Cybernetics and Systems: An International Journal, 42:502–525

ISSN: 0196-9722 print/1087-6553 online
DOI: 10.1080/01969722.2011.610270

Taylor & Francis
Taylor & Francis Group

Large-Scale High-Fidelity Agent-Based Simulation in Air Traffic Domain

PŘEMYSL VOLF, DAVID ŠIŠLÁK, and MICHAL PĚCHOUČEK
Agent Technology Center, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University, Prague, Czech Republic

We present the concept of dynamic partitioning of scalable, high-fidelity multi-agent simulation complemented with intelligent load-balancing processes. The simulation framework is designed to simulate entities to high details that require extended computation resources. To be able to simulate a huge amount of entities, distributed simulation is introduced using spatial partitioning and dynamic load balancing. A novel and important feature is the combination of the synchronous and asynchronous parts in the simulation. We use the domain of the air traffic simulation to verify the simulation framework. We present a method to perform spatial and temporal planning within 3D space and multilayer architecture using several collision avoidance algorithms to illustrate the high computational demands of each airplane. The platform has been used to support simulation of an entire civilian air traffic touching the national airspace of United States. A thorough evaluation of the system has been performed, confirming that it

The work has been supported by the Federal Aviation Administration (FAA) under project number DTFACT-08-C-00033 and by the Czech Ministry of Education under grant number 6840770038. The underlying AGENTFLY system was supported by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-06-1-3073. The views and conclusions contained herein are those of the author and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of the Federal Aviation Administration, the Air Force Office of Scientific Research, or the U.S. Government.

Address correspondence to Přemysl Volf, Agent Technology Center, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Technickaz, 16627 Prague 6, Czech Republic. E-mail: volf@labe.felk.cvut.cz

can scale up to a very high number of complex agents operating simultaneously (thousands of aircrafts) with full detailed models.

KEYWORDS agent-based simulation, air traffic, collision avoidance, large-scale simulation, trajectory planning

INTRODUCTION

Agent-based simulations have recently become a popular way to model and study complex multi-actor systems, complementing the long-established system dynamics and discrete-event simulation approaches (Borshchev and Filippov 2004). For a large number of situated entities, the high complexity of their reasoning and/or high complexity of entities–environment interaction, a single machine might not be able to host the whole simulation. In such a case, an effective distribution schema is required that would enable the simulation to scale.

In this article, we propose a particular distributed agent-based simulation architecture. The development of the platform has been motivated by the need to simulate, with a great level of detail, full civilian air traffic touching[1] the national airspace of the United States. The design and the implementation developed, however, provide a more universal recipe and can be applied to other large-scale (analytical) simulations requiring high simulation fidelity, scalability, and maximum simulation speed.

The proposed approach is based on dynamic partitioning of the simulation of the virtual world into a variable number of dynamically sized partitions, each handled by a dedicated environment simulation component. Thanks to the prevailing locality of entity-to-entity and entity-to-environment interactions, such partitioning allows effectively distributing the required computation between a number of machines while keeping the communication overhead low. The maximum simulation speed together with realistic behavior of the entities is achieved using a combination of the synchronous and asynchronous parts in the simulation.

The proposed architecture has been applied for the simulation of the full civilian air traffic touching the National airspace of the United States, requiring high-fidelity simulation involving a total of 52,799 flights with up to 6,500 simultaneous aircrafts operating over the whole Earth—long flights departing, arriving, or flying over the United States.

The air traffic simulation consists of two basic parts: the virtual environment and airplanes. The virtual environment contains global model data (wind, special use airspaces) and physical simulation of airplanes' bodies,

[1]Air traffic touching national airspace of the United States consists of all airplanes flying at least part of their trajectories inside national airspace.

which requires massive continuous computation of airplanes states—position, direction, fuel consumption, weight, etc. Simulation time of airplanes' reasoning is spent on planning their flight trajectory as well as collision detection and resolution that is highly non uniform in time. Thus, an effective distribution and load balancing are the key features of the simulation of the air traffic domain or any other virtual world based on moving entities.

PROBLEM SPECIFICATION

The proposed architecture is used to simulate large-scale civilian air traffic as well to simulate unmanned aerial vehicles (UAVs) and implements the concept of the free flight (Schulz et al. 1997; Hill et al. 2005). The simulation consists of two parts: simulation of the virtual world and simulation of airplanes.

The virtual world has to be evaluated synchronously in order to provide consistent evolution of the virtual world. This means that all states are updated each simulation tick. After the last update the states are valid until the next update throughout the simulation. Computation of these states is computationally demanding according to the complexity of the virtual world. We describe this environment in the section on the U.S. national airspace traffic domain.

The simulation of the airplanes has to be asynchronous because the computations aboard real airplanes (UAVs) are not synchronized to the real world. The airplane in the system is represented as a fully autonomous airplane with the capability of trajectory planning and collision avoidance. Most of the time, the airplane follows the flight plan. The flight plan is initially created using the trajectory planner based on the navigation waypoints. In the case of a collision situation, the fight plan is updated using the collision avoidance maneuver. Both trajectory planning and collision avoidance are computationally very complex. This is illustrated in the sections on trajectory planning and collision avoidance. The application of this concept leads to a high and non uniform computational load to the simulation system and makes it hard to scale up.

Simulation Approach

There are two basic approaches to agent-based simulations: analytic simulation and distributed virtual environments. The analytic simulation is typically used for high-fidelity simulation of a given problem with a maximum detail and accuracy to provide results for further analysis of the problem. Analytic simulation runs typically without human interaction and thus a repeated simulation run with the same initial configuration leads to the same results.

Furthermore, the relation to real-world time (referred to also as *wall clock*) is not important in analytic simulation and the simulation is therefore executed at maximum speed to deliver the results as soon as possible (this mode is also referred to as *as-fast-as-possible mode*; Fujimoto 1999). The simulation is synchronous and usually deterministic and can be based on either time steps or events. The time-stepped simulation advances by predefined equally sized (virtual simulation) time steps (Guo et al. 2000; Wu and Gong 2001), and new states of the simulation are computed after each such a step. Event-driven simulation uses events as a basis for its execution (Banks et al. 2001; Nicol and Yan 2004). Each event is scheduled with a time stamp and the simulation framework advances to the time of the event with the earliest time stamp and determines the new states.

Distributed virtual environments (DVEs; Waters and Barrus 1997) are used to create a realistic illusion of real-world environment to a human user for purposes such as training or entertainment. Because of the involvement of human users, DVEs are usually paced with wall clocks and do not produce the same results when repeated. They also usually do not support absolute synchronization and ordering of messages (Lui and Chan 2002; Zhou et al. 2004; Morilla et al. 2006).

