# Migrating to a Real-Time Distributed Parallel Simulator Architecture

**Bernardt Duvenhage and**
**Derrick G Kourie (Role of Masters degree supervisor)**
**Espresso Group, University of Pretoria**
**bduvenhage@csir.co.za and dkourie@cs.up.ac.za**

**Keywords:** peer-to-peer, parallel simulation, discrete time step simulation, publish-subscribe, high real-time frame-rate

## Abstract

A legacy non-distributed logical time simulator is migrated to a distributed architecture to parallelise execution. The existing Discrete Time System Specification (DTSS) modelling formalism is retained to simplify the reuse of existing models. This decision, however means that the high simulation frame rate of 100Hz used in the legacy system has to be retained in the distributed one—a known difficulty for existing distribution technologies due to inter-process communication latency.

A specialised publish-subscribe simulation model is used for the new simulator architecture. The simulation model, including the process synchronisation, is implemented using a low latency peer-to-peer TCP messaging protocol. The TCP send and receive buffers and TCP's Nagle algorithm are also tweaked to ensure low latency communication. Gigabit Ethernet is used at the hardware layer. A parallelised execution speed-up of four to five times is reached with six to eight machines at a simulation frame rate of 100Hz.

## 1. INTRODUCTION

The South-African National Defence Force's (SANDF's) need for decision support and concurrent tactical doctrine development within a Ground Based Air Defence System (GBADS) acquisition program offered an ideal opportunity to establish an indigenous and credible modelling and simulation capability within the South-African defence acquisition environment [1][2]. The broad requirement of the capability is to simulate a GBADS battery of existing and still to be acquired (possibly still under development) equipment and their related human operators at a system of systems level within a realistic Synthetic Environment (SE). During the concept and definition phases of the acquisition life cycle [3] the capability was successfully provided by a non-distributed simulator and its architectural predecessors [4]. A selection of the models were derived from high fidelity engineering models, some by OEMs, and developed within a 100Hz logical Discrete Time System Specification (DTSS) [5] that simplified the time and causality management. The non-distributed simulator evolved within this 100Hz logical time DTSS modelling formalism and was implemented to run As Fast As Possible (AFAP).

Real-time simulation execution became a prioritised requirement during the development phase of the acquisition life cycle due to the realised impact of *realistic* human-simulation interaction when doing tactical doctrine development. Human interaction would happen through an Operator In the Loop (OIL) console with the possibility to record the operator's actions to be re-used in statistical simulation runs when and as required. To support the real-time requirement it was decided to parallelise the simulator across multiple Commercial Off the Shelf (COTS) PC nodes connected with Gigabit Ethernet.

For simplicity in the economical reusability of all the existing models it was also decided to retain the 100Hz logical time and DTSS modelling formalism. To achieve real-time execution, the parallelised logical time DTSS simulator is run AFAP, but the execution is throttled to not exceed real-time. To guarantee causal message delivery for the discrete time execution though, severe real-time constraints is placed on the minimum required inter-node communication latency as each simulation frame is a mere 10ms. The next section introduces existing distributed and parallel simulator technologies and their applicability to a 100Hz DTSS modelling formalism. The following sections then pose a research question on a new simulator architecture and develop the proposed architecture to inform the research question through analysis. The paper finds resolution in the flexibility and performance analysis of the new architecture and a concluding assessment of the architecture's suitability as a distributed parallel, high resolution logical time, DTSS simulator.

Within the Mathematical and Computational Modelling (MCM) Research Group of the Defence, Peace, Safety and Security (DPSS) operating unit of the South-African Council for Scientific and Industrial Research (CSIR) the main author had the responsibility of developing the new distributed parallel simulation architecture. This architecture is under investigation as part of an MSc project in Computer Science with the co-author (role of the Masters degree supervisor) from The University of Pretoria.

## 2. EXISTING DISTRIBUTED AND PARALLEL SIMULATOR TECHNOLOGIES

This section introduces existing distributed and parallel simulator technologies and their applicability to a 100Hz logical time DTSS modelling formalism. Specifically technologies suitable for deployment on a distributed COTS PC infrastructure are discussed.

