A Peer-to-Peer Simulation Architecture

Bernardt Duvenhage and Willem H. le Roux

Council for Scientific and Industrial Research
Pretoria, South Africa
Email: bduvenhage,whleroux@csir.co.za

Abstract— **A distributed parallel and soft real-time simulation architecture is presented. It employs a publish-subscribe communication framework layered on a peer-to-peer Transport Control Protocol-based message passing architecture. Mechanisms for efficient implementation and control of information flow between simulated entities form part of the architecture. A light-weight base simulation object model is also employed to provide maximum modularity and extensibility while keeping complexity manageable. The simulation architecture evolved over time to allow for the efficient implementation of a system of systems, virtual simulation. It has been successfully applied in an air defence simulation as a decision support tool and for standard operating procedure concept evaluation.**

KEYWORDS

Distributed, Parallel, Soft Real-Time, Simulation, Architectures, Peer-to-peer, Publish-Subscribe.

I. INTRODUCTION

In a decision support environment, system of systems level simulations are applied to provide end-users with the capabilities to identify, define, implement (virtual) and evaluate concepts that would otherwise be costly, time-consuming or impractical. Systems of systems level simulators typically involve multiple modelled entities with complex interactions and are executed in virtual or constructive simulation modes [1].

This paper presents a simulation architecture that evolved over time during decision support to an air defence procurement programme [2, 3]. Although there was no need for a generic, re-usable architecture – It was to be only used for a single simulation environment – it still had to provide standardised interfaces for efficient integration of models, services and other simulation-logistical functions. It should also support both constructive and virtual simulations, hence it should at least be soft real-time compatible when performing operator-in-the-loop simulations. Operators will mainly interact with the simulation via integrated mock-up consoles and not full immersive synthetic environments, which may be supported by integrated, external systems. Furthermore, it must allow both distributed and non-distributed simulation execution. The first is required to maintain soft real-time compliance when employing the virtual simulation mode if model processing loads are high. The latter is required for easier test and debugging as well as batch executions for statistical analyses. A conservative, discrete stepped time management mode forms an inherent part of the simulation architecture, as almost all of the models used in the air defence simulation environment are discrete time-stepped. To provide an efficient and effective decision support capability, specifically during system conceptualisation and field exercises, the architecture should allow for quick implementation and integration of new models. The same holds for the integration of external systems. Interoperability with other simulations is not an absolute requirement, but should not be excluded by design.

Several peer-to-peer architectures are reported in the literature, of which some are aimed at internet-based information sharing, discrete event simulations [4]–[7] or cooperative computing, such as solving processing intensive problems with *ad hoc* peer-to-peer networks [8]. Some architectures are also aimed at massively multi-player online role playing and other games [9, 10]. Giesecke [11] quantitatively investigated availability in peer-to-peer systems for prediction and identified basic characteristics to derive a formal model for describing architectures. Kotilainen [12] reports on an efficient peer-to-peer network simulator used to study artificial neural network algorithms.

Other simulation architectures or frameworks include the Aggregate Level Simulation Protocol (ALSP) [13], Distributed Interactive Systems (DIS) [14] and the High Level Architecture (HLA) [15]. The first two are seen as precursors to HLA. Although all three were developed for the defence community of the United State of America, HLA was intended to be adopted by the wider simulation community. The Standard Simulation Architecture [16] provides an additional framework to HLA to allow more flexibility, but be more cost-effective without paying performance penalties. The Open Simulation Architecture (OSA) [17] is a discrete event simulation architecture that promises integration of new and existing contributions at all levels. Hawley [18] proposes an object-oriented simulation architecture that separates the implementation of the dynamic system being modelled (application layer) from the simulation management functions (executive layer). The Extensible Modelling and Simulation Framework (XMSF) [19] aims to harness web-based technologies to promote interoperable simulations and provides mechanisms for systems to discover and use web services.

Of these architectures only HLA was evaluated since it is a fully fledged approach that covers all aspects of the simulation life cycle. However, in the South African defence environment it was not the optimal choice at the time. Although HLA promotes interoperability, the federation object model should still

Proceedings 21st European Conference on Modelling and Simulation
Ivan Zelinka, Zuzana Oplatková, Alessandra Orsoni ©ECMS 2007
ISBN 978-0-9553018-2-7 / ISBN 978-0-9553018-3-4 (CD)

be agreed or translated when two simulations are integrated. This was not always the case, therefore interoperability was not easily achievable [3].