High-level architecture (HLA; Dahmann et al. 1997; Kuhl et al. 1999) is an industry standard developed as an extension of DVE simulation. The architecture has been defined by a working group formed by the Defense Modeling and Simulation Office in order to standardize simulation development and allow interoperability between various simulators produced by multiple vendors by using standardized simulation roles, predefined communication protocols, and the data exchanging format.

The approach described in this article uses a combination of both the analytic and DVE approach. Agent's communication, internal processes, and nonphysical interaction with other agents is asynchronous and nondeterministic, possibly containing a human in the loop, and the implementation of these processes therefore uses the DVE approach. The simulation of the environment, situated entities, and physical interactions in the virtual world is synchronous and deterministic, and the analytic approach with maximal accuracy is therefore used. The simulations built on the platform can run in the as-fast-as-possible mode, preserving the accuracy of the simulation together with asynchronous and nondeterministic behavior of the agents in faster than wall clocks–paced time.

As far as our primary application area of interest—that is, air traffic simulation—is concerned, both the analytic and DVE approaches are used. DVE is primarily used for training purposes and not directly relevant to our objectives. Analytic simulation, on the other hand, is used for the purposes of obtaining insight into various processes and phenomena related to air traffic. Numerous simulations with varying levels of detail and different purposes and coverage have been developed. Yamamoto et al. (2008) presented

the ZASE (Zillions of Agents-based Simulation Environment) platform for simulation up to millions of agents. The platform supports only static load balancing and does not support a virtual world with interaction between the physical bodies of the agents and environment. Jamali and Zhao (2008) presented an approach for balancing computation resources in wide-area networks. Agogino and Tumer (2008) used agent-based simulation for incremental enhancements of the current air traffic management (ATM). Krozel and Doble (2007) used the FACET (Future ATM Concepts Evaluation Tool) simulation system to model air traffic in inclement weather. Gorodetsky et al. (2008) used an agent-based simulation for ATM within the airspace of large airports. Hwang et al. (2007) used simulation for verification of collision avoidance algorithms. All of the above systems focused on simulating a particular subset of the overall air traffic. On the other hand, the presented approach aims to provide a model of the entire air traffic. The huge computational requirements resulting from this goal were consequently one of the driving forces behind the development of the presented distributed simulation platform.

We use A-globe middleware for interagent communication based on the survey (Weyns et al. 2005). A-globe employs an approach similar to the JADE (Java Agent DEvelopment Framework) platform. For agents it uses peer-to-peer communication (which can be implemented as a multicast too), whereas the simulation environment can use bus-like communication (components join groups and data are delivered to the entire group). A-globe supports both asynchronous real-time and synchronous communication.

U.S. National Airspace Traffic Domain

The proposed simulation approach is used for the simulation of civilian air traffic within the U.S. national airspace (U.S. NAS). The resulting airspace simulator is used to study concepts for the Next Generation Air Transportation Systems (NGATS; National Research Council Panel on Human Factors in Air Traffic Control Automation 1998) in cooperation with the Federal Aviation Administration (FAA).

Each civilian flight operated by an airplane under instrumented flight rules (IFR; Nolan 2004) is represented as an entity within the simulation. One simulation run typically covers one day and simulates operation of all civilian IFR flights that traverse, fully or partially, U.S. NAS. Even though the area of the study is limited to U.S. NAS, some effects external to U.S. NAS are also studied, covering almost the whole Earth for long-distance flights. The overall number of simulated airplanes is 52,799 (up to 6,500 are simultaneous) corresponding to real air traffic for a particular day in 2007.

The complexity of the simulated virtual world and entities increases with the external data used that influence all parts of the simulation. The

main factors influencing the computational complexity of the simulation are as follows:

- Usage of the Global Positioning System (GPS) coordinate system. The GPS coordinate system significantly increases the computational complexity of the trajectory planning and computation of airplanes' positions/states.
- Usage of the BADA (Base of Aircraft DAta) airplane performance model (Nuic et al. 2005). The BADA model specifies a precise flight model for each airplane type. The model allows computation of thrust, drag, and lift forces during different phases of the flight. This allows computation of actual and total fuel consumption for each flight.
- Usage of weather/wind data; these data are used for computation of the airplanes's positions/states and increase the complexity of the trajectory planning if used as a part of the optimization criterion.
- Usage of special-use airspaces (SUAs). SUAs define airspace where it is restricted to fly. SUAs are dynamic—they can have a specified time interval when they are applied and when not. This increases the overall complexity of trajectory planning.

The distribution of the number of airplanes during the simulation day is depicted in Figure 1. Note that there are significant peaks every half hour of the simulation time; these peaks lead to extreme loads on the system with more than 300 airplanes created at one simulation cycle in the highest peak of the day. For such newly created airplanes, the computation-intensive flight trajectory planning and flight control algorithms are executed.

The simulation cycle is 12 s long, which is suitable for this domain. A scalable simulation platform is required because the system is used for

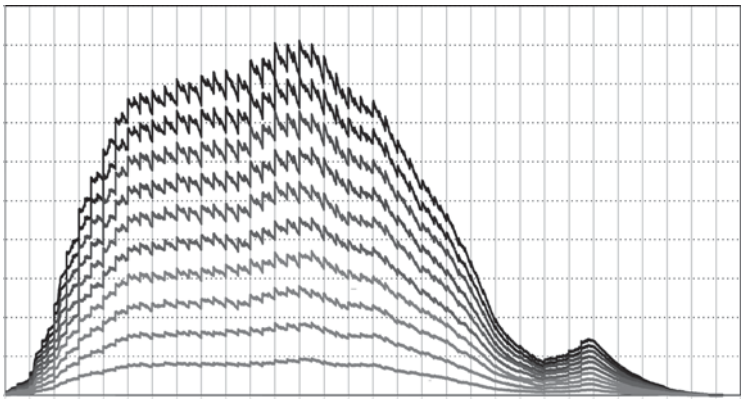

FIGURE 1 Number of simultaneous airplanes during the simulation.

analysis of concepts related to the predicted growth of air traffic in the upcoming decades (i.e., a simulation of the expected density of traffic in 5, 10, 15, and 20 years). The Boeing Company (2008) forecasted that the number of cargo flights will triple over the next 20 years.

Trajectory Planning

The AA* algorithm is introduced to show the computation complexity of the flight trajectory planning and thus its direct influence on the distributed simulation. Path planning causes the simulation to become very nonuniform in two ways. First, the path planning is run in a single moment, consuming a lot of resources, and is idle for the most of the time. Second, a lot of airplanes are planning a trajectory in some sectors but not in others. More detailed information can be found in Šišlak et al. (2009a, 2009b).

