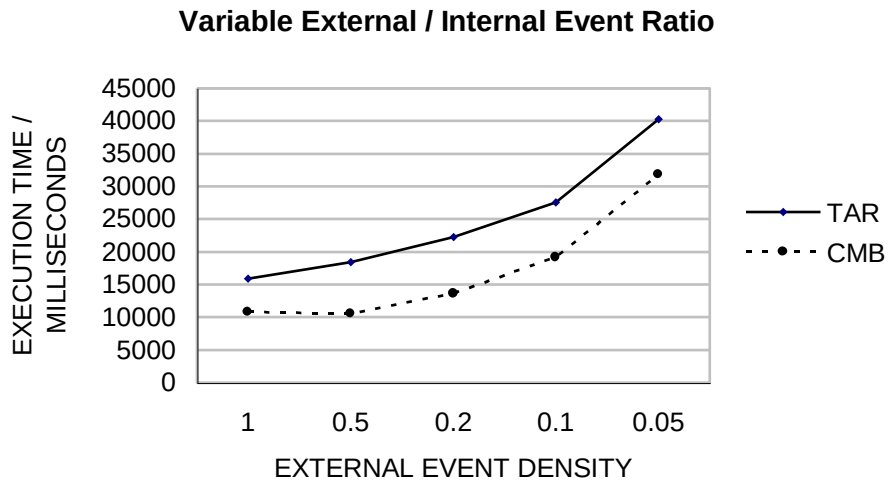
Experiment	Entities	Machines	External events	Internal events	External/ internal ratio	Machine Setup time	Machine Processing time	Lookahead
1	1000	1	1	1	1	5	5	10
2	1000	2	1	2	0.5	5	5	10
3	1000	5	1	5	0.2	5	5	10
4	1000	10	1	10	0.1	5	5	10
5	1000	20	1	20	0.05	5	5	10

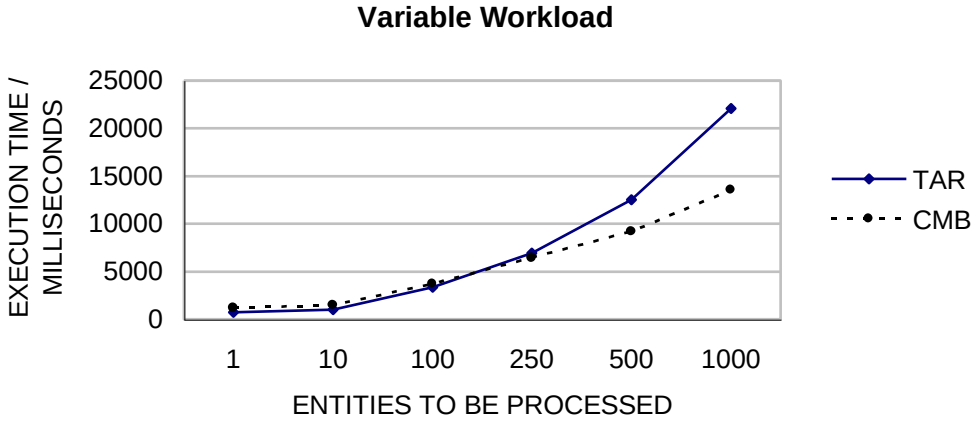
Table 2: Variable Workload

Experiment	Entities	Machines	External events	Internal events	External/ internal ratio	Machine Setup time	Machine Processing time	Lookahead
1	1	5	1	5	0.2	5	5	10
2	10	5	1	5	0.2	5	5	10
3	100	5	1	5	0.2	5	5	10
4	250	5	1	5	0.2	5	5	10
5	500	5	1	5	0.2	5	5	10
6	1000	5	1	5	0.2	5	5	10

Table 3: Variable Lookahead

Experiment	Entities	Machines	External events	Internal events	External/ internal ratio	Machine Setup time	Machine Processing time	Lookahead
1	1000	5	1	5	0.2	5	5	2
2	1000	5	1	5	0.2	5	5	4
3	1000	5	1	5	0.2	5	5	6
4	1000	5	1	5	0.2	5	5	8
5	1000	5	1	5	0.2	5	5	10