The simulation architecture presented is not offered as an alternative for the above-mentioned architectures or frameworks, but rather to highlight the mechanisms used to implement an efficient architecture against the backdrop of system of systems simulation criteria. Efficiency in terms of implementation is required since a very small development team was used. In terms of simulation execution, soft real-time execution for multiple entities with update rates of 100Hz are used, requiring an architecture with low overheads.

II. System of Systems-level Simulation Architecture Needs

This section addresses the specific needs for a simulation architecture to meet the criteria as outlined in Section I and is discussed in the following subsections.

A. *System of Systems Simulation*

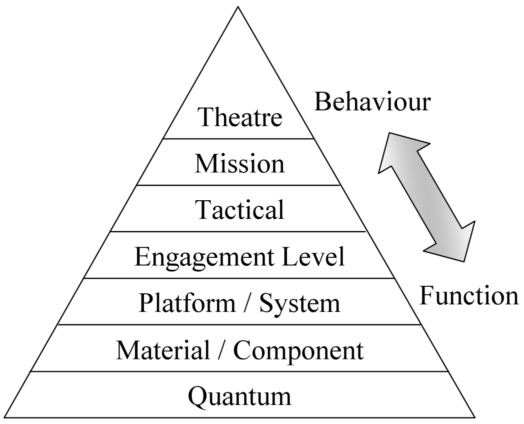

Fig. 1: The Systems Hierarchy applied to Modelling and Simulation (Adapted from [1, 20])

An adapted version of the systems hierarchy is shown in Figure 1 as applied to modelling and simulation [1, 20]. Systems of systems simulations are typically applicable at the engagement and tactical levels where entities are modelled at equipment and individual operator level. Individual entities are modelled, but forms "systems" when grouped organisationally. Tactical simulations also require one-on-one engagements, whereas higher order simulations require strategic actions of aggregated entities. For mission and theatre simulations entities are aggregated into units and generally referred to as wargaming simulations [1]. Lower system hierarchy simulations tend to focus more on the function of entities, whereas the higher levels on the behaviour of entities.

B. *Constructive and Virtual Simulation Mode Support*

Tactical level simulations (Figure 1) imply that not only equipment is simulated, but also the use of equipment, including standard operating procedures. This again implies that the human operators, human-equipment and human-human interactions be modelled, so that it becomes a constructive simulation. Simulation execution requirements for constructive simulations tend to be less stringent, but the faster a simulation executes, the more applicable it becomes as a what-if type analyses tool, as it allows a simulation user to test and evaluate scenarios quickly. Virtual simulations need to be at least soft real-time compliant to maintain realism [21].

C. *Distributed and Non-Distributed Simulation*

Non-distributed simulations are generally less complex to test and debug than distributed simulations, as simulation execution does not have to be traced across multiple processing nodes. However, non-distributed simulations may be soft real-time incompatible when model processing loads become too high for a single processing node.

Distributed simulations on the other hand require efficient inter-process communication frameworks, such that the inter-processing node communication overheads do not counter the advantage of extra processing nodes. Virtual simulations with multiple entities that are modelled at system of systems level, typically require distributed simulation to either provide faster than or real-time compatibility. The ideal simulation architecture would support both distributed and non-distributed simulation execution without having to alter the implementation, but only its configuration.

D. *Modularity and Extensibility*

Since the simulation architecture is used in a decision support environment, including the evaluation of system concepts, it should be efficient to add, maintain and upgrade models of equipment, operator terminals, external system interfaces and operators. In addition to the entities that participate in a synthetic environment, it should also be efficient to extend, maintain and upgrade the synthetic environment itself. Services such as inter-entity line of sight calculations should be inherently part of the synthetic environment. External system interfacing should be supported, but note that external systems may have requirements that cannot be met by the simulation architecture, such as hard real-time compatibility.

E. *Time-stepped Simulation*

Conservative time management is an integral part of the simulation architecture and is enforced by using a discrete time-stepped mechanism. Spatio-temporal properties play a pivotal role in any air defence system, therefore time-line accuracy is of importance in a simulation environment, and hence architecture.

Although models may internally use predictive event-based time management, their external interfaces should support conservative time management. This is necessary as both predictable and non-predictable events occur in an air defence simulation environment, of which the non-predictable events may violate causality, if non-conservative time management is used.

