AN ALTERNATIVE TO DEAD RECKONING FOR MODEL STATE QUANTISATION WHEN MIGRATING TO A QUANTISED DISCRETE EVENT ARCHITECTURE

Arno Duvenhage and Bernardt Duvenhage
The Council for Scientific and Industrial Research
Pretoria
South-Africa
Email: aduvenhage@csir.co.za, bduvenhage@csir.co.za

KEYWORDS

Quantised discrete event, state quantisation, state estimation, dead reckoning, pure pursuit, alpha-beta filter

ABSTRACT

Some progress has recently been made on migrating an existing distributed parallel discrete time simulator to a quantised discrete event architecture. The migration is done to increase the scale of the real-time simulations supported by the simulator. This however requires that the existing discrete time models be modified to work within the quantised discrete event environment. To this end the use of model state quantiser and quantised integrator pairs are required. An alternative to dead reckoning is suggested for the quantized integrator algorithm: a state estimation algorithm that has successfully been used to inject live aircraft into a discrete time simulator.

INTRODUCTION

The South African National Defence Force (SANDF) currently requires a system of systems simulation capability for supporting the different phases of a Ground Based Air Defence System (GBADS) acquisition program as discussed by Baird and Nel(Baird and Nel, 2005) and others(Oosthuizen, 2005)(Naidoo and Nel, 2006). A non-distributed, fast-as-possible discrete time *synthetic environment* simulator was developed by the Council for Scientific and Industrial Research (CSIR) in support of the simulation capability during the *concept and definition* phases of the acquisition life cycle as detailed by le Roux(le Roux, 2006). The acquisition life cycle is part of the system life cycle shown in Figure 1. Real-time simulation execution has, however, now become a prioritised requirement to support the development phase of the acquisition life cycle.

In support of the real-time requirement a distributed parallel simulator was implemented. The logical discrete time management and modelling approach was kept without modification to economically reuse the existing models. Duvenhage and Kourie (Duvenhage and Kourie, 2007) have since shown that the distributed discrete time architecture adequately supports the currently required parallelisation speed-up, but that a parallelisation speed-up ceiling exists just beyond the current use. Progress has been made on the migration of the discrete time simulator to a quantised discrete event architecture as discussed by Duvenhage(Duvenhage, 2008). It is well known from the literature, (Zeigler et al., 2000) among others, that a discrete event approach to modelling a dynamical system is more efficient than a discrete time approach, both in terms of the communication bandwidth and the model execution time. According to Duvenhage(Duvenhage, 2008), selective application of quantiser and quantised integrator pairs (QQIPs) have shown promise in creating quantised DEVS envelopes around the existing discrete time models that significantly improve upon the simulator scalability.

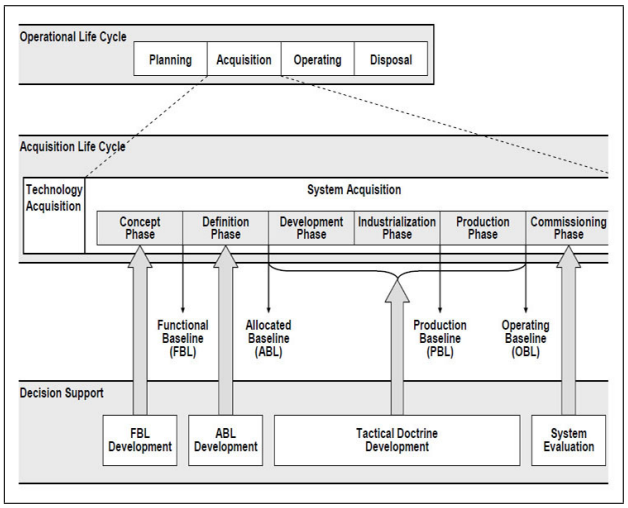

Figure 1: The System Life Cycle(Naidoo and Nel, 2006)

Live aircraft injection refers to modelling an aircraft through state estimation in real-time when only plot and track data from sensors are available through the relevant command and control (C2) links. This has successfully been used to engage live aircraft with simulated air defence batteries as detailed by Duvenhage (Duvenhage, 2007) within the SANDF GBADS environment. Aircraft injection can also be used to facilitate interoperability or collaboration between different C2 systems and simulators.

