Distributed Air Traffic Control Simulator

Ranko Radovanović, Zaharije Radivojević, and Miloš Cvetanović

Abstract — **During initial training air traffic control students acquire theoretical knowledge in various fields including air traffic management, aircraft performance, air traffic control equipment and systems, navigation and others. This paper proposes a simulator and explains its use and features that allows students to gain a practical insight into their coursework in order to complement their training. The goal of the simulator is to realistically implement all the key functionalities needed to cover the topics that were presented in class. The simulator offers a user friendly, distributed, and multi-role environment that can be deployed on regular PCs. Moreover, this paper discusses and resolves some of the main conceptual and implementational issues that were faced during simulator development.**

Keywords — **Air traffic control, air traffic controller training, distributed application, simulator.**

I. Introduction

THE training of Air Traffic Controllers in Europe today is largely standardized by EUROCONTROL's Common Core Content [1]. Concentrating on the Initial part of the training, this concept consists of theoretical courses such as air traffic management, aircraft performance, navigation, equipment and systems, human factors, etc. All course topics intertwine and can be practically presented by the use of a simulator that contains all the functionalities and characteristics required for students to grasp everything they learned in class. The key functionalities that a simulator should support are aircraft symbol and label interpretation, controller tools, safety nets, coordination difficulties, phraseology practice, etc. Apart from possessing all the necessary functionalities, the simulator usability characteristics need to be realistic [2], unambiguous and user-friendly, as are the systems intended for the use by air traffic controllers [3] [4].

This paper discusses some of the issues raised while implementing the simulator that provides key functionalities and characteristics appropriate for training purposes. Section 2 of the paper outlines the systems used in air traffic control today. Section 3 explains the architecture of the proposed simulator, while Section 4 explains the implementation of the simulator, as well as some issues that were raised during the process. Section 5 gives a brief evaluation of the proposed simulator and Section 6 outlines the future work and concludes the paper.

Ranko Radovanović, (e-mail: ranko.r@sezampro.rs).
Zaharije Radivojević, School of Electrical Engineering, University of Belgrade (e-mail: zaki@etf.bg.ac.rs).
Miloš Cvetanović, School of Electrical Engineering, University of Belgrade (e-mail: cmilos@etf.bg.ac.rs).

II. Systems Used in Air Traffic Control

An air traffic control system as a whole is very extensive in its complexity and its key element is a data processing system (DPS). The working process of the DPS can be simplified as gathering data from different surveillance sensors, such as automatic dependent surveillance (ADS) sensors, processing them for each uncovered aircraft, and correlating them together with data from flight plans and other flight plan related messages. The result is a set of information containing processed aircraft positions (tracks) associated with their corresponding group of flight plan data (labels). Afterwards, the set of information is distributed to each of controller machines or controller working positions (CWP), which then display it on their screens in the form of symbols and letters. The described process is repeated at defined intervals, but every time using new information as the sensors constantly collect new data and flight plans are updated with new incoming messages. Fig. 1 depicts a future ADS environment that is based on position reports received over ADS-B datalink from an aircraft that calculates its position using a global navigation satellite system (GNSS).

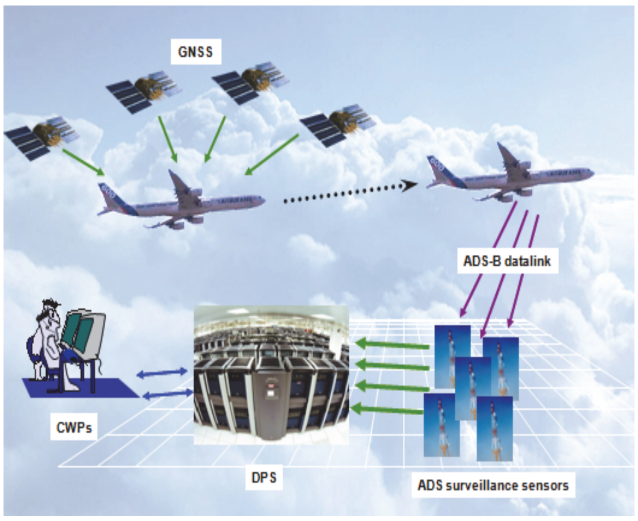


Fig. 1. Example of a future ADS environment.