Most of the external systems that will be integrated, produce spatio-temporal data, be it in the form of positions of an aircraft from a flight simulator, or time-stamped detections from a sensor such as a radar. External systems that may be integrated includes equipment, data sources and simulators.

III. High-Level Simulation Architecture Design

In order to meet the needs as identified in Section II, a simulation architecture evolved from a single application to a fully distributed simulation architecture. After providing a short overview of the present architecture, each part is carefully explained in subsequent subsections.

The simulation architecture is based on an inter-process communication (IPC) framework using the Transfer Control Protocol (TCP). Processing nodes are fully connected in a peer-to-peer fashion and message-passing is managed via a publish-subscribe mechanism. Processes that need to communicate within the simulation architecture are:

- Models - Models of equipment, humans and operator consoles (interfaces).
- Services - Includes line-of-sight, terrain elevation and peripheral services such as data loggers.
- Consoles or Gateways - All external systems that need to be integrated with the simulation architecture is implemented via a gateway which in effect translates the protocol of the external system into the simulation object and spatial reference models of the simulation architecture. Mock-up operator consoles are also integrated via this mechanism.

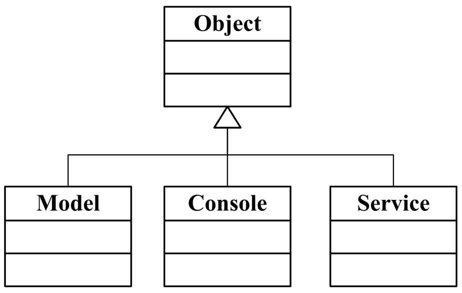

Fig. 2: Base Simulation Object Model

The above list of items is grouped under a base object to form the simulation object model of the architecture as indicated in Figure 2. An economical simulation object model (SOM) has been designed to curb implementation complexity.

A. Publish-Subscribe Object Communication Framework

All models, services and consoles (hereafter collectively referred to as objects) should be able to communicate efficiently in a distributed environment. Furthermore, it should be easy to establish and manage the communication channels between objects. A layered approach will reduce future migration effort to other IPC frameworks: keep the object implementation and communication logistics of an object separate. The object communication framework should also hide the underlying distributed implementation from the user of the framework and not dictate model fidelity or simulation granularity. The framework should allow distributed and non-distributed execution for easier test and debugging without requiring implementation changes of the framework or client software.

The publish-subscribe mechanism employed in the object communication framework is analogous to magazine subscriptions and also similar to object management in HLA [22]:

- Different publishers advertise their sets of titles available for subscription.
- Subscribers may subscribe to titles of their choice when they would like to.
- Publishers will then publish issues at regular intervals, which all subscribers will receive.
- All subscribers receive identical copies of an issue of a title.
- Publishers may also add titles at any given time to their collections. Cancellation of titles is not supported at present.
- A subscriber can read a copy of an issue as many times as they would like until a new one arrives.
- A subscriber may also elect to ignore old issues and only read the latest. Note that this is one of two supported modes, the second is an extension to the analogy (See next list).
- A publisher cannot change the content of an issue once it has been published and received by its subscribers.

The implementation of the publish-subscribe mechanism is somewhat extended over the analogy to allow more flexibility:

- Subscribers can select at what interval (rate) they want to receive issues. The maximum update rate is limited by the smallest time increment (frame) of the simulation.
- Subscribers can select if they want all issues of a given publisher, or just selected titles, without subscribing separately to each title.
- The subscriber may select what will happen with copies of issues that are received but not read. If kept, all copies from the oldest to the most recent have to be read to get to the latest issue. If not kept, the most recent copy is always available to be read. An in between mode is not supported.
- A specialised extension to support modelled communications between objects (modelled equipment or operators, not IPC related communications) is provided with additional parameters to support transmission functionality (status, delays, sender/receiver identification, etc.). Messages destined for transmission are passed immediately to the receiver where they are delayed in a cache to model the correct transmission delay, until delivery. The minimum transmission delay of a message is limited by the minimum time increment of the simulation. Messages are also rather delayed at the receiver than the transmitter, as the receiver knows, implicitly, its own position and, from the issue meta-data, the sender's position which are both required by the communications model.