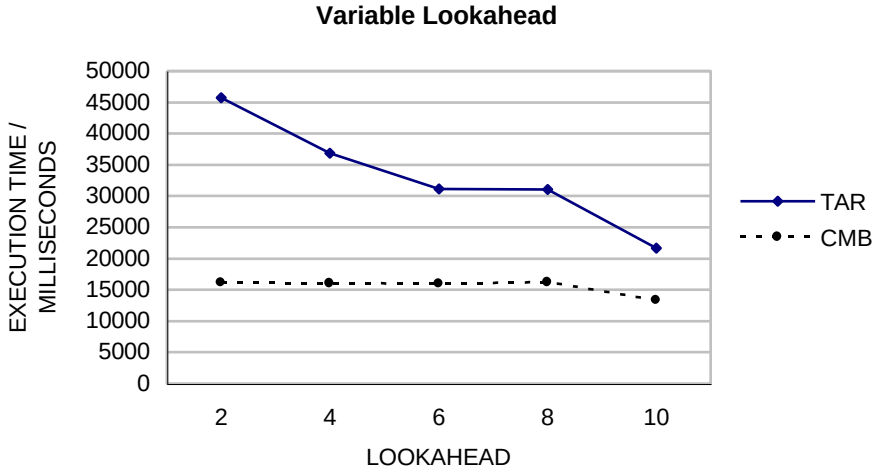


Figure 8: Performance Results for Pipeline Topology

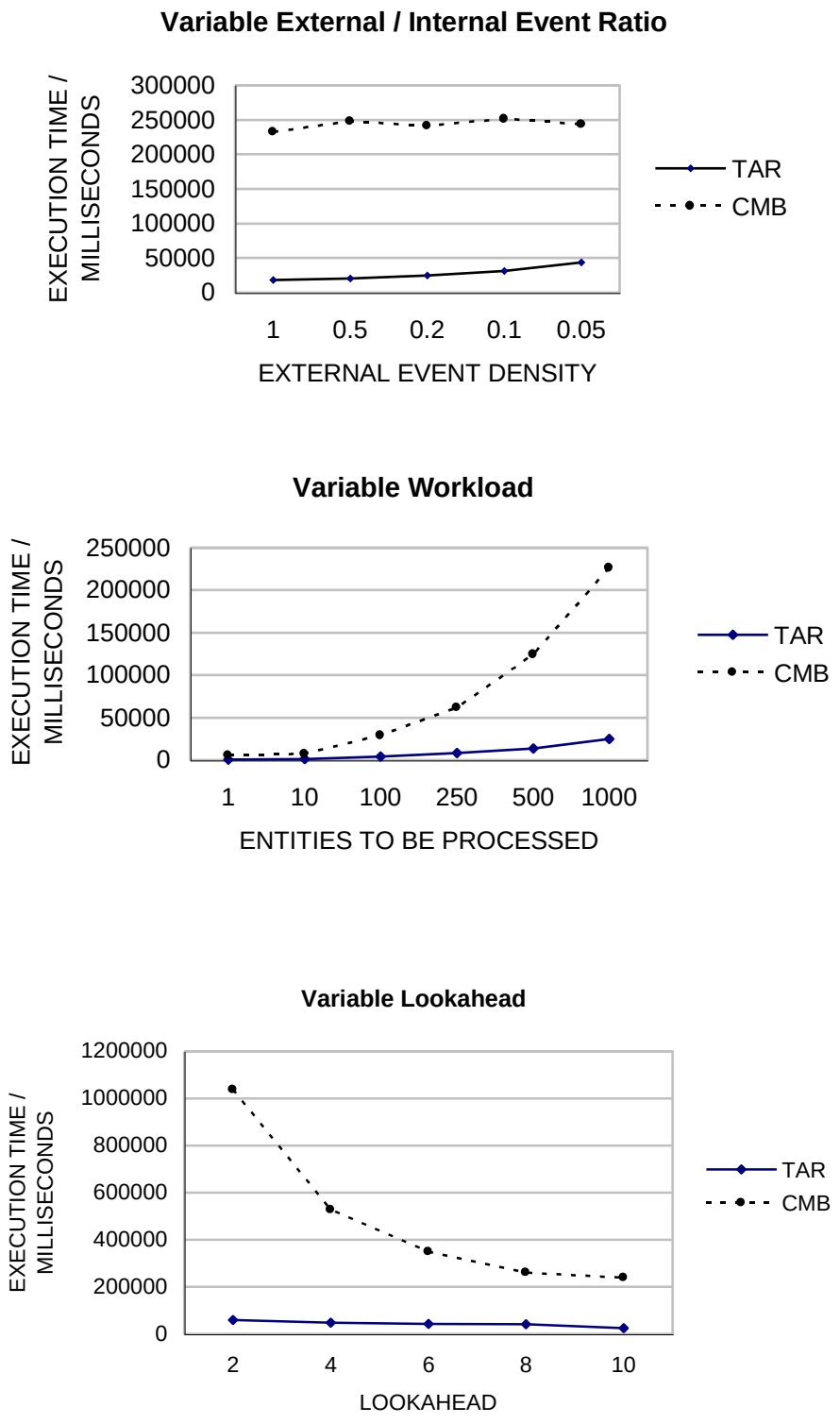