Apart from its main objective the DPS is used for an increasing number of different system functions: sector allocation, automatic coordination, controller tools and aids, safety nets, etc. As the number of functionalities increases so does the number of machines the DPS is run on. In order to reduce the vulnerability of the system, the DPS is nowadays most often implemented as a distributed system with its functionalities logically placed in different groups which are then run on different server machines.

Certain functionalities have real-time behavior (meaning that they have a guaranteed response time) and the system as a whole is failure resistant, however these characteristics are not essential to air traffic control simulators since they are not used on real traffic and other systems do not depend on them.

Other systems used in air traffic control are of lesser importance for this paper and can be briefly enumerated as communication and navigation systems. The communication systems are used by air traffic controllers for communication with each other, with aircraft crew, or with any other necessary unit. The navigation systems enable aircraft crew to determine its horizontal and vertical position, safely fly and navigate the aircraft, and execute air traffic control instructions.

III. Proposed Solution

The simulator is intended to be run on regular PCs to decrease the cost and to avoid the necessity for specialized hardware. Section 2 of the paper implies that in order to be representative the simulator needs to be a distributed system consisting of applications with three different roles: server (DPS and administrator), controller (CWP) and pseudo-pilot (PWP) working positions. All the applications are run on different machines as controller and pseudo-pilot positions need to be both logically and physically separated, while the server is left on a separate machine with the aim of increasing its efficiency. There is always a single server, whereas the number of controller and pseudo-pilot positions may vary (theoretically it is infinite) and is always equal as each controller position has its corresponding pseudo-pilot position. This is due to the limitation imposed by radio frequency based controller-pilot communication. Thus the simulator is run on k=2n+1 machines, where n is a number of controller positions needed for the exercise.

An additional important limiting factor is the way the applications communicate with each other. Due to reasons outlined in Section 4 of the paper, communication is accomplished through both TCP/IP and UDP protocols. Since the UDP protocol is unreliable (does not assure that the sent package is received) the problem of reliability arises. This issue is solved by placing all simulator engaged machines on a local LAN network, preferably with a gigabit bandwidth, minimizing the possibility of lost packages. The arising restriction is not a problem considering that the controller training is done in a class-like fashion meaning that the machines will be placed in proximity to each other.

Another emerging limitation is the group address reservation which is needed for sending and receiving multicast UDP packages. In order to receive a multicast package all of the machines need to subscribe to a group address, meaning that all the packages sent to the group address will automatically be resent to their IP addresses. To simplify this problem and avoid different TTL (Time To Live) of packages during the communication process the local network must have only one router [5]. For the network communication to work in parallel, certain logical division of the messages needs to be made and certain ports and IP addresses need to be reserved for the simulator use only. Fig. 2 outlines the simulator architecture and different protocol packages which are exchanged through the network during application communication.

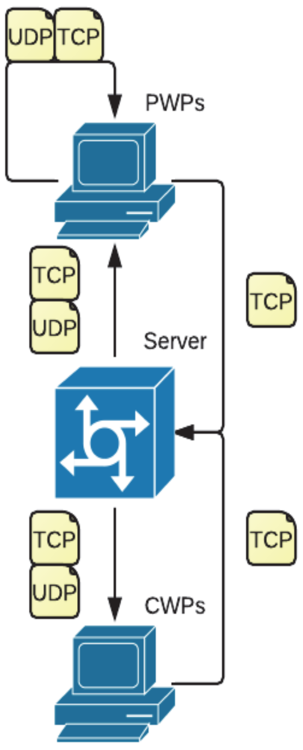


Fig. 2. Outline of simulator architecture.

In order for the solution to be platform independent, the simulator was developed in Java programming language, hence it can be compiled and run on almost all of today’s operating systems through the use of an appropriate Java Virtual Machine. The simulator uses java.net package and Java 2D API which have all the necessary classes needed for network and two-dimensional graphics programming. During the development phase of the simulator, an apprehensive object-oriented analysis of all user interaction was made like the ones in [6], [7] and [8].