The AA* algorithm extends the original A* algorithm to be usable in large-scale environments without neglecting the search precision. AA* removes the trade-off between speed and precision by introducing adaptive sampling. During the expansion, child states are generated by applying vehicle elementary motion actions using elements' adaptive parameterization. A set of elementary motion actions is defined by the model of the nonholonomic vehicle movement dynamics. The adaptive parameterization varies and thus the algorithm makes larger steps when the current state is far from obstacles and restricted areas and smaller steps when it is closer. Figure 2 shows the advantage of the adaptive sampling of the AA*. The adaptive sampling can be seen as different step sizes (distance of the arc in the figure) depending on the distance to the obstacle.

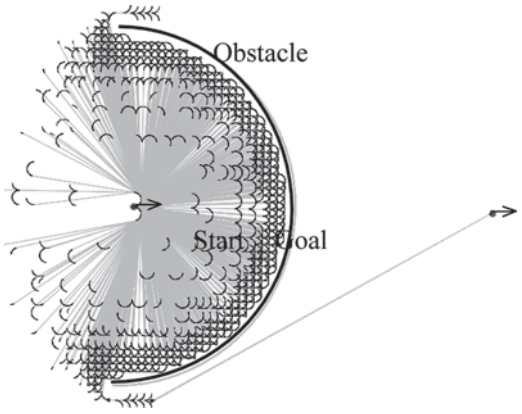

FIGURE 2 Adaptive sampling example in the two-dimensional setup. The adaptive sampling can be seen as different step sizes (distance of the arc) depending on the distance to the obstacle.

There is a defined search precision specifying the minimal sampling grid step that is used in the areas closest to obstacles. The search precision is defined so that the AA* algorithm does not skip any existing gap between obstacles larger than this precision. Specifically, the AA* algorithm uses the highest possible parameterization that ensures that the distance to the closest obstacle is not smaller than the distance corresponding to two respective sampling steps.

The adaptive sampling in the AA* algorithm requires a different definition of identity tests when working with open and close lists. The original equality implementation is replaced by a similarity check. Two states are similar if their Euclidean distances and their direction vector variations are less than a threshold derived from the respective sampling parameterization. Otherwise, the adaptive sampling of a nonholonomic vehicle trajectory causes an infinite state generation in the continuous space. To remove the effects of varying sampling, each path candidate generated during the search is smoothed.

Properties of the AA* concept were evaluated on a set of two- and three-dimensional setups. The original A* algorithm with a distance-to-target heuristics was chosen as a comparator because it is the only one that provides an optimal solution and does not require any preprocessing of the environment definition. The AA* algorithm provides acceleration of the path planning up to 1,400 times faster in comparison with the original A* algorithm. Moreover, it was found that the AA* algorithm accelerates the result in case of failure (the path does not exist) due to the reduced number of all generated states.

The computational complexity of the algorithm increases according to the data used—GPS coordinate system, BADA model, wind optimization, and special-use airspaces.

Collision Avoidance

The collision avoidance component is shortly introduced to show its influence on the distributed simulation. Collision avoidance causes the greatest changes in the uniformity of the simulation. Each collision avoidance causes repeated calls to the trajectory planner on several airplanes flying in the same area. That causes a rapid increase in computation load in the respective area. Solving several collision situations in the same simulation partition leads to a slowdown of the whole system. Thus, it is important to divide the areas with a high density of air traffic into more simulation partitions. More detailed information can be found in Šišlak et al. (2007, 2008) and Volf et al. (2007).

The multilayer collision avoidance (MLCA) component is capable of solving future collisions by means of a combination of different avoidance methods. There is no central planner providing a collision-free flight plan;

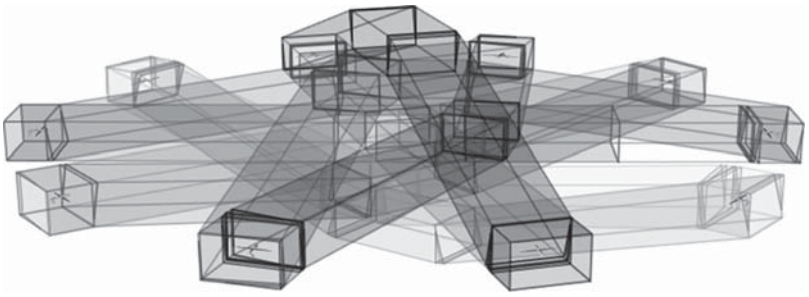

FIGURE 3 A 3D solution after several IPPCA iterations for the super conflict setup of 10 airplanes. The IPPCA algorithm is used in the experiments for collision avoidance. It deploys multiagent negotiations aimed at finding the Pareto-optimal CA maneuver. Software agents hosted by each asset generate a set of viable CA maneuvers (by means of the planning mechanism described earlier) and compute the costs associated with each maneuver (based on, for example, the flight plans total length, time deviations in mission waypoints, altitude changes, curvature, flight priority, fuel status, possible damage, and type of load).

hence, the individual plans are provided by the planning agents.[2] Based on priority, MLCA selects an appropriate collision solver for a specific possible future collision. Sophisticated switching of the collision avoidance algorithms is inevitable because the algorithms have different properties. Different algorithms provide different qualities of the collision avoidance solution, and they require different amounts of time to find such a solution. Specifically, the negotiation-oriented algorithms (cooperative) may provide better collision avoidance solutions than noncooperative algorithms, though they may be more time consuming (due to multiparty interactions).

There are several collision avoidance algorithms implemented in the current system. The cooperative rule-based collision avoidance (RBCA) algorithm is a domain-dependent algorithm based on the FAA's visual flight rules. The iterative peer-to-peer collision avoidance (IPPCA) algorithm is another example of cooperative algorithm based on agent communication and negotiation (see Figure 3).

[2]The proposed modular architecture is domain independent. Therefore, it is ready for deployment on autonomous vehicles like airplanes or ground vehicles (VFRs; Visual flight rules). The implementation in AgentFly is used as a reference mechanism to test the efficiency of the following three CA algorithms. The iterative peer-to-peer collision avoidance (IPPCA) algorithm employs pair-wise negotiation to solve the collision. The multiparty collision avoidance (MPCA) algorithm extends the first two algorithms by allowing several assets to negotiate over their collective CA maneuvers. The noncooperative collision avoidance (NCCA) algorithm supports CA when communication between assets is impossible. This class of algorithms is based on modeling and predicting the noncooperative object's future airspace occupancy and representing its possible future positions in terms of dynamic no-flight zones.

DYNAMIC PARTITIONING OF DISTRIBUTED SIMULATION

Distributed simulation is described as a common architecture usable for a different domains. The simulation involves large numbers of situated entities[3] (aircrafts) operating and interacting in a realistically modeled large-scale Earth-like 3D environment limited by a maximum altitude virtual world. The entities are not present during the whole simulation but are dynamically introduced or removed based on the simulation requirements.

Each situated entity in the simulation carries a state, components of which can be either (1) observable—that is, public and external—or (2) hidden—that is, nonobservable, private, and internal. The fundamental component of the entity's observable state present in the simulation is its position and orientation in the virtual world.