This article will briefly discuss the proposed quantised discrete event approach. The focus of the article is however the use of the model state estimator, as described in

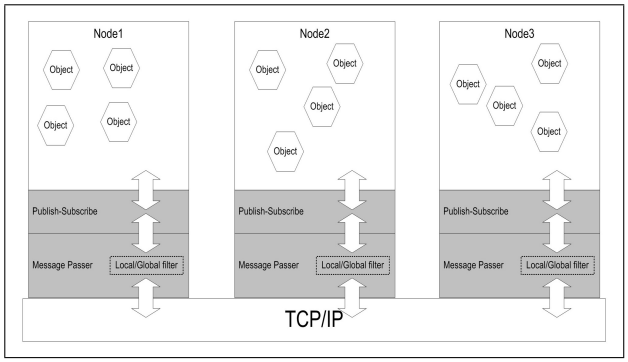

Figure 2: The Layered Discrete Time Architecture.(Duvenhage, 2008)

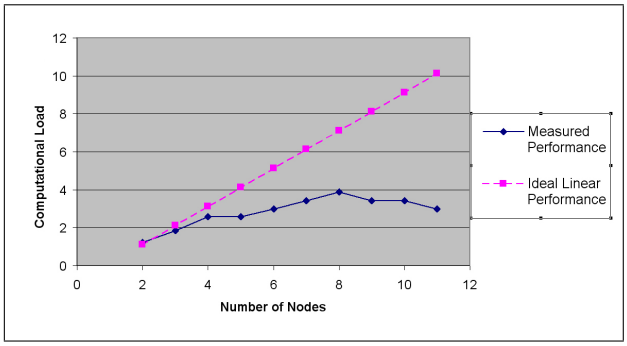

Figure 3: The Discrete Time Architecture's Parallelisation Speed-Up Against Number of Processing Nodes.(Duvenhage, 2008)

(Duvenhage, 2007), as a quantised integrator of a QQIP.

MIGRATING TO A QUANTISED DISCRETE EVENT SIMULATOR ARCHITECTURE

The distributed parallel discrete time architecture is a peer-to-peer message passing architecture discussed by Duvenhage and le Roux(Duvenhage and le Roux, 2007) with a publish-subscribe simulation model layered on top of it as shown in Figure 2. Many of the existing models have evolved from high fidelity engineering models often based on numerical method solutions. This resulted in a modelling dependence on a high 100Hz discrete time simulation frame rate and logical time management. In case of real-time execution the simulation execution is throttled to not exceed the wall clock time, but nothing can be done if the simulation consistently runs slower than real-time.

Duvenhage and Kourie (Duvenhage and Kourie, 2007) have analysed the performance of the discrete time architecture. It was found that the architecture has a *real-time* parallelisation speed-up ceiling of around 4 while its current use often requires a parallelisation speed-up of 3.4. The parallelisation speed-up (also referred to as computational load) performance of the discrete time architecture is shown in Figure 3. The performance of an ideal distributed simulator is also shown. An ideal distributed simulator's distribution overhead has no impact on the performance.

Notice that in the case of the discrete time simulator, its maximum parallelisation speed-up is reached on 8 processing nodes. Adding more processing nodes to the simulator would not increase the overall performance, but actually decrease it due to the added distribution overhead. The authors feel that the architecture's current usage is too close to the speed-up ceiling and a quantised discrete event simulator is proposed by Duvenhage(Duvenhage, 2008) as a future simulator migration step.

The proposed discrete event architecture aggregates the existing discrete time models into groups and then wraps the groups within quantised discrete event envelopes. This concept as applied within the GBAD system of systems model is shown in Figure 4.

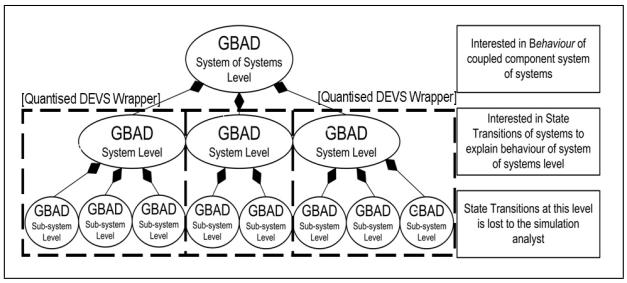

Figure 4: DEVS Envelopes are Wrapped Around Groups of Discrete Time Models.(Duvenhage, 2008)

The parallelisation speed-up behaviour, shown in Figure 5, of an experimental implementation of the proposed architecture resembles that of an ideal distributed simulator. This is due to the fact that the architecture is now processor limited and not communication bandwidth limited as was the case in the discrete time simulator. It should be noted that the experimental implementation enveloped the existing discrete time models and the same bench mark scenarios were used for both simulator architectures.