IV. Implementation

As described in Section 3, the simulator is implemented as a distributed system consisting of three different inter-dependent types of applications that constantly exchange information. One application is used for calculation, data processing and communication mediation, therefore it is referred to as the server. Other two types are largely concentrating on information visualization and user input gathering. They are referred to as client applications (controller and pseudo-pilot).

A. Server application

Server application is the core of the simulator. It is used to simulate the DPS and to provide administrative tools such as exercise selection, initialization and control. DPS is implemented through different concurrent threads, each thread representing a separate group of functionalities. The main function of the DPS is tracking and track distribution as described in Section 2. During simulation two key issues arise: coherency and synchronization of tracks.

Coherency is related to the necessity that all of the controller positions have the identical representation of key track elements at any given time. The key track elements supported by the simulator are: call sign, actual position, position history and prediction, flight level and ground speed. The coherence is achieved by ensuring that all of the tracks are distributed to every controller position at the same time, which is done by using multicast. The distribution is repeated every 5 seconds in resemblance to some present ATC (Air Traffic Control) systems. The distribution process creates a predicted series of timestamps at which the distribution is made, thus raising the question of track synchronization problem.

Lack of track synchronization could be easily observed on controller screens as a phenomenon where after an update one aircraft changes its position and the other does not, or a slower aircraft travels a longer distance than the faster one. The problem occurs due to the fact that surveillance sensors receive information asynchronously, especially when different aircrafts are concerned. Consequently, there is no way to precisely calculate the time at which the new position of an aircraft will be received. The problem is resolved by extrapolation. Extrapolation is a process of predicting the future position of an aircraft based on present projected trajectory and ground speed. In order to insure that tracks are synchronized the DPS or server will distribute tracks extrapolated to the exact time of distribution. Nevertheless, the solution may raise a question of distributed tracks reliability. The question is answered by changing the status of a track when its third repeated update is based solely on extrapolated data.

For coherency and track synchronization to be reliable, a certain time keeping mechanism needs to be implemented so that the server and its clients have synchronized clocks. Hence a central clock is introduced and stored on the server within a thread that "ticks" (it is put to sleep and then awaken) every second. Every time this thread "ticks", it multicasts the new time value to all clients on the network, guaranteeing that all the applications have the same stored time values at each moment. Because of using the synchronized clocks, a certain cumulative loss of accuracy has to be taken into account, since time is lost during the clock thread awaking and multicasting process. Anyhow, as these losses are measured in milliseconds and the projected duration of an exercise is 30-60 minutes the total loss of accuracy can be regarded as negligible.

The server application also keeps flight plans of all active aircraft, a corresponding chain consisting of all the controller positions under whose jurisdiction an aircraft should come to (sector chain) together with an indication of the position under whose control the aircraft is currently (controlling position). By using this information the server implements system supported coordination (SYSCO), allowing different controller applications to coordinate assigned levels, headings, horizontal and vertical speeds, routes and other parameters of aircraft through their associated labels. The coordination is limited so that only the current controlling position and the position next in the sector chain may coordinate for a certain aircraft. Moreover, only the controlling position can alter the flight plan data of an aircraft. The coordination is implemented through the use of defined messages which are based on the EUROCONTROL OLDI standard [9] and under the TCP/IP protocol. The conflict resolution of messages is resolved on the part of the controller application, where the message coming from the accepting position has a higher priority.

Other simulated functionalities of the DPS are controller tools, aids and safety nets like Safety and Emergency Procedures Tool (SEP), Cleared Flight Level Adherence Monitor (CLAM) or Short Term Conflict Alert (STCA) which are in the process of standardization by EUROCONTROL [10] [11], and will not be detailed in this paper. Fig. 3 illustrates the SEP tool and STCA displayed on the controller application screen, STCA warning is given because the projected track separation is less than 5 nautical miles horizontally and 1000 feet vertically. Apart from DPS simulation, the server application also provides administrative functions for selecting, initializing and controlling (play, pause, etc.) different exercises. Every exercise is defined in a prescribed form and kept in a file. Among other parameters exercise definition has the required number of controller positions. This number is used in the process of exercise initialization and dynamic role allocation.