The evolution of the entity's state is governed by the defined entity's physics. Physics can be divided to intra-entity and inter-entity. Intra-entity physics captures those aspects of physical dynamics that can be fully ascribed to a single entity. Although the equations of the intra-entity physics can refer to any states in the simulated world, they only govern the states carried by its respective entity. In contrast, the inter-entity physics captures the dynamics that affect multiple entities simultaneously and that cannot be fully decomposed between them (consider, e.g., the effects of a physical collision of two entities).

In addition to physical interactions, autonomous entities can interact via communication—that is, exchange electronic messages—and by using sensors to perceive their surrounding virtual environment.

Having defined the type of simulations addressed and the necessary terminology, we can now state the desired properties of a simulation platform:

- High fidelity: The simulation has to be accurate with respect to the defined physics. Simulation level of details (LOD; Brom et al. 2007) techniques cannot be used to reduce the complexity of used intra- or inter-physics because this could lead to significant errors in the simulation output.
- Scalability: The simulation platform has to be scalable in terms of the number of simulated situated entities (both the number of simultaneously running entities and their total number) and the virtual world dimension. Simulation architecture should allow simulation scale-up simply by increasing the number of computational resources used for the simulation.
- Maximum simulation speed: The high-performance simulation platform should be able to produce results for a given simulation scenario in the minimum possible time without sacrificing the accuracy of the high-fidelity

[3]By *situated entities* we mean that they are embedded in a synthetic model of a 3D physical space.

simulation. In other words, the speed with which the simulation is executed should not affect simulation results.

Any situated entity in the virtual world consists of up to three subcomponents:

- Body: The entity's body encapsulates the entity's intra-physics that governs the evolution of entity's state as a function of other, possibly external, states. Typically, the body defines motion dynamics for the entity.
- Reactive control: The entity's reactive control contains real-time loop-back control for the entity. The loop-back control affects the entity's states based on the observation of other states. Reactive control—for example, handling urgent reactive behaviors (such as avoiding an obstacle)—is triggered in emergency situations.
- Deliberative control: The entity's deliberative control contains complex algorithms employing planning, prediction, and communication with other entities. The output of deliberative control are typically fed back into the entity's reactive control module and provides new control for the entity. Typically, complex deliberative algorithms providing high-level entity control are invoked based on the sensing perceptions. The deliberative control belongs to the asynchronous part of the simulation.

Due to their principal similarity in the loop-back approach, the body and the reactive control are collectively referred to as the entity's *state updater*. The state updater belongs to the synchronous part of the simulation.

Distributed Simulation Architecture

Components presented in this section are implemented as software agents in a multi-agent middleware A-globe (Šišlak et al. 2005) supporting the architecture with effective agent-to-agent communication, Yellow Pages services, agent full migration, and agent life cycle management. All components in the simulator architecture are divided into two groups; see Figure 4:

- Environment simulation component agents (ESC agents): These agents are responsible for application of all entities' state updaters and for the updates resulting from the inter-physics. The ESC agent computes the synchronous part of the simulation and is designed to run at one processor core and use its whole capacity. Therefore, a single ESC agent for one partition represents all state updaters assigned to this partition.
- Deliberative control agents: The deliberative control part of each situated entity is encapsulated in a separate deliberative control agent. Deliberative

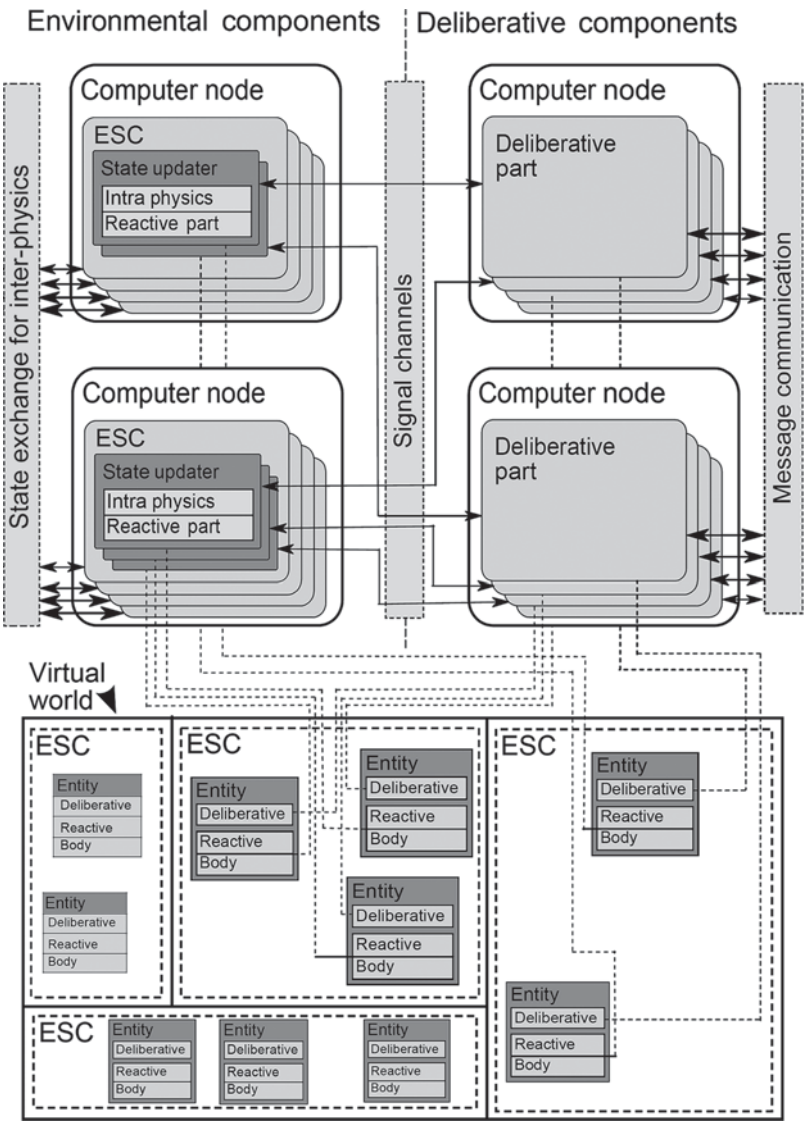

FIGURE 4 Architecture overview of the distributed simulation approach.

parts can communicate via agent-to-agent message transmission if this is allowed by inter-physics governing the communication medium.

Even though the deliberative control part and the state updater is decoupled in the simulator architecture, the deliberative control and the state updater of each entity in the simulation are connected by a signal channel through which sensing perceptions (one way) and control commands to the reactive control (the other way) are transmitted.