Being bandwidth limited, as in the case of the discrete time simulator, implies that the simulator's performance may be improved by upgrading the communication infrastructure. This is unfortunately not always feasible. On the other hand, being processor limited, implies that the available distributed processing power of the architecture is well utilised. It also has the additional advantage that the simulator's scalability may continually be improved by adding processing nodes or upgrading the existing processing nodes.

MODEL STATE QUANTISATION

State quantization can be divided into two parts as shown in Figure 6. The dead reckoning (DR) algorithm can be used to do model state quantization. This section will however discuss the use of the state estimation algorithm described in (Duvenhage, 2007) as an alternative to dead reckoning for quantized state integration. An alternative

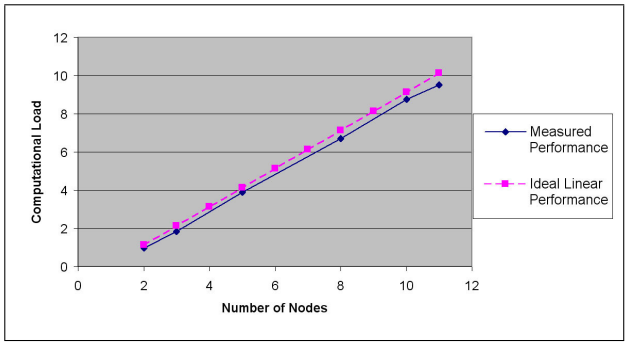

Figure 5: The Discrete Event Architecture's Parallelisation Speed-Up Against Number of Processing Nodes.(Duvenhage, 2008)

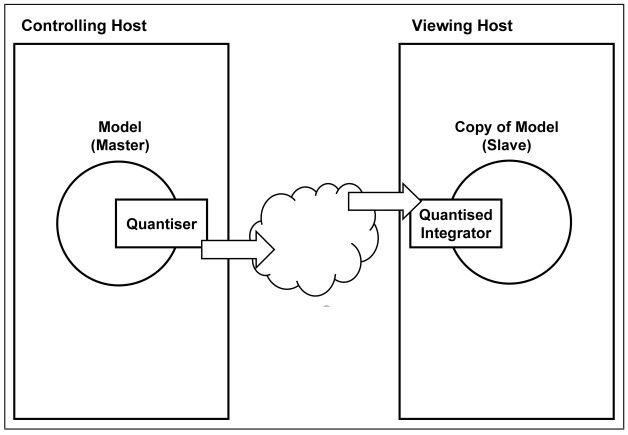

Figure 6: State quantiser and quantised integrator deployment

state quantiser is used in this case.

Dead Reckoning

Dead reckoning is a technique used by simulation technologies like the Distributed Interactive Simulation (DIS) and High Level Architecture (HLA) to reduce the frequency and consequently the bandwidth requirements of distributing state updates and deals with network issues. This is done by predicting a model's state on all hosts until the model's controlling host deems a state update necessary. The model's controlling host will send out a state update if the prediction error becomes too large. These error correcting updates will correct the model state and cause a state update jump on all hosts viewing the model. Larger errors are tolerated if bandwidth requirements are more strict. DR can be setup with several prediction algorithms depending on the model's expected behaviour (Duvenhage, 2007).

The dead reckoning algorithm can be used to do both the quantization and integration of the model state. The dead reckoning state update *jump* in the reconstructed or integrated model state can also be smoothed by using time stamped updates and *phasing in* position updates. It is important to note that the quantiser and integrator should be setup with the same prediction algorithm (Fujimoto, 2000).