Looking at the literature, the most popular and thoroughly analysed, distributed simulation technologies within the military domain [6] seem to be Distributed Interactive Simulation (DIS) and the High Level Architecture (HLA), which implement Discrete Event System Specifications (DEVS) rather than a DTSS. The HLA is a generalisation and extension of DIS and the Aggregate Level Simulation Protocol (ALSP), both of which evolved from the Simulator Networking (SIMNET) project. SIMNET is, according to Page and Smith [6], the first meaningful attempt to interoperate military simulators within the United States' Department of Defence.

DTSS may however be embedded [5] within DEVS. Ogata, et al. [7] tested the real-time performance of DIS and different versions of the RTI-NG HLA Run-Time Infrastructure (RTI). Their real-time vehicle model simulation within a 3D graphical environment reached a frame rate ceiling of around 30Hz with both DIS and HLA implementations.

The HLA's real-time performance, for both RTI-NG and DMSO RTI implementations, is also studied by Jolibois, et al [8] in the context of a beyond visual range air to air combat simulation. The performance is shown to be less than ideal for 10Hz and higher simulation frame rates. This is due to message latency, object time advance latency and message deliveries leaking into adjacent simulation time steps.

Fujimoto and Hoare [9] investigated an alternative for the current versions of the HLA RTIs that can achieve latencies that are suitable for high simulation frame rates, but these are based on a low latency Gigabit Myrinet [http://www.myricom.com/myrinet/overview/] hardware layer and specialised RTIs. When Fujimoto and Hoare analysed the latency for DMSO RTI1.3 over an Ethernet TCP and UDP implementation it was found to be in the order of 10ms, which is too large to sustain a 100Hz simulation frame rate. They also found that the DMSO RTI supports a time advance frequency of more than 2000 per second between two nodes, but for three and more nodes the time advance frequency unfortunately dropped sharply to values as low as 10Hz with even only a few objects per node.

Watrous, et al. [10] explain that in the HLA, and in fact in any distributed algorithm, a time management scheme, such as the logical time DTSS modelling formalism, which requires contributions from all other nodes are relatively expensive. For this reason the HLA allows federates (simulation components) to employ their own unconstrained time management to avoid the time synchronisation overhead. In such an unconstrained case each model will synchronise itself against its simulator's wall clock without explicit synchronisation with other models, or between simulators, but at the risk of loosing message causality.

A recent distributed parallel simulator architecture is the Aurora master/worker architecture [11]. According to Park and Fujimoto the goal of Aurora is to harness available computation time from a large number of machines rather than strictly achieving high speed-ups on dedicated hardware. Aurora is unique among its class of distributed parallel architectures in the fact that it is aimed at parallel discrete event simulation (PDES) and, for example, includes simulation time management. It is unfortunately, like HLA, not well suited to tightly coupled high resolution discrete time simulations.

From the indicated examples it is clear that using existing distributed and parallel technologies such as HLA or DIS for implementation of a logical time DTS parallel simulator might introduce technical risks in getting the frame rate to or beyond 100Hz while still ensuring message causality. It is worth noting that this is purely due to HLA, DIS and similar architectures not being designed for high resolution logical time communicating parallel processes. These architectures, in particular HLA, seem rather to be aimed at providing simulation development efficiency, economical interoperability and geographical distribution, typically implementing a Discrete Event System Specification (DEVS) modelling formalism [5].

An important architecture driver is that of simulator interoperability. It is not a present or foreseeable priority [4] within this specific acquisition environment and is not a national imperative at this stage. According to Straßburger[12] these factors further decrease the drive behind and thus viability of the use of interoperability standards.

## 3. RESEARCH QUESTION

*Can a peer-to-peer publish-subscribe simulator architecture implement a logical time DTSS modelling formalism to support a four to five times parallelised 100Hz closed loop simulation?*

The research question firstly enquires whether or not a publish-subscribe simulation model can provide the required flexibility to support the simulation capability. Secondly it enquires whether or not a TCP message passing implementation of the simulation model and simulation synchronisation can maintain real-time execution of the 100Hz logical time DTSS at increased levels of parallelisation.