Functionality as listed in the above two lists, are directly supported in the simulation architecture as part of the base object model and the simulation backbone, which is a set of classes and functions providing the necessary mechanisms.

B. Peer-to-peer Processing Node Architecture

A peer-to-peer processing node architecture was ultimately selected above the client-server architecture, as the server may form a bottleneck due to the double latency and bandwidth usage for messages transmitted from a client to the server and then to the receiving client from the server (Figure 3(a)).

To minimise traffic at a server, an intermediate layer of servers were considered before using the peer-to-peer architecture. The intermediate servers (Figure 3(b)) have less traffic to route, and will only transmit messages to other intermediate servers via the top-level server if a receiving client requires it. The scheme is efficient, but has one major drawback: The architecture is not domain independent, as the clustering of clients per intermediate server requires prior knowledge of clients that can be grouped by type or anticipated traffic.

Note that the peer-to-peer architecture will result in a single latency for messages passed between nodes (indicated as peers in Figure 3(c)), but IPC connections have to be brokered or configured in some way before a simulation execution starts. In the client-server case, all clients connect to the same server. The peer-to-peer architecture suffers from the same domain knowledge challenge as the intermediate server solution, i.e. which models may be grouped for acceptable execution performance. The ultimate architecture would allow for both the automatic distribution of models across processing nodes, as well as automatically introducing intermediate server layers for optimal execution performance. Each processing node executes a subset of all objects (models, consoles and services). In the case where a single processing node is used, all objects are executed on it. As conservative time management is used in a time-stepped fashion, the slowest or most processing intensive object governs the global execution performance of the simulation. Load balancing is therefore necessary and is supported either as a static configuration or with dynamic load management [23]. The latter requires passing of objects in-process between processing nodes during run-time.

The peer-to-peer architecture is fully connected, thus each node is connected to each other node at start-up using TCP. This results in $\frac{n(n-1)}{2}$ connections, where n is the number of nodes (peers). For the client-server case the number of connections equals the number of clients. Connections between objects are made using a proprietary, binary-packed protocol, irrespective if objects are on the same processing node or not, or if only one processing node is used.

C. TCP Implementation Details

Specific TCP implementation tweaks to ensure lower message latency between nodes are discussed in this subsection.

TCP messages are grouped per destination node and sent off together instead of sending each message separately. Message latency still turned out to be a problem due to TCP's Nagel algorithm [24]. The Nagel algorithm usually improves bandwidth (saves on message header overhead) by caching short messages for a certain time-out or until they are big enough to fill a complete data packet before sending the data. Typically message groups were much smaller than the normal packet size of 1.5 kilobyte. Turning off the Nagel algorithm gave the simulation architecture complete control over message sending times which decreased latencies considerably. The communication model would cause a node to send information to every other node once every simulation time increment which means that the shorter the latencies the faster the simulation can execute.

The TCP sending buffer was also made bigger than the default to allow the simulation architecture to push messages into the sending buffer without blocking to allow the simulation to continue processing while the TCP operating system thread continues sending. This approach saves the overhead of implementing the simulation's TCP sending code in a separate processing thread.

To start a distributed simulation, all the nodes except the first node may be started up in any order. As soon as the first node is started it makes connections to all the other nodes, which in turn make connections to each other and finally start the simulation.

IV. Results

Initial experiments with a non-distributed and intermediate server-client (see Section III) architecture showed that in order to execute large enough simulations, distributed processing would be required to maintain soft real-time compatibility [25]. Approximately 6-8 processing nodes were estimated for real-time compliance, but less could be used, as the model loading metrics were very conservative. Between 40 and 100 objects, with varying levels of fidelity were anticipated. With the actual architecture, a fully populated scenario translates to 177 entities that require processing. The architecture is still efficient enough to execute this at approximately soft real-time. Of the entities, 160 are models, 8 consoles and 9 services.

Soft real-time execution is maintained by synchronising with the local processing node clock. Processing time is yielded not to exceed real-time. However, this only works when the processing nodes are not overloaded, i.e. able to process all models within a simulation time frame, otherwise extra processing nodes may be added. If this still does not help, soft real-time compatibility cannot be maintained.