The presented simulation platform employs a time-stepped simulation approach (Fujimoto 1999). The virtual simulation time driving the dynamic processes of the simulation entities is incremented uniformly by a constant time step in each simulation cycle. Defined physics and loop-back control from the reactive parts are expressed in terms of difference equations; such equations define the entity's state (after the time increment) as a function of the current state only. All of these difference equations, and therefore all physics and reactive parts, have to be evaluated synchronously in order to provide consistent evolution of the virtual world. Synchronous evaluation means that states are updated only after difference equations (relevant in the given moment in time) have been evaluated.

The above virtual time steps and thus the resulting state updates can be applied regularly with respect to the external wall clock time or in an as-fast-as-possible manner. In the as-fast-as-possible mode, the next time step is initiated immediately after the previous one has been completed. This mode therefore allows completion of a given simulation scenario in the shortest possible time.

Environment Partitioning and Load Balancing

Both the environment simulation and deliberative control agents can be distributed over multiple computer nodes during the simulation. The whole virtual simulation world is spatially divided into a number of partitions. The partitions are mutually disjoint except for small overlapping areas around partitions' boundaries. Each partition has a uniquely assigned environment simulation agent (ESC agent) that is responsible for updating states corresponding to all entities located within its assigned partition. Figure 5 presents an example of airspace partitioning from the air traffic simulation domain.

In contrast to the state updaters, the entity's deliberative control parts are deployed to the computer nodes irrespective of the partitions and the corresponding entity's position in the virtual world.

The above virtual world partitioning implies that whenever the location of an entity changes in such a way that the entity moves (spatially) to a new partition, the entity's state updater needs to be migrated to the ESC agent responsible for the new partition. In contrast, the relatively heavy-weight deliberative control parts are never moved between computer nodes.

The signal channel between the entity's reactive control and its deliberative control modules is transparently reconnected and it is guaranteed that no signal transmission is lost during the migration of a state updater between two ESC agents. The small overlapping areas around partition boundaries introduced above are used to suppress oscillatory migration of state updaters between neighboring ESC agents in the case when the corresponding situated entity is moving very close along partition borders.

FIGURE 5 Airplanes over U.S. NAS and world partitioning.

The consistent evolution of the whole virtual world is achieved by coordinated operation of all ESC agents. Each simulation cycle is decomposed into the following phases:

1. Application of deliberative controllers' changes: Deliberative parts control situated entities by modification of loop-back rules within corresponding reactive parts. All modifications received in the previous simulation cycle take effect in this phase.
2. Evaluation of state updaters: During this phase, ESC agents determine new state values as a result of difference equations defined in intra-physics and reactive control modules of respective entities.
3. Evaluation of inter-physics: In this phase, ESC agents exchange state information required to calculate state updates controlled by inter-physics (and therefore possibly crossing partition boundaries).
4. Migration of state updaters: Entities whose location after the state update no longer belongs to the partition corresponding to their hosting ESC agent are migrated to the appropriate ESC agent based on the relevance to its partitions.
5. Application of new states: From this moment, all precomputed new state values are considered current ones.

The next phase of the simulation cycle is invoked only once the current phase is completed by all ESC agents.

In order to provide maximum performance throughout the whole simulation, the proposed architecture implements load-balancing mechanisms. Load balancing is applied to both environment simulation agents (through dynamic repartitioning) and deliberative control agents (through balanced start-up deployment).

The virtual world decomposition to partitions is not fixed during the simulation but is dynamically updated depending on the load of computer nodes hosting environment simulation agents. Based on the time required for processing of the phases 2 and 3 of the simulation cycle by each ESC agent, the world is repartitioned so that the number of entities belonging to partitions is proportional to the measured times. The ESC agent performing phases 2 and 3 faster will be assigned a larger area with more situated entities and thus more state updaters and vice versa. If there are no situated entities in the simulation, partitions are equally distributed to cover the whole virtual world.

The above-described repartitioning is performed before the migration of state updaters (phase 4 of the simulation cycle). After repartitioning, all state updaters that no longer belong to their currently assigned ESC agent partition are subsequently migrated to the appropriate ESC agent. Repartitioning is not performed during each simulation cycle but only if the difference in state update evaluation times by all ESC agents exceeds a predefined threshold. The repartitioning is also triggered whenever a new computer node is allocated to the environment simulation part.

The load of computer nodes running deliberative control agents is balanced only when new deliberative agents are instantiated. Deliberative parts of newly introduced entities are spread proportionally among computer nodes according to individual nodes' average loads in several previous simulation cycles, expressed as the sum of each node's idle time over those simulation cycles. The computer node with the highest idle time will receive more deliberative control agents and vice versa. There is no dynamic rebalancing of already running deliberative control agents because it would introduce issues with agent-to-agent addressing and proper conversation protocol consistency during the migration.

EVALUATION

We have evaluated the proposed distributed simulation architecture on a large-scale simulation of civilian air traffic described in the section on the U.S. national airspace traffic domain. The objective of the experiments were twofold: to (1) understand the performance of the simulation platform with constant computing resources under varying levels of system load—that is, the size/complexity of the simulation scenario—and (2) understand the

performance of the simulation system under varying distributions of the available resources between key platform components.

We discuss the simulation results from the point of view of the ability to effectively distribute and use available resources. The whole system also generates a large number of parameters for each airplane (e.g., airplane type, engine type, length and duration of the flight plan, total fuel consumption, the number of separation violations, and many others). These data can be evaluated and used for comparison with real data provided by the appropriate authority (Federal Aviation Authority, EUROCONTROL, etc.).

Metrics

The scalability and maximum simulation speed simulation requirements have been analyzed based on the time required to complete a specified simulation in the as-fast-as-possible simulation mode. The high-fidelity requirement is guaranteed by using a time-stepped simulation approach with constant length of time cycle period and no simulation level of details techniques.

In all experiments, the time to complete the simulation was measured. For a given simulation configuration it is the real time (wall clock) from the start of the simulation to the moment where whole simulation scenario (flights) are completely simulated. Each configuration has been measured using identical initial conditions (freshly launched JVMs [Java Virtual Machines], no allocated memory, etc.) five times and the values presented are the averages from these runs.

The processing time is inferred from the time to complete the simulation multiplied by the number of processing units (computational threads in CPUs) available for the particular simulation configuration. The simulation load corresponds to the number of total flights that have to be simulated. Each flight typically has different durations, departures, arrival airports, and intermediate waypoints. The simulation load is expressed as a percentage of the complete air traffic and 50% load means that each second airplane has been simulated and the selection is done based on departure time. Even though there can be several airplanes scheduled at the same time, the selection process is deterministic and thus is the same for each repetition.

The number of effective airplanes is the mean number of airplanes running during the whole simulation in a particular simulation load. The normalized processing time expresses the computational real time (wall clock) per one situated entity in the simulation. Using normalized processing time, we can study the efficiency of the computational resources used in various configurations.