The publish-subscribe paradigm is chosen for its simplicity and familiarity. The TCP message passing implementation is chosen for TCPs stream based and reliable nature and the fact that COTS PCs with Gigabit Ethernet Network Interface Cards (NICs) and a commercial switch will be used at the physical and data-link layers. Initial TCP messaging tests [13] also revealed that a Gigabit TCP connection could quite possibly support the required low message latencies.

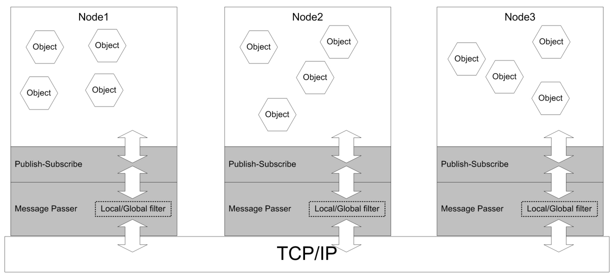

**Figure 1.** Layered Peer-to-Peer Simulator Architecture

## 4. THE PUBLISH-SUBSCRIBE DISTRIBUTED PARALLEL SIMULATOR ARCHITECTURE

The discussion on the new publish-subscribe simulator architecture is structured around the layered architecture of the simulator (shown in Figure 1), which includes a publish subscribe simulation layer, a message passing implementation of the simulation model and at the bottom layer a low latency TCP messaging protocol for Gigabit Ethernet.

The attraction of the layered architecture was the separation of concerns, in terms of design, between the simulation model and the distributed execution thereof. An additional advantage is of course the ability to change the implementation of the bottom layers without affecting the top layer simulation application.

### 4.1. Publish-Subscribe Simulation Model

The top layer simulation model encompasses a couple of aspects, which include the simulation time management, the system specification modelling formalism, the object communication framework and the synthetic environment services.

As mentioned, the pre-existing models have been implemented within a conservative logical time management scheme and a DTSS modelling formalism. It was decided to keep these aspects unchanged to simplify the reuse of the existing models. The object communication framework that is under investigation for the simulation model is a specialised publish-subscribe framework to be discussed next. Discussions on the synthetic environment services will then follow.

#### 4.1.1. The Publish-Subscribe Object Communication Framework

The publish-subscribe paradigm is well known to anyone that has ever needed to organise to get information, for example a magazine, on his or her topic of interest on a regular basis. Each magazine within your topic (category) of interest has a title and a regular interval at which the categorical information is made available (published). You, the subscriber, may request that the information be delivered to your doorstep in the form of, say, a weekly or a monthly magazine issue.

The publish-subscribe simulation framework is a direct analogy to the magazine example. An instance of a simulation model (an object) may express its desire to receive information within a certain category of interest, e.g. aircraft positions, by adding the category (and title name, if known) to its *Subscription Wish List*. An object may also express its willingness to share information within a certain category, such as its own position, by adding a title (name and category) to its *Owned Title List*. A subscribing object has no guarantee that any object will share information under the title category or name in which it is interested. Similarly a publisher object has no guarantee that any other objects will be interested in the information that it is willing to share. At simulation start-up each object's owned title list is made known to the rest of the simulation. The titles are then processed against the objects' wish list subscriptions. Each title matching a wish list subscription generates a subscription which is sent back to the title owner to be added to the title's subscriber list.

At simulation run-time each object will go through regular increment, publish and gather cycles. Within the DTSS modelling formalism an object is incremented every n'th discrete time simulation frame where n is the object's *trigger frame*. Each wish list subscription, and thus each subscriber in a title's subscriber list, is also associated with a trigger frame. During a simulation frame, each subscriber of each owned title will be visited and an issue sent to the subscriber if it is the subscription's trigger frame. An important publish rule that is required to ensure consistent issues is that the contents of a title issue may only be updated during the publisher's increment cycle.

Objects may at simulation run-time express their wish to share a new category of information or a new title within an existing category. This is done by submitting a *run-time title* to the communication framework. Similarly objects may express interest in categories (or titles within categories) of information at run-time by submitting a *late subscription*

An object has an issue pigeon hole for each of its wish list subscriptions. When an issue is received (gather phase) it is placed in the appropriate pigeon hole. A pigeon hole may have subscription history turned off or on. If history is off then a newer version of an issue replaces all old issues that may remain in the pigeon hole. If history is turned on then issues will be added to the pigeon hole in chronological order. The object may then read issues and manually delete them as required during increment cycles. Turning history on for a specific wish list subscription is typically required when a subscriber doesn't want to miss any important updates (events) for that subscription. Having history off allows the subscriber to always have access to the current issue without the overhead of always caching and processing a subscription's recent history.