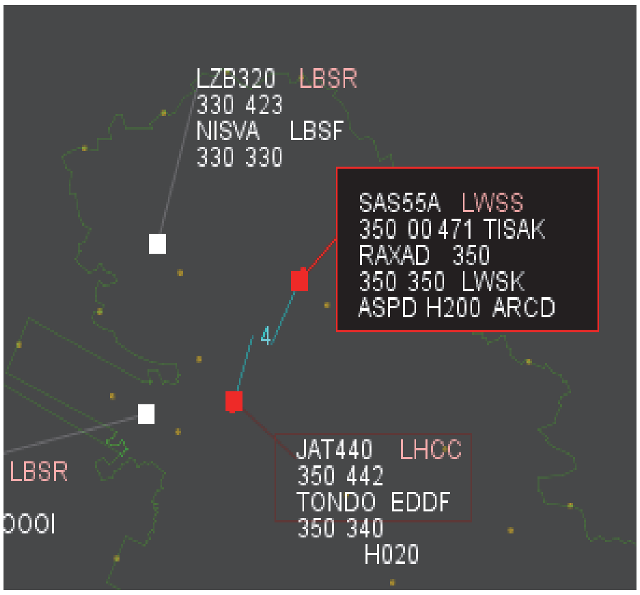


Fig. 3. Representation of SEP and STCA.

After exercise initialization is started by the administrator, the server enters a loop. In this loop the server repeatedly multicasts its IP address on a predefined port and collects client applications (controller or pseudo-pilot) responses on a different port which consist of their IP address and a flag indicating if they are controller or pilot positions. The server exits the loop after it obtains a required number of client applications for the exercise and allocates their roles (assigns sectors to controller and frequencies to pseudo-pilot applications). If the number of