Testing Configuration

The configuration contained a set of several heterogenous (both in terms of the operating system and hardware) computer nodes. During experiments,

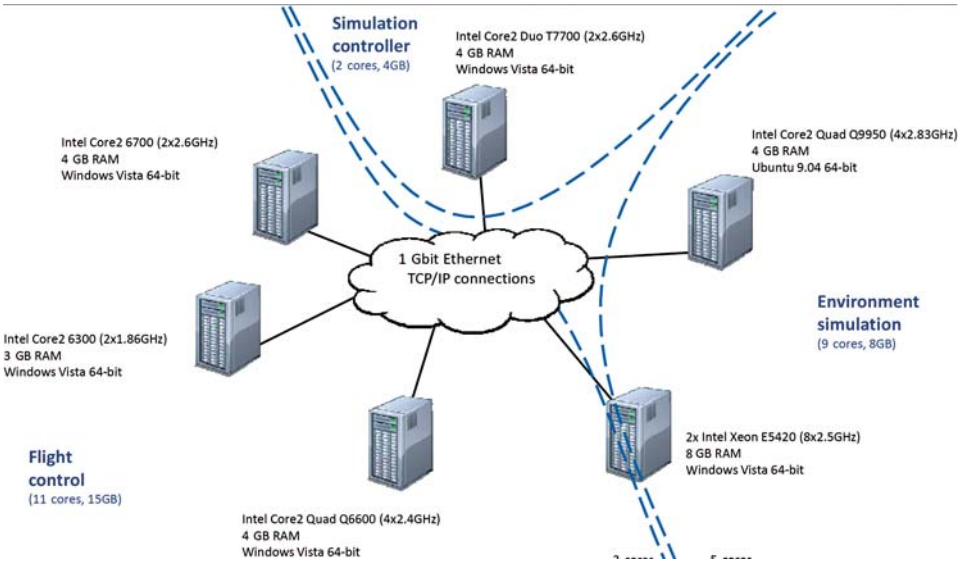

FIGURE 6 Computer nodes allocation during experiment: environment simulation, hosts environmental components; flight control, hosts all deliberative control parts; and simulation controller, other components for simulation control and input/output interfacing with the simulation (color figure available online).

nodes were dedicated for ESCs or deliberative controllers; see computer node allocation in Figure 6.

The testing configuration consisted of 9 cores with 9-GB RAM for ESCs and 11 cores with 15-GB RAM for the flight control integrated in deliberative controllers. All computers were interconnected through a 1-Gb Ethernet network (connection through a single switch) using the TCP/IP protocol. All systems were running a Sun Java 64-bit 1.6.0 14 environment. All empirical values are averaged from five independent measurements with identical initial conditions; for example, a freshly launched JVM, no allocated memory, etc.

Experiment 1: Varying Number of Entities

In the first experiment, hardware resources were fixed and time to complete the simulation was measured for varying simulation load. Specifically, simulation load varied from 10 to 100% of all flights for the same data set with a 10% step.

Figure 7 shows the dependency of time to complete the simulation and normalized processing time on the number of situated entities within the system. The maximum simulation duration was less than 13 min for all 52,799 flights in the experimental data set. First, the normalized processing time decreased with the increasing simulation load up to 70% and then increased.

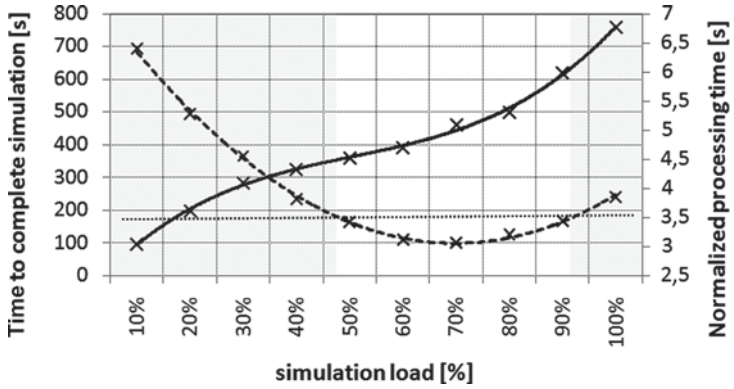

FIGURE 7 Experiment 1: Varying number of entities. Time to complete simulation—solid line, left axis. Normalized processing time—dashed line, right axis.

We can define a threshold based on the lowest normalized processing time where hardware resource utilization is efficient. In our case, we specified that the simulation was efficient when normalized processing time is within 20% limit from the best utilization. Therefore, for the best normalized processing time of 3 s, the threshold was set to 3.6 s (dotted line in Figure 7).

Based on the identified threshold and the normalized processing time, we can identify three regions (grey areas in Figure 7 from left) in simulation load: (1) underloaded, (2) efficiently loaded, and (3) overloaded simulation.

For the fixed computational resources and low system load (few situated entities), the simulation is underloaded and is thus inefficient. Available resources are not efficiently used due to the few entities allocated and there is a maintenance overhead for the distributed architecture that depends on both the number of partitions given by the number of available processing computational threads and the number of situated entities in the simulation. The overhead for the distributed architecture includes both the coordination between partitions and exchange of states required for computation of inter-physics.

In the efficient simulation load region (middle of Figure 7), the best ratio is between resources consumed for distributed simulation maintenance and computation of state updaters. This leads to an almost linear dependency of time to complete the simulation on the number of simulated entities.

With increasing simulation load we can reach a limit where available hardware resources become overloaded (right side of Figure 7). The simulation can be overloaded due to three main reasons:

- CPU overload: The simulation management is designed to run the simulation as fast as possible considering the CPU load. If the CPU performance is not sufficient, the simulation continuously slows down.
- Memory overload: The higher memory requirement for the simulation than memory available to the operating system results in memory swapping and

higher probability of page faults, decreasing the performance of the simulation.

- Network overload: Despite the distributed design, which inherently aims to minimize the network traffic, the network can become overloaded. If the network is not able to transmit all traffic required by the simulation especially related to the environment simulation, the simulation is paused and waits for their delivery.

In our configuration, both CPU and memory resources were usually overloaded. While overloaded, the simulation slowed down exponentially in increasing load.

Experiment 2: Varying Computational Resources

In the second experiment, in contrast to the first, the simulation load was constant and time to complete the simulation was studied for varying computational resources. Specifically, a 40% simulation load was selected in order to use less hardware than in the full configuration. This experiment covered two different cases: (1) varying resources for environmental simulation agents and (2) varying resources for deliberative control agents.

In the first case, varying resources for ESC agents and all available computational resources for deliberative control agents were used (see Figure 6) and these were not changed. Such a configuration should minimize the impact of the deliberative control configuration on the measurement of dependency on ESC agents' configuration. In fact, the number of assigned cores for ESC agents specifies the number of partitions during the simulation—each partition has one core assigned.