#### 4.1.2. The Synthetic Environment Services

The two types of simulation services supported are, firstly, low level services that are built into the simulation model and, secondly, high level services that run on top of the simulation model as simulation objects. The only low level service currently implemented is that of delayed issues. An issue may be given a future delivery time by either the publisher, or the subscriber upon delivery. Such an issue would be delivered to the subscriber immediately, but once there it resides in a delayed issue list until the time of delivery arrives at which point the issue is put into the appropriate pigeon hole of the subscriber. Delayed issues are handy if transmission delays of messages within the SE are to be modelled. In the current simulator the issue delays of tactical communication subscriptions are, when required, calculated by a radio and cable network model.

High level synthetic environment services subscribe to the objects' state titles and then apply environmental tools such as Line Of Sight (LOS) and terrain engines to give each object individual feedback on its height, which objects it can see, etc. To accomplish the personalised feedback a service advertises what is called a differentiated title. Each time a subscription is made to a differentiated title the simulator automatically creates a personalised title and subscription for the subscriber. The service may then use the created titles to publish to individual objects.

A service need not always publish data back to the simulation, though. Logging, for example, is a high level service that accumulates object states and other information. The logging service may then apply user configured data analysers to the accumulated data and log the results to disk.

## 4.2. Peer-to-Peer Message Passing and Node Synchronisation

The publish-subscribe communication framework and the simulator synchronisation is implemented with a peer-to-peer message passing architecture. A peer-to-peer architecture is specifically preferred above a client-server architecture to avoid the double latency that exists when communicating via a server to a third machine. The messaging implementation of the publish-subscribe communication framework is presented, followed by the implementation of the simulation synchronisation.

#### 4.2.1. Messaging Implementation of Publish-Subscribe

The publish-subscribe framework naturally translates to a messaging architecture containing only three message types. A title may be advertised as a title message containing all the title and publisher details. A wish list subscription may similarly be a message containing the details of the wish list subscription and the subscriber. The third message type is an issue message that contains the subscriber's node-number delivery address, the targeted wish list subscription pigeon hole and the actual issue payload. The messaging implementation has a local/global filter (see Figure 1) that will loop a node's self addressed messages back to be cached for the next simulation frame without passing anything down to the TCP layer.