Distributed performance tests were done with a processing intensive test object that takes exactly 1ms PC time to increment and publishes a single title which is a text string of length 512 bytes. Each test object subscribes to the titles of all the other test objects, including its own title. The communication setup is thus fully connected over all objects. A 100Hz closed loop distributed simulation is run over one to six machines in as fast as possible mode. Simulation distribution and communication overhead results are presented in Table I. The simulation frames are 10ms in length, equating to a 100Hz update rate, giving ten 1ms slots for a maximum of ten models per node to sustain soft real-time execution. It can be seen that a single node is very efficient, running at

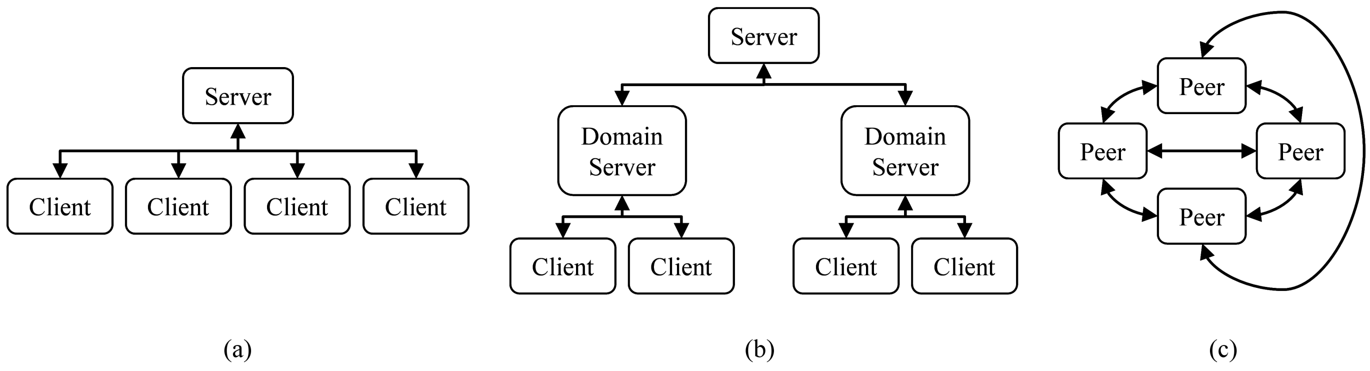

Fig. 3: Client-Server (a) Intermediate-Server (b) and Peer-to-Peer (c) Processing Node Architectures

99% of real time with 10 models which translates to a 1% communication overhead. Table I also shows that:

1) to run up to 9 test objects real time at least 1 node is required;
2) to run up to 18 test objects real time at least 2 nodes are required;
3) to run up to 24 test objects real time at least 3 nodes are required;
4) to run up to 32 test objects real time at least 4 nodes are required;
5) to run up to 35 test objects real time at least 5 nodes are required;
6) to run up to 42 test objects real time at least 6 nodes are required.

To run 42 test objects at real time at least 6 nodes are required running 7 objects each. This translates to an average communication overhead of just under 30%. Network usage was measured by using Windows XP's task manager network performance window. Note that the tests performed are worst case scenarios. All objects subscribe to all other objects and themselves (n^2 subscriptions) and the issue size is 512 bytes which is big enough for 21 double precision 3D coordinate triplets or a list of approximately 64 English words.

The simulation architecture has been successfully applied in an air defence simulation as a decision support tool and for standard operating procedure concept evaluation. It has been applied in extensions of the air defence simulation that include satellite-based optical and radar sensors for maritime surveillance concept development. It was also used for integration with various external systems, including situational air picture systems, operator console mock-ups, air traffic radars, flight simulators and similar simulations.

V. Future Work

A key challenge with soft real-time simulations is what should be done if the simulation slips on world-time? This often occurs, as models tend to have spurious high processing requirements. What techniques should be used to catch up again on world-time, specifically when a simulation is connected to external systems, such as a flight simulator or air picture systems? One solution is to use innovative schemes in console implementations to external systems to cater for data that arrives at the wrong-time, i.e. too late, or to too early. In the first case, prediction algorithms are necessary and in the latter buffering schemes.

Future work includes conducting comparative studies between peer-to-peer, intermediate server and client-server architectures. There is also ample opportunity for load balancing research: How to measure model loading per processing node efficiently and effectively, and load balancing algorithms.