Figure 9: Performance Results for Local Feedback Topology

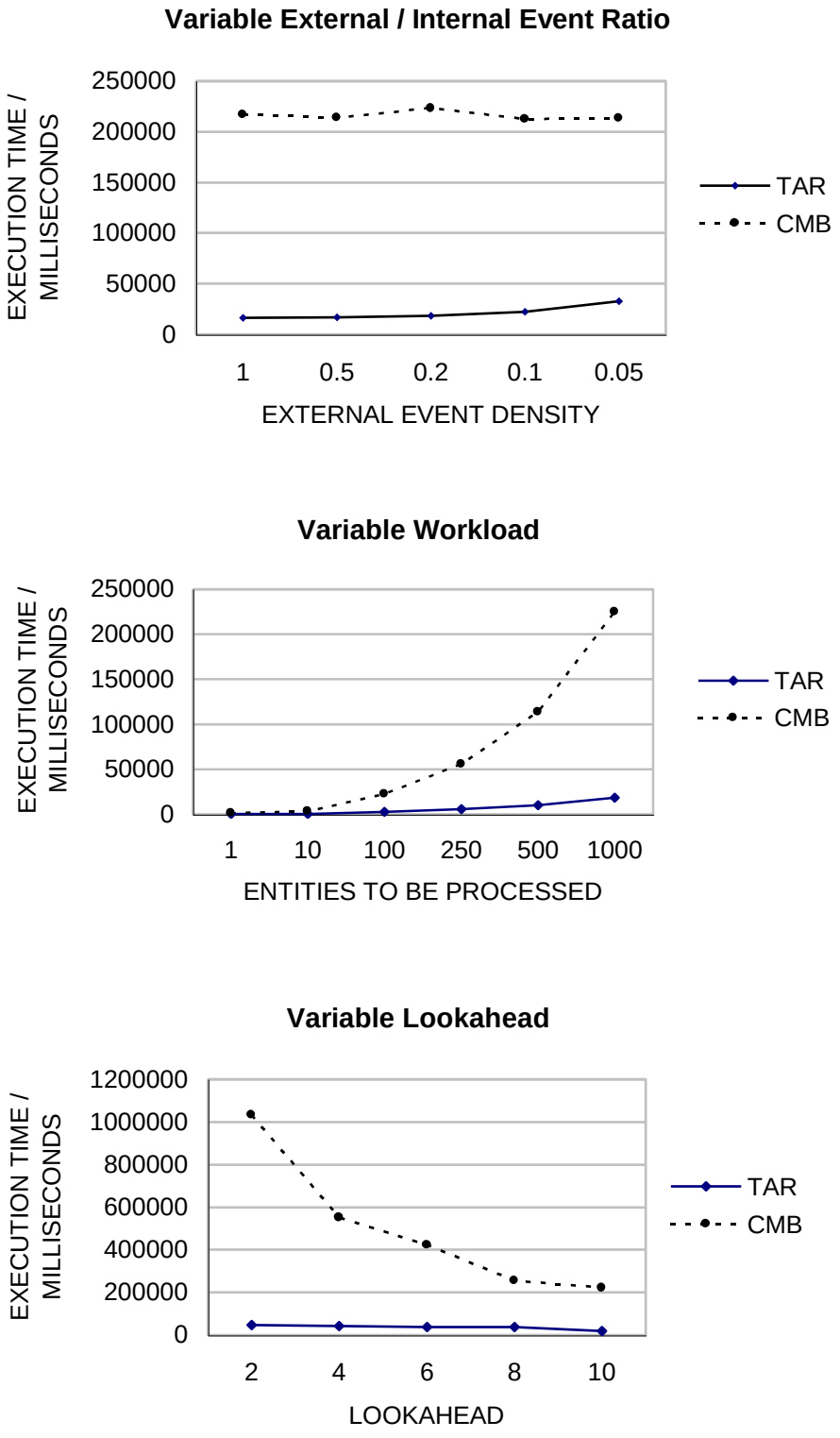


Figure 10: Performance Results for