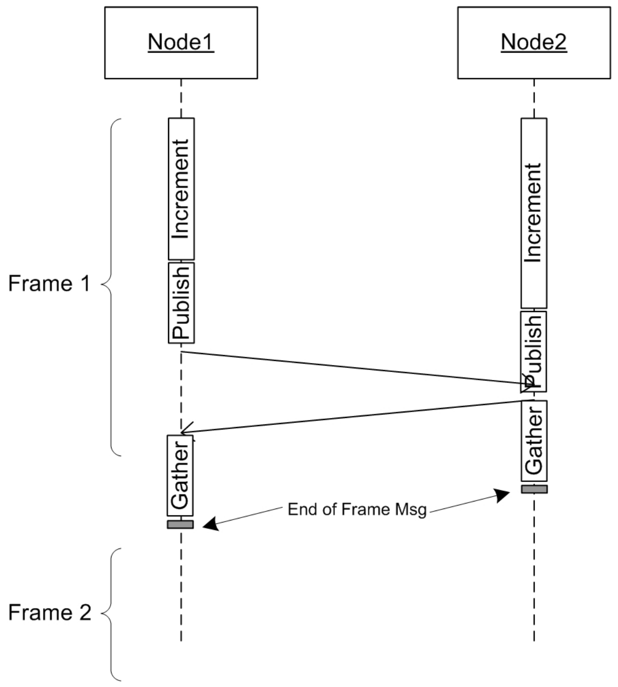

**Figure 2.** Peer-to-Peer Message Passing and Simulation Synchronisation

#### 4.2.2. Peer-to-Peer Node Synchronisation

The peer-to-peer synchronisation scheme is shown in Figure 2. Each simulation frame has three consecutive execution phases. Within the first phase, which is the *increment phase*, all the objects are put through their increment-publish cycles. The published issues are not messaged directly, but are grouped per destination node and cached until the second, so called publish, phase. The cached issue groups may now be sent to their respective destination nodes. The publish phase must be followed by a time-stamped end-of-frame message to each peer node to signify that all the issues for the current simulation frame have been sent. The end-of-frame messages perform a similar function as Chandy-Misra null messages [14] for dead-lock avoidance and time management in DEVS implementations. A simulator node will wait in the gather phase until it has received and processed an end-of-frame message from each of the other simulator nodes after which it starts with the increment phase of the next simulation frame.

## 4.3. TCP Message Passing Implementation

The TCP messaging implementation consists of two components. The first of which is an address translation from destination node number to destination IP and port before any message can be sent via TCP. This translation is pre-configured and fixed for each distribution configuration.

The second component is a two-tiered approach to lowering TCP message latency. The first tier is to ensure that as much as possible of the TCP send and receive overhead happens in parallel to the node execution. This is accomplished by increasing TCP's send and receive buffers to an adequate size such that the buffers have enough space for two simulation frames worth of data. This ensures that all TCP sends are non-blocking. It also facilitates CPU time, from a second CPU or hyper-thread or that's not used by the simulation, to be used to transport as much data as possible from the nodes' send buffers across TCP to their receive buffers for quick retrieval when needed.

The second tier takes control of the TCP message send times. TCP's Nagle algorithm tries to optimise bandwidth usage by conglomerating sent messages in the send buffer until it is large enough to fill a TCP packet or until a certain timeout is reached. The unfortunate side effect of the Nagle algorithm is that control over message latencies is lost. To give control over the message latency back to the simulator the Nagle algorithm is disabled.

## 5. ANALYSIS AND RESULTS

The two aspects of the simulator architecture to be analysed in support of the research question are, firstly, the simulator's applicability to a 100Hz DTSS modelling formalism and, secondly, the flexibility of the publish-subscribe simulation model and its suitability for a system of systems tactical and SE simulation.

## 5.1. Experimental Setup

The simulator nodes are similar Pentium 4 3.2GHz machines with 2GB of dual-channel RAM each and WindowsXP SP2. The network infrastructure is, as mentioned, Gigabit Ethernet with a D-Link DGS-3324SR managed switch. Each node has an Intel D945PAW mother board with an on-board Intel Pro/1000 PM Gigabit Ethernet network card.

The simulator nodes will be populated with instances of a "test" model. The test model has a fixed processing requirement of 1ms per 10ms simulation frame and an owned title with a fixed issue size of 512 bytes. Furthermore each instance of the test model subscribes to every other instance, creating the worst case communication scenario of a fully connected communication graph.

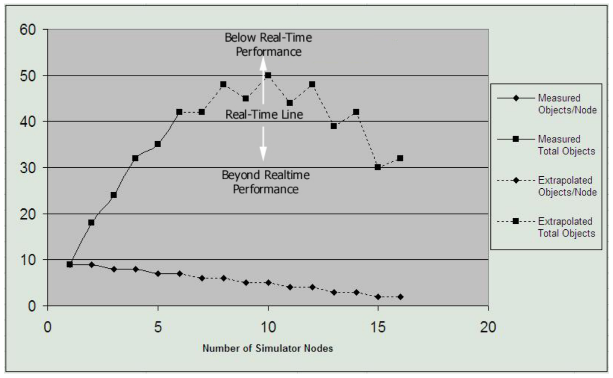

**Figure 3.** Total Object Performance of 100Hz Peer-to-Peer Simulator

## 5.2. Applicability of the Peer-to-Peer Simulator to a 100Hz DTSS Modelling Formalism

The proposed architecture's real-time performance is analysed over distributions of one to six simulator nodes on the target infrastructure. With each node configuration the number of objects per node will be limited to achieve a real-time frame-rate. Finally a simple predictive model for the distributed performance behaviour is derived from the analysis data and used to do a first order estimate of the simulator's scalability to seven and more nodes. Accurate evaluation over more nodes should however be part of the future work section to verify the speculation about the simulator's scalability.