VI. Conclusion

The ability to execute an entire simulation in an all-in-one mode on a single processing node (desktop computer) is a key advantage in how the simulation architecture is used. It is efficient and quick to configure scenarios for simulation, and to visually verify them using peripheral two and three dimensional viewers. Similar techniques are used to verify and validate newly integrated models, consoles or services. It is then merely a matter of changing a configuration to execute the simulation distributed to achieve soft real-time execution. The simulation architecture is suitable for parallel execution on a small to medium scale infra-structure.

The publish-subscribe architecture is very flexible in terms of objects and connection management. However, care should be taken to adhere to a standard way of implementing objects and not to abuse the flexibility.

The soft real-time performance figures obtained with worst-case object loadings indicate that the architecture is adequate for a small number of nodes with less than 10 objects per node. It is expected that network overheads will limit scalability to less than 100 objects in total, and therefore, the architecture is not considered to be scalable for large simulations (100's of objects), requiring soft real-time performance.

Author Biographies

BERNARDT DUVENHAGE has been with the CSIR within the Mathematical and Computational Modelling Research Group, that's part of the Defence, Peace, Safety and Security (DPSS) Research Group, since January 2004. Past responsibilities have included the development of a distributed simulation architecture, development of an optimized Line of

TABLE I: Distribution and Communication Overhead Results

Number of Nodes	Objects per Node	Issue Size	Percentage Real-Time	Network Utiliza-tion	Total Objects
1	7	512 bytes	142%	0%	7
2	7	512 bytes	132%	6%	14
3	7	512 bytes	126%	12%	21
4	7	512 bytes	117%	16%	28
5	7	512 bytes	110%	20%	35
6	7	512 bytes	102%	25%	42
1	8	512 bytes	124%	0%	8
2	8	512 bytes	116%	7%	16
3	8	512 bytes	110%	14%	24
4	8	512 bytes	100%	18%	32
5	8	512 bytes	92%	23%	40
6	8	512 bytes	85%	27%	48
1	9	512 bytes	110%	0%	9
2	9	512 bytes	104%	8%	18
3	9	512 bytes	99%	15%	27
4	9	512 bytes	90%	21%	36
5	9	512 bytes	81%	26%	45
6	9	512 bytes	75%	27%	54
1	10	512 bytes	99%	0%	10
2	10	512 bytes	93%	9%	20
3	10	512 bytes	88%	17%	30
4	10	512 bytes	77%	23%	40
5	10	512 bytes	70%	28%	50
6	10	512 bytes	60%	32%	60

Sight (LOS) algorithm and LOS service, development of a terrain attitude and altitude service, development of models in cooperation with OEMs and development of a 3D analysis tool based on the Open source Scene Graph (OSG). He completed his BSc(hons) in Computer Science in 2005 and is currently pursuing an MSc in Computer Science on real time simulation architectures.

HERMAN LE ROUX has been with the South African Council for Scientific and Industrial Research since April 1998 and is at present a Principal Engineer in the Mathematical and Computational Modelling Research Group. He is involved in Modelling and Simulation-based Acquisition Decision Support, specifically for the South African National Defence Force. Interests include information fusion, biometrics, artificial intelligence and software engineering. Le Roux completed a Masters Degree in Computer Engineering at the University of Pretoria in 1999 and is currently pursuing a PhD in Information Fusion.

ACKNOWLEDGEMENTS

The authors would like to thank Dr Jackie Phahlamohlaka, Cobus Nel, Anita Louis and Arno Duvenhage, all colleagues at the CSIR, for their valuable inputs, as well as the Armaments Corporation of South Africa for supporting this work.

REFERENCES

[1] DOD, "Department of Defense: Modelling and Simulation Master Plan," Under Secratary of Defense for Acquisition and Technology, DoD 5000.59-P, Alexandria, VA, October 1995.

[2] J. J. Nel and H. J. Baird, "The evolution of M&S as part of smart acquisition using the SANDF GBADS programme as example," in *Twelfth European Air Defence Symposium*, Shrivenham, June 2005.

[3] W. 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*. Stockholm: Simulation Interoperability Standards Organization, June 2006.

[4] S. Naicken, A. Basu, B. Livingston, S. Rodhetbhai, and I. Wakeman, "Towards yet another peer-to-peer simulator," FOURTH INTERNATIONAL WORKING CONFERENCE PERFORMANCE MODELLING AND EVALUATION OF HETEROGENEOUS NETWORKS, West Yorkshire, September 2006.