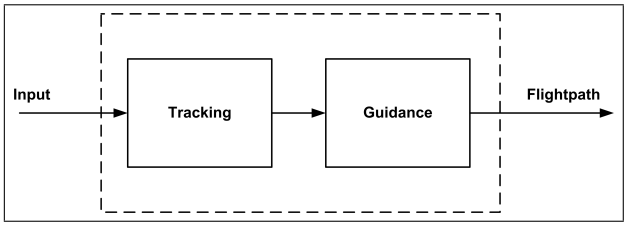

Figure 7: The Tracking and Guidance Algorithm (Duvenhage, 2007)

The Aircraft Injection Algorithm

The aircraft injection (state estimation) algorithm shown in Figure 7 includes both state estimation (tracking) and guidance.

The basic principle is that the guidance component guides the simulated aircraft from its current position on an ad hoc basis (in real-time) to position predictions calculated by the tracking component. The position predictions are made for the next expected update time each time a new state update is received.

The tracking component is responsible for:

- Estimating the position of the aircraft from the state updates,
- estimating the time of the next state update,
- compensating for temporal variance and lag in state updates, and
- mitigating possible lag caused by the guidance algorithm by being able to predict guidance input into the future.

The tracking component can be a kinematic tracking filter like an $\alpha-\beta$ or $\alpha-\beta-\gamma$ type tracking filter: These are 2nd and 3rd order kinematic filters that are easy to implement compared to the more complex Kalman, particle and Interacting Multiple Model (IMM) type tracking filters that may also be used (Blackman, 1986).

The guidance component is responsible for:

- Guiding the aircraft to the predicted aircraft positions on an ad hoc basis (in real-time as new predictions are made),
- calculating the aircraft orientation if required, and
- limiting aircraft movement to stay within the dynamics of the relevant aircraft type.

Guidance laws that can be used for the guidance component are pure pursuit and proportional navigation. Pure pursuit guidance can also be called a tail chase algorithm since the pursuer always tries for a trajectory directly toward the target. Proportional navigation on the other hand over steers in an attempt to keep the line of sight angle between the pursuer and target constant. The line

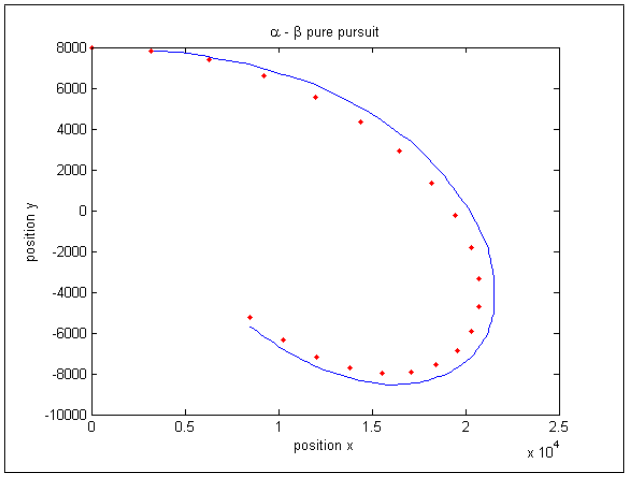

Figure 8: Target (dotted line) and output of alpha-beta-pure-pursuit algorithm (solid line) (Duvenhage, 2007)

of sight angle can be defined as the angle between *the line of sight from the pursuer to the target* and the pursuer trajectory (Duvenhage, 2007).

Alpha-beta-pure-pursuit Quantized Integrator

The aircraft injection algorithm can be called an *alpha-beta-pure-pursuit* algorithm since it uses an $\alpha - \beta$ tracker and a pure pursuit guidance algorithm (Duvenhage, 2007). Figure 8 shows the output of this algorithm for a specific set of state updates. The algorithm accepts the state updates in real-time and always outputs an up to date estimation of the model state.

The $\alpha - \beta$ filter is a 2nd order kinematic filter that assumes the model moves at a constant velocity between measurement updates. This is not the case for maneuvering or accelerating models, although the assumption may hold for sections of the model's path. This specific implementation allows for variable update rates and delayed observations (also using time stamped updates) (Duvenhage, 2007).

The pure pursuit guidance moves the pursuer (or model in this case) directly toward its target. Pure pursuit guidance will cause the pursuer to trail the target and intercept it from behind. The model's orientation can be calculated based on the model's acceleration and velocity (Duvenhage, 2007).