The performance result that is recorded is the maximum number of objects per node (see Figure 3) such that the simulation can still reach real-time. If the total number of objects are increased above the "Total Objects" graph, the performance will drop below real-time. Conversely, if the total number of objects are decreased below the "Total Objects" graph, the performance will grow beyond real-time. Both the total number of objects and the performance speed-up graphs are derived from the measured objects-per-node graph (Figure 3 and Figure 4).

Quantifying the measured communication overhead it seems that each time a simulator node is added, the number of model instances per node must be decreased by an average of 0.5 to maintain real-time which is a 0.5% overhead of the 10ms simulation frame. The explanation of these results is quite likely not a simple task, see the Section 7. on future work, as it may be dependent on multiple factors such as message structure and grouping. However, assuming for the purpose of first order performance predictions, that the results do indeed indicate a linear distribution overhead

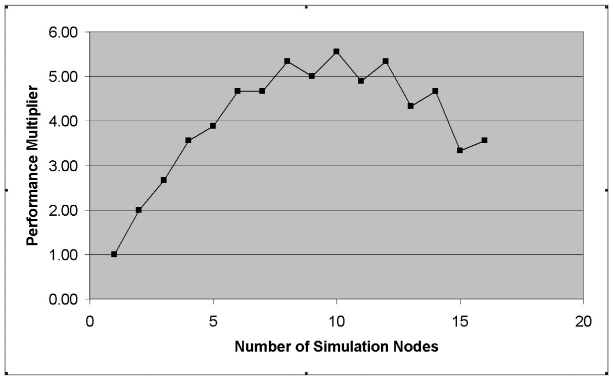

**Figure 4.** Real-Time Performance Speed-Up of 100Hz Peer-to-Peer Simulator

of 5% for each simulator node added, such a linear overhead would most probably be in the receive loop of each simulation frame. Amdahl's Law [www.wikipedia.org] specifies that the speed-up attainable by parallel execution is limited by the sequential components of the system which in this case is a single NIC and thus a single, though full-duplex, communication channel per simulator node.

A linear performance might seem counter intuitive to what is expected of an $n$ node and fully connected peer-to-peer structure where the total number of connections grows by $n^2$. The linear nature does however make sense if one remembers that the processing is done by $n$ nodes resulting in a processing time of $\frac{n^2}{c.n}$ which is proportional to $n$ and therefore linear. In other words, each node must receive data from each of the other nodes in turn, limiting the potential parallelisation.

The first order objects-per-node performance for seven and more nodes is estimated by linearly extrapolating the measured objects-per-node curve (Figure 3) under the previous assumption. The spikiness of the performance graphs is due to the granularity of the objects which, in general, leaves a fragmented processing slot (idle time) on each node. The linear extrapolation provides an estimate for the scalability of the simulator, but as the number of nodes increases to beyond 10 the total number of objects eventually start to decrease which implies that the communication bandwidth will also decrease again. Around this point it is expected that the linear nature of the objects-per-node curve might change which requires, as mentioned, analysis over more nodes to draw accurate scalability conclusions beyond 10 nodes.

## 5.3. Suitability and Flexibility of the Publish-Subscribe Simulation Model

The flexibility of the publish-subscribe simulation model and its suitability for a DTSS system of systems tactical and SE simulation is analysed to resolve the first part of the research question. The publish-subscribe simulation model is shown to be suitable for the implementation of a DTSS modelling formalism and the simulator synchronisation to be free of deadlock which is sometimes a problem for distributed simulators. Finally the measure of flexibility of the simulation model is defined and demonstrated as the range of application domains within which the simulator have been successful.

**Table 1.** Successful Application Domains of the Peer-to-Peer Publish-Subscribe Simulator Architecture

| Application Domain | Description |
|---|---|
| GBADS | Ground Based Air Defence System Performance Analysis |
| MobADS | Mobile Air Defence System Performance Analysis |
| Navy SBADS | Concept Demonstrator for Developing Tactical Doctrine for Naval Air Defence Scenarios |
| Sensor Webs (Awarenet) | Concept Demonstrator for value of additional coastal surveillance in creating situational awareness when patrolling South-Africa's fishing zones |

The publish-subscribe simulation model sets up virtual communication channels between the objects. These channels provide a way for a subscriber to sample the publisher's state periodically as required within the DTSS modelling formalism [5]. This is accomplished by the subscriber periodically receiving the current issue of a certain title of the publisher.