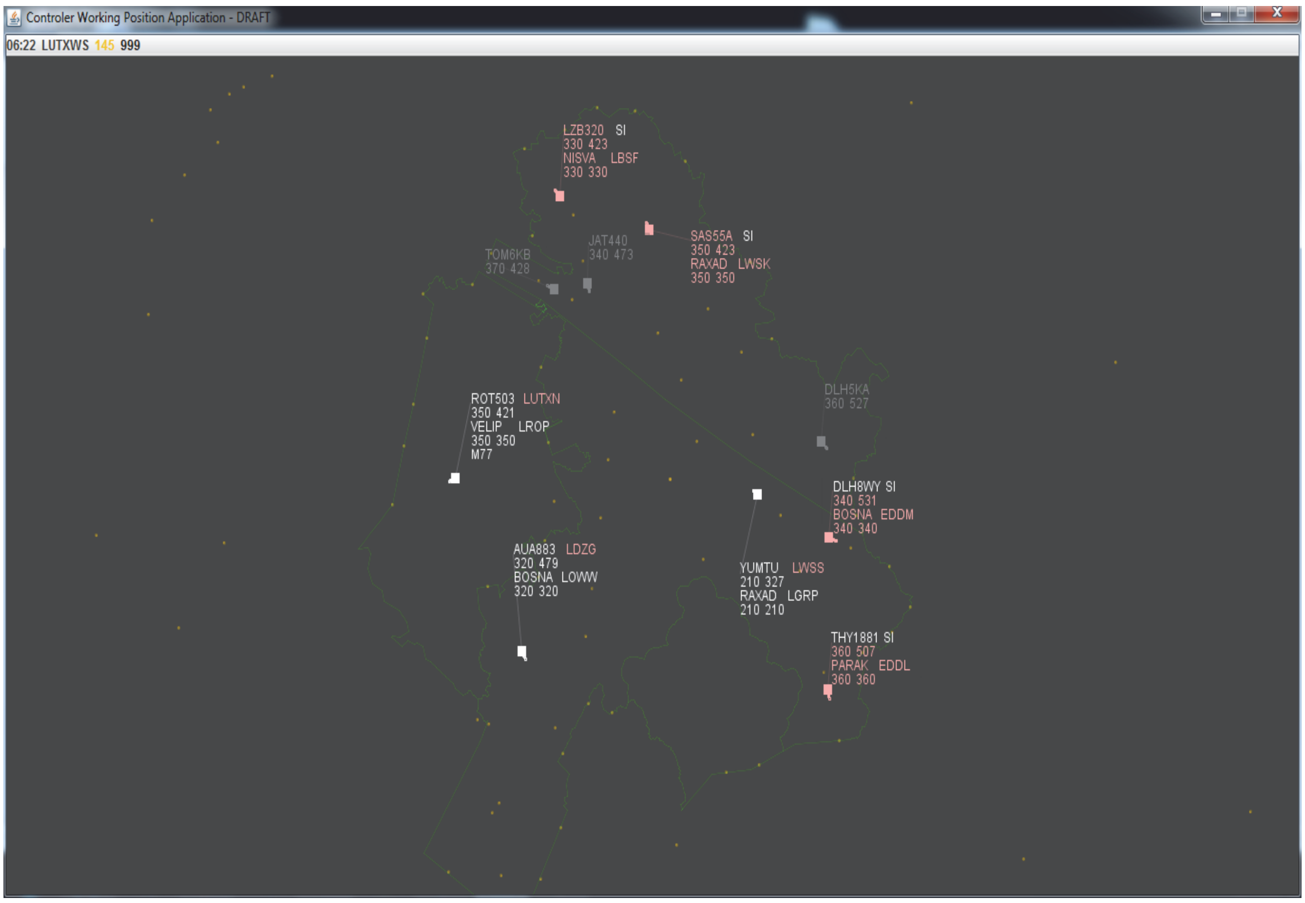


Fig. 4. Typical appearance of Controller application.

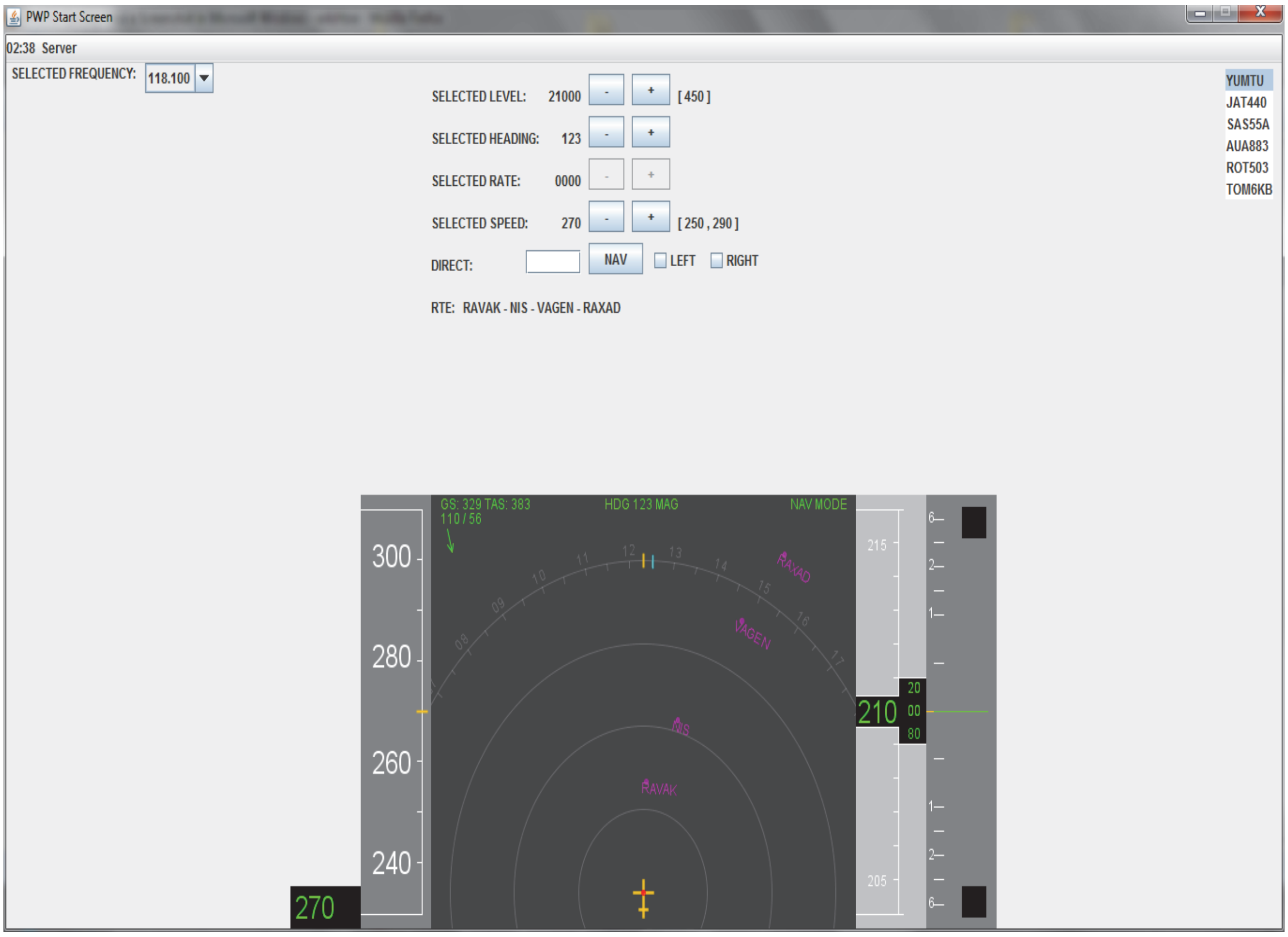
clients is exceeded some of them are unused. The initialization process guarantees that the simulator can be run on any group of PCs that form a local network as described in Section 3 without the need of pre-setup.