[5] A. Montresor, G. D. Caro, and P. E. Heegaard, "Architecture of the simulation environment," Information Society Technologies, Italy, Project Report IST-2001-38923, January 2004.

[6] S. Cheon, C. Seo, S. Park, and B. P. Zeigler, "Design and implementation of distributed DEVS simulation ina peer to peer network system," in *Proceedings of the 2004 Military, Government and Aerospace Simulation Symposium*, Virginia, April 2004.

[7] B. Gedik and L. Liu, "A scalable peer-to-peer architecture for distributed information monitoring applications," *IEEE Transactions on Computers*, vol. 54, no. 6, pp. 767–783, June 2006.

[8] G. D. Costa, "Peer-to-peer simulation," Presentation at the Federal University of the Rio Grande Do Sul, Porto Alegre, 2002.

[9] Y. Kawahara, H. Morikawa, and T. Aoyama, "A peer-to-peer message exchange scheme for large scale networked virtual environments," Proceedings of the 8th IEEE International Conference on Communications Systems (ICCS 2002), Amsterdam, April 2002.

[10] S.-P. A. van Houten and P. H. M. Jacobs, "An architecture for distributed simulation games," in *Proceedings of the 2004 Winter Simulation Conference*, R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, Eds., Washington DC, December 2004.

[11] S. Giesecke, T. Warns, and W. Hasselbring, "Availability simulation of peer-to-peer architectural styles," in *Proceedings of the 2005 workshop on Architecting dependable systems (WADS 2005)*. Waterloo: ACM Press, 2005, pp. 1–6.

[12] N. Kotilainen, M. Vapa, T. Keltanen, A. Auvinen, and J. Vuori, "P2prealm peer-to-peer network simulator," in *Proceedings of the 11th Intenational Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks*, June 2006, pp. 93–99.

[13] R. Weatherly, D. Seidel, and J. Weissman, "Aggregate Level Simulation Protocol," Presented at the 1991 Summer Computer Simulation Conference, Baltimore, July 1991.

[14] "IEEE standard for information technology - protocols for distributed interactive simulations applications," IEEE Std 1278-1993, 1993.

[15] F. Kuhl, R. Weatherly, and J. Dahmann, *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*, ser. Artificial Intelligence. Upper Saddle River: Prentice Hall, 1999.

[16] J. S. Steinman and D. R. Hardy, "Evolution of the standard simulation architecture," http://www.modelingandsimulation.org/issue10/SISO/steinman.html, vol. 3, nr. 2 , issue 10, April-June 2004, Accessed 16 January 2007.

[17] O. Dalle, "OSA: an open Component-based architecture for Discrete-event Simulation," INRIA, Tech. Rep. RR-5762, February 2006, available from http://www.inria.fr/rrrt/rr-5762.html.

[18] P. A. Hawley and T. J. Urban, "An object-oriented simulation architecture," AIAA Modeling and Simulation Technologies Conference and Exhibit, Providence, August 2004.

[19] D. Brutzman, M. Zyda, J. M. Pullen, and K. L. Morse, "Extensible modeling and simulation framework (XMSF) challenges for web-based modeling and simulation," TECHNICAL CHALLENGES WORKSHOP, STRATEGIC OPPORTUNITIES SYMPOSIUM, www.movesinstitute.org/xmsf, San Diego, October 2002.

[20] J. H. S. Roodt, L. Berglund, and P. Klum, Worksession between FOI and CSIR, Linköping, 2001.

[21] M. R. Macedonia, M. J. Zyda, D. R. Pratt, P. T. Barham, and S. Zeswitz, "NPSNET: A network software architecture for large scale virtual environments," *Presence*, vol. 3, no. 4, 1994.

[22] "High Level Architecture interface specification: Version 1.3," U.S. Department of Defense, April 1998.

[23] B. Duvenhage, "The contribution of static and dynamic load balancing in a real-time distributed air defence simulation," 2007, to be submitted to the Summer Computer Simulation Conference, San Diego.

[24] B. A. Forouzan, *TCP/IP Protocol Suite*, 2nd ed., ser. Forouzan Networking. Boston: McGraw-Hill, 2003.

[25] B. Duvenhage and W. H. le Roux, "MobADS: TCP-based distributed simulation architecture investigation," Council for Scientific and Industrial Research, Tech. Rep. DEFT-MSADS-00151, July 2004.