The end of frame notifications are a special case of the Chandy-Misra null messages [14] with a fixed look ahead equal to the discrete time step. The simulation model may thus, according to Chandy-Misra, be shown to be deadlock free. An advantage of the DTSS implementation of null messages is that the number of null messages is limited to only one per simulator node per simulation frame. This improves on the typical null-message overhead within DEVS implementations which may generate excessive null-messages [15].

The flexibility of the simulation model is defined as the range of application domains, (see Table 1), within which this publish-subscribe simulation model and parallel peer-to-peer simulator have been successful. Each of these application domains has been validated against reality by running experiments that analyse various aspects of the system of systems. For each experiment the systems' simulated emergent behaviour is compared against the pre-defined expected behaviour and motivated, or corrected, by subject matter experts or from what is already known of the system. A list of external systems successfully integrated with the simulator may be found in Table 2.

**Table 2.** External Systems Successfully integrated with the Peer-to-Peer Publish-Subscribe Simulator Architecture

| External System | Description |
|---|---|
| Simulation Viewers | Integration with 2D and 3D online and offline visual analysis tools |
| OIL Consoles | Integration with mock-up OIL consoles for realistic real-time operator-equipment interaction |
| Hardware In the Loop (HIL) Tracking Sensor | Integration with a Mechanised Optical and Radar Tracker (MecORT) for real sensor input when doing system analysis and validation |
| HIL Air Picture Sources | Integration with civilian and military air pictures supported by run-time simulated aircraft generation |
| Flight Simulator | Integration with a flight simulator for inclusion of realistic reactive pilot behaviour when doing system analysis. |

## 6. CONCLUSION

The new 100Hz logical time DTSS publish-subscribe peer-to-peer simulator architecture achieves a measured speed increase, due to execution parallelisation, of above 4.5 when distributed over six simulator nodes. This equates to a distribution efficiency of 75%. The wide success of the application of the simulator architecture is used to motivate the publish-subscribe simulation model's suitability as a general purpose DTSS simulation model. The authors therefore conclude that the peer-to-peer publish-subscribe simulator is suitable to support the real-time execution of the 100Hz logical time DTSS simulation requirement.

The simulator's 100Hz logical time performance is also explained and modelled in a simple way which provides a first order estimate on the scalability of the parallelisation. The simulator is estimated to reach a parallelisation ceiling at a speed increase of approximately 5.5 which is achieved when distributed over 10 simulator nodes. This equates to a distribution efficiency of 55%. The scalability of such a high resolution logical time parallel simulator thus seems to be limited to applications requiring low to medium levels of parallelisation. The limiting factor for the parallelisation is, as mentioned, the sequential nature of the network communication.

The DTSS modelling formalism does seem to pose a technical difficulty in implementing large scale parallelisation of high resolution logical time simulations. In hind sight it seems like a good idea to rather develop a hybrid DTSS-DEVS modelling formalism, that has a DEVS layer enveloping the DTSS layer, to further migrate this specific simulation capability towards supporting large scale parallelisation. The two layer approach allows the existing DTSS models to be grouped and aggregated into systems level models for example, which may then be better suited to a DEVS modelling formalism. The DEVS layer then communicates only what is required and its parallelisation is not constrained by the underlying DTSS layer's logical time resolution.

## 7. FUTURE WORK

The reason for retaining the DTSS modelling formalism was for ease of reuse of existing discrete time models. In other words the authors believe that developing this parallel distributed simulator was more viable than, for example, migrating the entire modelling formalism to DEVS. This is a valid position for the currently required simulation capability, but future work to increase the scalability and usability of the simulator should include:

- Further investigation and a proper explanation of the scalability behaviour which might reveal ways of improving that behaviour,
- investigation into the different timings and groupings for execute-send cycles, possibly interleaving send with execute in a different way,
- investigation into using lower overhead UDP message passing, because in a dedicated and lightly loaded switched Gigabit Ethernet scenario it is known [16] that UDP packet losses occur very seldom, and
- investigation into the possible migration of the system specification modelling formalism to a hybrid DEVS-DTSS modelling formalism.