B. Controller application

The main function of the controller application is to provide the air traffic controller (or student) with an adequate display of the present air situation, by means of interpreting information and messages coming from the server. An unambiguous representation of aircraft symbols and their corresponding labels, statuses, messages and alerts is achieved by using different colors, letters and shapes, placed on a defined map. The controller can then identify targets, visualize their expected trajectories, do calculations, find conflicts, monitor adherence to given instructions, and interact with the server and other controller positions [12]. The maps are stored in individual files (as part of the controller application) that contain a series of geographical points (latitude and longitude). Fig. 4 illustrates a typical appearance of the controller application or the situation display. Graphical interpretation used in this simulator is largely inspired by existing commercial systems. The airspace used is the one published for the Beograd flight information region. The figure shows a situation display of the south-west sector, where white (ROT503, AUA883, and YUMTU) and pink-white (DLH8WY and THY1881) aircrafts are under its jurisdiction, pink aircrafts (LZB320 and SAS55A) are the ones that are bound to enter its airspace and fall under its jurisdiction but are presently on the north sector. Gray aircrafts (JAT440, TOM6KB, and DLH5KA) are those that are of no concern, as their flights are carried outside of its airspace (through the north sector).

C. Pseudo-pilot application

Pseudo-pilot application is the simulation generator since it creates aircrafts at scenario-defined moments and simulates their flights through the airspace allowing its user to control and monitor aircrafts' trajectory, airspeed, vertical speed, flight level and change the radio frequency that an aircraft is using which ultimately transfers the aircraft to a different pseudo-pilot machine. The scenarios are stored in separate files that are a part of the pseudo-pilot application. During exercise initialization process each pseudo-pilot application receives the name of the scenario it should activate from the server. The scenario contains aircraft generation time, starting horizontal (latitude, longitude) and vertical (flight level) position, vertical profile, call sign, aircraft type, flight planned route, requested flight level and all other information that could be relevant to the exercise. Every aircraft is implemented as an individual thread, which continuously changes its vertical and/or horizontal position based on the present values of numerous changing parameters. Moreover, the pseudo-pilot application simulates surveillance sensors by periodically sending its position (vertical and horizontal) to the server, which is then used for tracking. An important thing to emphasize is that the information passed as sensor information is similar to the ones that would be received by the Automatic Dependent Surveillance receivers containing call sign information and



PWP Start Screen

02:38 Server

SELECTED FREQUENCY: 118.100

SELECTED LEVEL: 21000 [450]

SELECTED HEADING: 123

SELECTED RATE: 0000

SELECTED SPEED: 270 [250 , 290]

DIRECT: NAV LEFT RIGHT

RTE: RAVAK - NIS - VAGEN - RAXAD

YUMTU
JAT440
SAS55A
AUA883
ROT503
TOM6KB

Fig. 5. Typical appearance of Pseudo-pilot application.

avoiding the need for squawks.