Thus, the dependency of time to complete simulation on the number of partition was studied. The results from this experiment are shown in Figure 8.

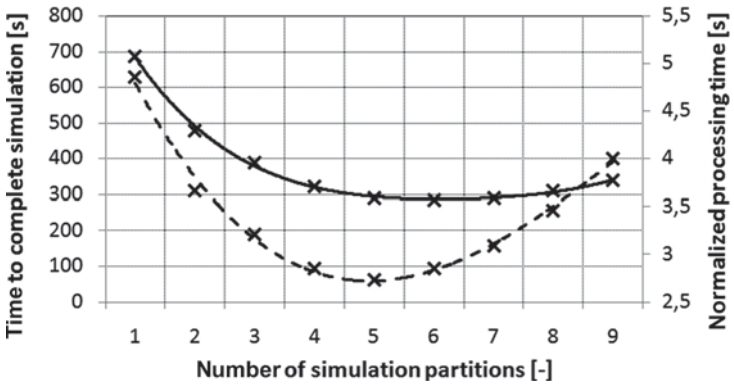

FIGURE 8 Experiment 2: Varying the number of partitions. Time to complete simulation—solid line, left axis. Normalized processing time—dashed line, right axis.

Similar to the results of experiment 1, we can observe three utilization regions: (1) overloaded, (2) optimally loaded, and (3) underloaded. In the left part of the figure (one, two, and three partitions), the normalized processing time is significantly decreased with more partitions. For fewer partitions, ESC agents including state updaters and computing inter-physics are overloaded. For four and five partitions, the time to complete simulation is still reduced; however, the normalized processing time stagnates. This is caused by rising demands for partition-to-partion coordination. Adding even more computational resources for ESC agents (thus using more partitions) results in a decreasing normalized processing time. In this underloaded case, the overhead for partition-to-partition coordination dominates the parallelized processing of state updaters. Note that values for nine partitions in Figure 8 are the same as for the 40% simulation load in Figure 7.

In the second case, varying resources for deliberative control agents, a fixed number of partitions was used. Thus, the effect of changing resources for the deliberative control part is primarily studied. Figure 9 presents the dependency of time to complete simulation and normalized processing time on increasing hardware resources. In the left part of the figure (two, three, and four cores for deliberative control agents), there is a significant speed-up given by parallelization of execution of heavy-weight deliberative controllers. For less computational power, all cores are overloaded and the simulation is slower.

For increasing number of computational cores assigned for deliberative controllers, the time to complete simulation is almost the same. In such a case, the overall simulation bottleneck is not in the speed of evaluation of deliberative controllers. We cannot speed up more due to environment simulation. Looking at normalized processing time, we can see that for increasing resources (six, seven, and more computational cores), computational power is wasted and thus normalized processing time is increasing due to increased

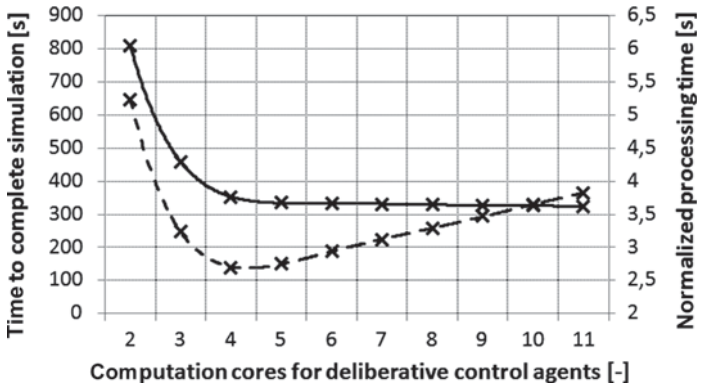

FIGURE 9 Experiment 2: Varying the computational resources for deliberative control agents. Time to complete simulation—solid line, left axis. Normalized processing time—dashed line, right axis.

processing time and not increase time to complete the simulation. In contrast to the first case, the addition of more resources for deliberative controller agents does not slow down the simulation.

CONCLUSION

We have investigated the problem of efficient simulation of a large number of situated entities operating in large-scale virtual worlds. The distributed, fully scalable multi-agent approach with automated load balancing has been proposed to address the problem both in environment simulation agents by dynamic repartitioning and for deliberative controllers by start-up balancing. The novelty of this approach is running both synchronous and asynchronous parts of the simulation at the same time, leading to more realistic behavior of the entire simulation. The approach efficiently exploits the predominantly local nature of interactions in situated agent-based simulations and uses it to efficiently partition the virtual world and distribute it over multiple computer nodes. The approach has been implemented as a multi-agent system and evaluated in the air traffic domain while simulating complete civilian air traffic touching the U.S. NAS involving a total of 52,799 flights with up to 6,500 simultaneous airplanes in less than 13 min using six computers.

The evaluation confirms the ability of the approach to handle very large-scale simulations. However, the hardware resources have to be carefully assigned. The simulation performance is maximized by the proper distribution of computational resources among environmental simulation agents and deliberative controller agents. Underresourcing of any part leads to a very significant decrease of the overall simulation performance. Overresourcing of the deliberative controllers has no negative effect to the simulation performance, and available resources will be utilized when the simulation load increases. On the other hand, overresourcing of the environment simulator agents and thus over partitioning of the virtual world cause a decrease in the simulation performance due to an additional overhead for partition-to-partition coordination induced by a higher number of partitions.

The architecture presented uses available hardware exclusively by environment simulation agents or by deliberative controllers, and this assignment is fixed during the simulation. In the future, computational resources should be adaptively assigned during the simulation depending on current needs so that the simulation performance is maximized under changing conditions during the simulation.

REFERENCES

Agogino, A. and Tumer, K. "Regulating Air Traffic Flow with Coupled Agents." In *Proceedings of the 7th International Joint Conference on Autonomous Agents*

and Multiagent Systems, edited by Lid Padgham, David Parkes, Jorg Muller, and Simon Parsons, Vol. 2. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2008.

Banks, J., Carson, J. S., Nelson, B. L., and Nicol, D. M. *Discrete-event System Simulation*. Upper Saddle River, NJ: Prentice Hall, 2001.

The Boeing Company. *Current Market Outlook 2008–2027*. Seattle: Author, 2008.

Borshchev, A. and Filippov, A. "From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools." In *Proceedings of the 22nd International Conference of the System Dynamics Society*, edited by Michael Kennedy, Graham W. Winch, Robin S. Langer, Jennifer I. Rowe, and Joan M. Yanni, 25–29. Oxford, UK: John Wiley & Sons, 2004.

Brom, C., Šerý, O., and Poch, T. "Simulation Level of Detail for Virtual Humans." In *Proceedings of the 7th international conference on Intelligent Virtual Agents (IVA'07)*, edited by Catherine Pelachaud et al., 1–14. Paris: Springer-Verlag, 2007.