Alpha-beta-pure-pursuit Quantiser

The $\alpha - \beta$ filter requires updates to be able to stabilize for the integrator to reconstruct the model state with the minimum amount or error. One option is to simply sample the model state at regular intervals and use that as the quantized states. This will allow the integrator to stabilize as well as minimize the communications bandwidth requirements of the distributed simulation.

CONCLUSION

Progress has recently been made, as mentioned, on migrating an existing distributed parallel discrete time simulator to a quantised discrete event architecture. The migration is investigated to increase the scale of the real-time simulations supported by the simulator. Such a migration is however economically possible only if the existing discrete time models are reused. This implies that the discrete time models be enveloped in quantised discrete event wrappers.

From the discussion it is clear that the aircraft injection algorithm could be used as a quantised integrator algorithm. This offers a good alternative to dead-reckoning with the major advantages being that there already exists a proven implementation of the aircraft injection algorithm within the SANDF GBADS environment. The algorithm also doesn't require matching of prediction algorithms between the different simulation hosts since the tracking algorithm can estimate the model state based on arbitrary updates. One drawback however is that the quantiser will have to send more updates, compared to the active quantization done by DR, for the integrator to stabilise fast enough.

FUTURE WORK

Future work on using more advanced tracking and guidance algorithms will contribute to the success of the proposed simulator migration step towards a quantised discrete event architecture. Included in such a study would be the comparative analysis of the performance and error behaviour of all of these algorithms.

The error behaviour is of particular importance due to the fact that the GBAD System of systems model has become reliant on the 100Hz logical and discrete time management. Future effort is also required in re-validating, firstly, the individual enveloped models (models updated with QQIPs) and, secondly, the composed GBAD system of systems model.

REFERENCES

Baird, J. and Nel, J. J. (2005). 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.

Blackman, S. (1986). *Multiple Target Tracking with Radar Applications*. Artech House.

Duvenhage, A. (2007). A state estimation approach for live aircraft engagement in a C2 simulation environment. In *Proceedings of the 2007 Fall Simulation Interoperability Workshop*.

Duvenhage, B. (2008). Migrating to a real-time distributed parallel simulator architecture. Master's thesis, University of Pretoria.

Duvenhage, B. and Kourie, D. G. (2007). Migrating to a real-time distributed parallel simulator architecture. In *Proceedings of the 2007 Summer Computer Simulation Conference*, page 46.

Duvenhage, B. and le Roux, W. H. (2007). Peer-to-peer simulation architecture. In *Proceedings of the 2007, High Performance Computing & Simulation (HPCS'07) Conference*, pages 684–690.

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

le Roux, W. H. (2006). Implementing a low cost distributed architecture for real-time behavioural modelling and simulation. In *Proceedings of the 2006 European Simulation Interoperability Workshop*, pages 81–95.

Naidoo, S. and Nel, J. J. (2006). 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*, pages 551–558.

Oosthuizen, R. (2005). Doctrine development during systems acquisition and the importance of modelling and simulation. In *Proceedings of the 2005 European Air Defence Symposium*.

Zeigler, B. P., Kim, T. G., and Praehofer, H. (2000). *Theory of Modelling and Simulation, second edition*. Academic Press, San Diego, California, USA.

AUTHOR BIOGRAPHIES

ARNO DUVENHAGE is a Researcher for the Council for Scientific and Industrial Research (CSIR), South Africa. He joined the CSIR's Mathematical and Computational Modelling Research Group in January 2005 as a Software Engineer. Arno's current work involves modelling and simulation for acquisition decision support, focusing on air defense, specializing in distributed and networked systems. Arno has a BEng Degree in Computer Engineering from the University of Pretoria, South Africa, and is currently specializing in software engineering. His email address is aduvenhage@csir.co.za

BERNARDT DUVENHAGE is a Researcher at the Council for Scientific and Industrial Research (CSIR), South Africa. He joined the CSIR's Mathematical and Computational Modelling research group in 2004 as a Computer Scientist. He has since moved to the CSIR's Optronics Sensor Systems Simulation Research group and is currently involved in the modelling and development of the simulator architecture for a first principles physics based infrared and visual band synthetic scene simulator. Bernardt has a Bachelors of Science degree and Honours degree in Computer Science from the University of Pretoria. He is currently finishing off a Masters degree in Modelling and Simulation at the University of Pretoria's Computer Science Department. His email is bduvenhage@csir.co.za.