Through Pseudo-pilot application controllers (or students) gain a practical insight into the fields of cockpit workload, navigation, wind effect and aircraft performance limitations. Fig. 5 illustrates the typical appearance of pseudo-pilot application. The appearance of some of the flight instruments presented in this application was inspired by those used in the cockpits of commercial airplanes. Accent was given to navigation instruments. The figure shows an aircraft maintaining flight level 210, flying with indicated airspeed of 270 knots on course to RAVAK waypoint.

V. Evaluation

Two aspects of evaluation were made. The first one is concentrated on the simulator's efficiency and scalability to handle a certain optimal number of aircrafts and different sectors at the same time. The other one discusses the proposed concept and implementation in respect to some existing solutions.

A. Efficiency and scalability evaluation

In order to prove its efficiency and scalability the simulator was tested for CPU, dynamic memory and network usage. The number of client applications (sectors) has been varied from 2 to 8 (for scalability) and the number of aircrafts used in test exercises has respectably been within the interval of 16 to 64 distributed equally to all pseudo-pilot applications (for efficiency), as is expected to be the optimum for this type of training. Testing results have shown that the CPU usage has steadily remained within 5 to 10 percent (with rare peaks of 40%), dynamic memory used was constantly in the volume of 5MB, and the 100 Mb/s network load has ranged below 1%. Therefore, the simulator has proved both scalable and efficient. Testing was made on PCs dating from the year 2008 running on Windows XP operating system. All measurements for CPU and dynamic memory usage were made on server application as it was recognized as a potential bottleneck.

B. Comparative evaluation

Existing simulators can be divided into two groups: publicly available and commercial. The drawbacks of the first group of simulators can be summed as: limited number of sectors that can be run at the same time (usually one), the non-existence of pseudo-pilot machines (PWP), operating system dependence, obsolescence, static role allocation, etc. [13] [14] [15] [16]. Commercial simulators which are used in the industry are mostly unavailable for public reach, but their main flaws can be attributed to high exploitation costs, need for dedicated hardware and lack of expandability [17]. All of the mentioned drawbacks have been eliminated by the proposed solution as described in Sections 3 and 4. An overview of the publicly available simulators analyzed and their characteristics from the aspect of the most important functionalities are given in Table 1.

TABLE 1: COMPARISON OF SIMULATORS' FUNCTIONALITIES.

Functionality	***Proposed solution***	***ATC-SIM[13]***	***ATCSimulator[14]***	***SKY-HIGH[15]***	***ISENA Simulator [16]***
Scenario driven	+	+	+	+	+
Multiple simultaneous controller positions	+	-	-	-	+
Standalone mode	-	+	+	+	+
Pseudo-pilot positions	+	-	-	-	-
Platform independent	+	+	-	+	-
System supported coordination (SYSCO)	+	-	-	-	-
Controller tools (QDM and SEP)	+	-	-	-	+
Monitoring Aids (CLAM and RAM)	+	-	-	-	-
Safety NETs (STCA)	+	-	-	-	+

C. *Further simulator enhancement*

During simulator development some issues have been raised that represent tasks for further research and simulator enhancement. For example, an interesting issue that was raised is dynamic sector re-allocation. Although initial sector allocation is dynamic, once it is allocated it becomes static for the remainder of the exercise. There may be a need to open a new or close an existing controller position, which is not supported by the proposed solution. When an existing position is being closed it has to transfer the jurisdiction of aircrafts that were still under its control to the position that will overtake its assigned airspace, the opposite problem arises when a new position is being opened. The described problem deserves thorough research as it raises questions with regard to both software engineering and interface design, but the basic idea would be to use OLDI-like messages in order to change the statuses of certain tracks and their associated labels to emphasize which track is subject to a change of jurisdiction.

Another challenge relates to air-ground communication simulation. Such communication could be carried out through the TCP/IP protocol by transmitting streams of data through the network which represent recorded audio messages. The messages would be exchanged between controller positions and their corresponding pseudo-pilots. A possible solution would be using of two different simulators working in parallel as is the case with most of existing commercial systems where DPS and voice communication systems are separated. Although separated, in order to facilitate the dynamic roles allocation and re-allocation, the existing simulator and the additional communication simulator would constantly need to exchange information.