Totally Interconnected Topology

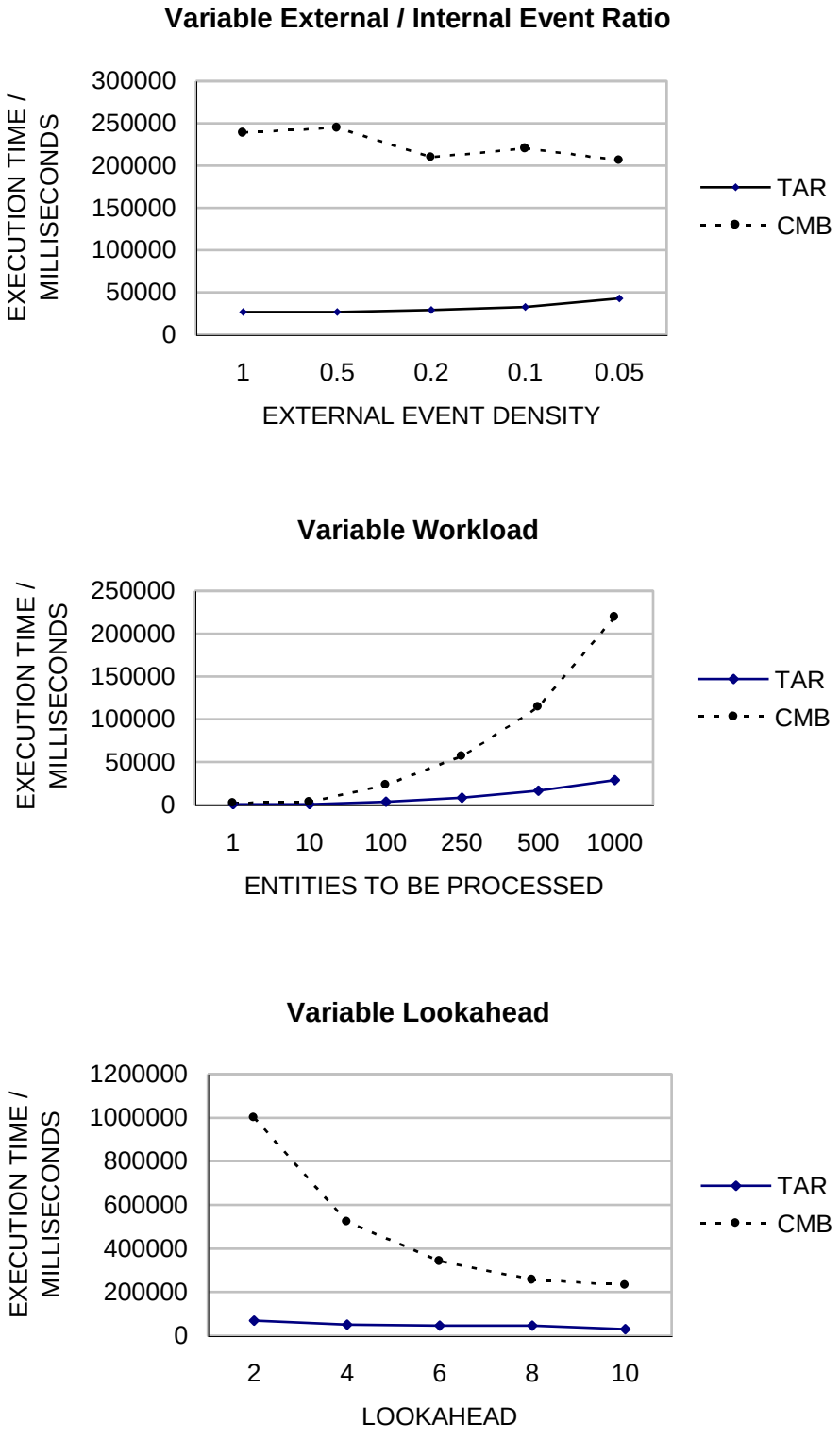


Figure 11: Performance Results for Producer-Consumer Topology