## 8. ACKNOWLEDGEMENTS

The authors would like to thank both the Armaments Corporation (Armscor) of South-Africa and the CSIR for supporting this research. Additionally the authors thank Anita Louis (CSIR) and Arno Duvenhage (CSIR) for their valuable review comments.

## REFERENCES

[1] Johannes Lodewikus Pretorius. Feasibility considerations for a tailored simulation based acquisition (sba) approach. Master's thesis, University of Pretoria, 2003.

[2] Jacques Baird and Cobus Nel. The evolution of m&s as part of smart acquisition using the sandf gbads programme as an example. In *Proceedings of the 12th European Air Defence Symposium*, volume 3694, pages 173–182, 2005.

[3] Shahen Naidoo and Cobus Nel. Modelling and simulation of a ground based air defence system and associated tactical doctrine as part of acquisition support. In *Proceedings of the 2006 Fall Simulation Interoperability Workshop*, 2006.

[4] Willem H. le Roux. Implementing a low cost distributed architecture for real-time behavioural modelling and simulation. In *Proceedings of the 2006 European Simulation Interoperability Workshop*, 2006.

[5] Bernard P. Zeigler. *Theory of Modelling and Simulation*. Academic Press, 2000.

[6] Ernest H. Page and Roger Smith. Introduction to military training simulation: A guide for discrete event simulationists. In *Proceedings of the 1998 Winter Simulation Conference*, 1998.

[7] Michihiko Ogata, Akira Higashide, Mike Cammarano, and Toshinao Takagi. Rti performance in the distributed real-time vehicle model simulation in a 3-d graphical environment. In *Proceedings of the 2001 European Simulation Interoperability Workshop*, 2001.

[8] Stephane Jolibois, Thierry Joubert, and Herve Wentzler. New hla based technologies and methods for an advanced air to air combat simulation. In *Proceedings of the 2003 European Simulation Interoperability Workshop*, 2003.

[9] Richard Fujimoto and Peter Hoare. Hla rti performance in high speed lan environments. In *Proceedings of the 1998 Fall Simulation Interoperability Workshop*, 1998.

[10] Ben Watrous, Len Granowetter, and Douglas Wood. Hla federation performance: What really matters? In *Proceedings of the 2006 Fall Simulation Interoperability Workshop*, 2006.

[11] Alfred Park and Richard M. Fujimoto. Aurora: An approach to high throughput parallel simulation. In *Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation*, 2006.

[12] Steffen Straßburger. *Advances in Simulation*. SCS Publishing House, 2000.

[13] Bernardt Duvenhage and Herman W. le Roux. Tcp simulation architecture investigation. Technical report, Council for Scientific and Industrial Research, 2004.

[14] K. Chandy and Jayadev Misra. Distrubuted simulation: A case study in design and verification of distributed programs. In *IEEE Transactions on Software Engineering*, volume SE-5, 2003.

[15] Richard M. Fujimoto. Parallel and distributed simulation. In *Proceedings of the 1999 Winter Simulation Conference*, 1999.

[16] Behrouz A. Forouzan. *TCP/IP Protocol Suite*. McGraw-Hill, Inc., New York, NY, USA, 2002.

## Biography

Bernardt Duvenhage obtained his B.Sc (Honour) degree in Computer Science from the University of Pretoria in 2005 and is currently pursuing a Masters Degree. While part of the Mathematical and Computational Modelling Research Group of the Council for Scientific and Industrial Research (CSIR) in South Africa, he played a key role in developing the group's distributed simulator architecture; the simulation's terrain and LOS services; and the 3D visualisation and analysis tool of the synthetic environment. He is currently employed in the Optronic Sensor Systems Competency Area of a division within the CSIR. He intends further research in virtual environment simulation and visualisation.

Derrick Kourie lectures in the Computer Science department at Pretoria University. While his academic roots are in operations research, his current interests include, but are not limited to software engineering and algorithm development. He is student adviser to some 20 postgraduate students working in these and related areas. He is editor of the South African Computer Journal and serves on various national and international academic committees.