VI. CONCLUSION

The training of air traffic controllers comprises theoretical knowledge in various fields including air traffic management, aircraft performance, air traffic control equipment and systems, navigation and others. Most trainings include not only lecture classes, but also practical work using simulators. The goal of the simulator is to implement all the key functionalities needed to cover the topics that were presented in class, all in accordance with existing standards. This paper introduces a new simulator, presents its architecture, describes its usage, and evaluates its performance.

The presented simulator met all the goals that were set at the beginning of the paper. It is realistic, scalable, expandable, low-cost, platform independent and relatively easy to use. Using the simulator, students become familiar with system functionalities and tools which are in use by air traffic control, as well as get an insight on the aircraft crew's point of view and aircraft performance limitations. Furthermore, students are able to practice radar skills, coordination and teamwork, phraseology and assimilate with their future airspace and professional environment.

REFERENCES

[1] European organisation for the safety of air navigation "Eurocontrol Specification for the ATCO Common Core Content Initial Training," http://www.eurocontrol.int/sites/default/files/content/documents/single-sky/specifications/20081021-atco-ccc-initial-training-spec-v1.0.pdf, October 2008.

[2] A. Nuic, V. Mouillet, "Enhancement in realism of ATC simulations by improving aircraft behaviour models," *29th Digital Avionics Systems Conference (DASC)*, IEEE/AIAA 2010.

[3] J. Wicks, S. Connelly, P. Lindsay, A. Neal, J. Wang, and R. Chitoni, "Simulation of Air Traffic Controllers' Behaviour Using the Operator Choice Model," Technical Report, ACCS-TR-05-01, ARC Centre for Complex Systems, School of ITEE, The University of Queensland, October 2005.

[4] S. Kocks, C. Paetzold, S. Schoenhals, and P. Hecker, "An ATM simulation environment for the development of HMI technologies," *29th Digital Avionics Systems Conference (DASC)*, IEEE/AIAA 2010.

[5] E. Kristiansen, S. Patella, and M. Mazzoccanti, "An Application Layer Gateway for Air Traffic Management Communication by Satellite," *26th International Communications Satellite Systems Conference (ICSSC)*, June 2008, San Diego, CA.

[6] P. Liguori, and G. Roberts, "A standards-based approach to distributed air traffic management simulation," *International Symposium on Collaborative Technologies and Systems (CTS 2008)*, vol., no., pp.157-165, May 2008.

[7] H. Li, X. He, and Z. You, "Development of the ATC Simulators," 16th International Conference on Artificial Reality and Telexistence--Workshops, ICAT '06. 2006.

[8] J. Wortmann, "Object-Oriented Analysis for Advanced Flight Data Management," Report No. 96-43, Technical University of Berlin, Berlin, Germany, 1996.

[9] European organisation for the safety of air navigation, "Eurocontrol Standard Document for On-Line Data Interchange," http://www.eurocontrol.int/oldi/gallery/content/public/doc/oldi_3-00.pdf, December 2001.

[10] European organization for the safety of air navigation, "First Air Traffic Control Support Tools Implementation homepage," http://www.eurocontrol.int/dossiers/fasti, December 2011.

[11] European organization for the safety of air navigation, "Safety Nets homepage," http://www.eurocontrol.int/safety-nets, December 2011.

[12] ICAO doc.4444 "Procedures for Air Navigation Services – Air Traffic Management, Fifteenth Edition," 2007.

[13] ATC-SIM, "ATC-SIM: a web-based air traffic control simulator," http://atc-sim.com/, December 2011.

[14] AeroSoft, "ATCsimulator®2," http://www.atcsimulator.com/, December 2011.

[15] SKY-HIGH, "SKY-HIGH - Air Traffic Control Simulator," http://playskyhigh.com/, December 2011.

[16] Simulated Sky, "ISENA Air Traffic Control Simulator," http://www.simulatedsky.aero/main.htm?home.htm~mainf, December 2011.

[17] M. Shifeng and W. Danxia, "Implementation of a Flight Control Tower Simulator using Commercial Off-the-shelf Hardware," *Simulation* vol.86, no.2, pp.127-135, February 2010.