Dahmann, J. S., Fujimoto, R. M., and Weatherly, R. M. "The Department of Defense High Level Architecture." In *Proceedings of the 29th Conference on Winter Simulation*, edited by Sigrun Andradottir, Kevin J. Healy, David H. Withers, and Barry L. Nelson, 142–9. Washington, DC: IEEE Computer Society, 1997.

Fujimoto, R. M. *Parallel and Distribution Simulation Systems*. New York: John Wiley & Sons, 1999.

Gorodetsky, V., Karsaev, O., Samoylov, V., and Skormin, V. "Multi-agent Technology for Air Traffic Control and Incident Management in Airport Airspace." In *Proceedings of the International Workshop Agents in Traffic and Transportation*, 119–25. Estoril, Portugal: International Foundation for Autonomous Agents and Multiagent Systems, 2008.

Guo, Y., Gong, W., and Towsley, D. "Time-stepped Hybrid Simulation (TSHS) for Large Scale Networks." In *IEEE INFOCOM 2000. Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 2, 441–50. Tel Aviv: IEEE, 2000.

Hill, J. C. F., Johnson, R., Archibald, J. K., Frost, R. L., and Stirling, W. C. "A Cooperative Multi-agent Approach to Free Flight." In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 1083–90. New York: ACM Press, 2005.

Hwang, I., Kim, J., and Tomlin, C. "Protocol-based Conflict Resolution for Air Traffic Control." *Air Traffic Control Quarterly* 15, no. 1 (2007): 1–34.

Jamali, N. and Zhao, X. "Distributed Coordination of Massively Multi-agent Systems." In *Massively Multi-Agent Technology*, edited by Nadeem Jamali, Paul Scerri, and Toshiharu Suguwara, 13–27. Hakodate, Japan: Springer, 2008.

Krozel, J. and Doble, N. "Simulation of the National Airspace System in Inclement Weather." In *AIAA Modeling and Simulation Technologies Conference*. Hilton Head, SC: American Institute of Aeronautics and Astronautics, 2007.

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

Lui, J. C. S. and Chan, M. F. "An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems." *IEEE Transaction on Parallel Distributed Systems* 13, no. 3 (2002): 193–211.

Morillo, P., Ordufia, J. M., and Duato, J. "A Scalable Synchronization Technique for Distributed Virtual Environments Based on Networked-server Architectures." In *Parallel Processing Workshops, 2006*, edited by Timothy Mark Pinkston and Fusun Ozguner. Columbus, OH: IEEE Computer Society, 2006.

National Research Council Panel on Human Factors in Air Traffic Control Automation. *The Future of Air Traffic Control: Human Operators and Automation*. Washington, DC: National Academy Press, 1998.

Nicol, D. M. and Yan, G. "Discrete Event Fluid Modeling of Background TCP Traffic." *ACM Transactions on Modeling and Computer Simulation* 14, no. 3 (2004): 211–50.

Nolan, M. S. *Fundamentals of Air Traffic Control*, 4th ed. Belmont, CA: Thomson Brooks/Cole, 2004.

Nuic, A., Poles, D., and Mouillet, V. "BADA: An Advanced Aircraft Performance Model for Present and Future ATM Systems." *International Journal of Adaptive Control and Signal Processing* 24, no. 10 (2010): 850–66.

Schulz, R., Shaner, D., and Zhao, Y. "Free-flight Concept." In *Proceedings of the AIAA Guidance, Navigation and Control Conference*. New Orleans, LA: American Institute of Aeronautics and Astronautics, 1997.

Šišlak, D., Rehak, M., Pěchouček, M., Rollo, M., and Pavlíček, D. "A-globe: Agent Development Platform with Inaccessibility and Mobility Support." In *Software Agent-Based Applications, Platforms and Development Kits*, edited by R. Unland, M. Klusch, and M. Calisti, 21–46. Berlin: Birkhauser Verlag, 2005.

Šišlak, D., Samek, J., and Pěchouček, M. "Decentralized Algorithms for Collision Avoidance in Airspace." In *Proceedings of the 7th International Converence on Autonomous Agents and Multi-Agent Systems (AAMAS)*, edited by Lid Padgham, David Parkes, Jorg Muller, and Simon Parsons, 543–50. New York: ACM Press, 2008.

Šišlak, D., Volf, P., Komenda, A., Samek, J., and Pěchouček, M. "Agent-based Multilayer Collision Avoidance to Unmanned Aerial Vehicles." In *Proceedings of International Conference on Integration of Knowledge Intensive Multi-Agent Systems (KIMAS)*, edited by Henry Hexmoor and Craig Thompson, 365–70. Piscataway, NJ: IEEE, 2007.

Šišlak, D., Volf, P., and Pěchouček, M. "Accelerated A* Trajectory Planning: Grid-based Path Planning Comparison." In *Proceedings of the 19th International Conference on Automated Planning & Scheduling (ICAPS)*, edited by Alfonso Gerevini et al., 74–81. Menlo Park, CA: AAAI Press, 2009a.

Šišlak, D., Volf, P., and Pěchouček, M. "Flight Trajectory Path Planning." In *Proceedings of the 19th International Conference on Automated Planning & Scheduling (ICAPS)*, edited by Alfonso Gerevini et al., 76–83. Menlo Park, CA: AAAI Press, 2009b.

Volf, P., Šišlak, D., Pěchouček, M., and Prokopova, M. "Convergence of Peer-to-peer Collision Avoidance Among Unmanned Aerial Vehicles." In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT)*. Los Alamitos, CA: IEEE, 2007.

Waters, R. C. and Barrus, J. W. "The Rise of Shared Virtual Environments." *IEEE Spectrum* 34, no. 3 (1997): 20–25.

Weyns, D., Van Dyke Parunak, H., Michel, F., Holvoet, T., and Ferber, J. "Environments for Multiagent Systems State-of-the-art and Research Challenges." In *Environments for Multi-Agent Systems*, edited by Danny Weyns, H. Van Dyke Parunak, and Fabien Michel. New York: Springer, 2004.

Wu, Y. and Gong, W. "Time-stepped Simulation of Queueing Systems." *Proceedings of SPIE* 4367 (2001): 262.

Yamamoto, G., Tai, H., and Mizuta, H. "A Platform for Massive agent-based Simulation and its Evaluation." In *Massively Multi-Agent Technology*, edited by Nareem Jamali, Paul Scerri, and Toshiharu Suguwara. Hakodate, Japan: Springer, 2008.

Zhou, S., Cai, W., Lee, B. S., and Turner, S. J. "Time-space Consistency in Large-scale Distributed Virtual Environments." *ACM Transactions on Modeling and Computer Simulation* 14, no. 1 (2008): 31–47.