*Pak. J. Engg. & Appl. Sci. Vol.*7, *Jul*, 2010 (*p.* 47-54)

An HLA Based Real Time Simulation Engine for Man-in-Loop Net Centric System

Asad Waqar Malik, Shoab A Khan S. Rauf ul Hassan
asad_maalik@yahoo.com; shoab@carepvtltd.com; rauf.hassan@gmail.com
College of Electrical and Mechanical Engineering (CEME)
National University Of Science and Technology (NUST)
PAKISTAN

Abstract

This paper presents a novel HLA based framework, designed to provide interfacing with real time man-in-loop simulation. There is no such framework exists that incorporates real time interaction from external devices. This framework acts as container for federates and responsible for federate management by creating new federates at run time based on real time data input. It is designed and implemented using a modular approach to minimize the delay generated due to run time federate management. In this paper we discuss the issues in implementing real time framework, particularly the time advancement strategy in distributed simulations. This novel framework can be used as a gaming engine for distributed simulations.

It is suitable for simulation war fields, missile systems. In our extensive research we haven't come across any simulation engine that provides interface with man-in-a-loop simulations. Although there are some HLA based framework exists that used to simulate battle fields, missile systems, etc. but our proposed simulation engine deals with man-in-a-loop simulation. Its unique architecture, features e.g. real time federate management, time advancement strategy and real time interfacing with simulation engine and individual federates are the unique characteristics of Real Time Simulation Generator Engine (RTSGE) . Experimental section demonstrates the effectiveness of proposed simulation engine for diverse type of scenarios including man-in-a-loop simulations.

Key Words: *HLA, RTI, man-in-loop, simulation, framework, air defense system*

1. Introduction

Modeling and Simulation (M&S) is widely used to understand the issues in designing physical systems. These systems vary from mathematical modeling to war gamesimulators. Different mathematical models are used for training by Department of Defense (DoD). These models are often implemented using parallel and distributed approach to avail the benefit of parallelism. Parallel and discrete event simulation is often characterized as distributed simulation. The field of PDES becomes more interesting with latest technology advancements, particularly with the availability of multi-core systems. PDES is categorized into two types, conservative and optimistic simulation. Conservative simulation requires all federate to process only safe events, whereas optimistic simulation allows events processing without taking care of time stamp, at the same time it provides a rollback mechanism [1] this approach is termed as *Local Causility Constraints* (LCC). High Level Architecture (HLA) exploit the approach of parallel and distributed simulation, PDES become the central theme for HLA. HLA was designed for interoperability that allows programmers to use build in models. DoD uses HLA for as a central framework for training. Currently most countries deploy High Level Architecture to analyze war techniques, air-to-surface missiles systems [2] before physically launching. The High Level Architecture is an IEEE standard framework for distributed simulation composed of different interacting simulation components [2, 3]. HLA supports reusability and interoperability. HLA is a complete generic framework, in spite of its lacks the features required for real time operations.

Human-in-a-loop is an adaptive simulation where user is directly interacting with the system. HLA and other simulation protocols are specially designed for war games like simulations [4]. War simulations are more adaptive, as it requires command and control system to efficiently managed war scenarios. Most of the government organizations allotted huge budget to train their army using distributed simulators. This is to analyze and increase their skills of decision making[5]. With the innovation of cluster and grid computing, the trend of parallel and distributed simulation becomes much more interesting and innovative; resources are distributed across network. In mid 1990's HLA is a part of many large scale distributed war gaming simulations. Most of the war simulations follow conservative approach as optimistic approach violates local causality constraint. In this paper we present HLA based man-in-a-loop simulation for network centric war games. In this paper, we propose a framework for addressing the real time issues in the development of man in loop simulation. We proposed a generic framework that acts as a Real time simulation generation engine (RTSGE), provides interfacing to real time applications, these applications could be video or data streaming. RTSGE provides simple API calls to connect with external data sources/ applications. RTSGE is responsible for federate management including creating and destroying federate.

In section II we analyze the similar frameworks available, and comparison is drawn with RTSGE, section III introduced the issues in implementing RTSGE; section IV and V shows the implementation details and experimental results and finally the conclusion and future work is presented in section VI.

2. Related Work

HLA become the widely accepted standard for war training, most of the countries uses HLA to simulate its weapon and radar systems. Distributed Object-Oriented war gaming (DOWs) is one of the similar approach used by Korea to test their missile system before physically launching [5]. Recent research shows some significant advancement in HLA and its interfacing with different network simulation tools [6] [7]. Network air defense simulation training system (NADSTS) is a training simulator based on HLA; it uses different plug-in tools like sniffer pro, trace route to provide an extensive training program [8]. NADSTS uses layering approach, divides all the functionality in the three layer architecture. Different countries/organizations working on modeling and simulation, Korean government uses Real time distributed simulation environment (RDSE) to simulate medium range surface-to-air missile system (Chelmae-II). RDSE is used as a performance analysis tool for Chelmae-II system. RDSE is HLA based uses RTI as lower level communication layer [9]. A Raytheon is a USA based company working on HLA based testing and evaluation framework, that allow input from real flight simulators [9]. Another similar work for network war defense was presented in [8], it used HLA as a base line of communication mechanism. Our approach is having some features common with NADSTS and RDSE. RTSGE is an outcome of research based project on training and evaluation having unique architecture, supports real time interaction for man-in-a-loop simulation and time advancement features are the key components. RTSGE is most suitable for defense simulations, and training purpose, the most promising usage of RTSGE is, with TEWA systems [10]. TEWA stands for threat evaluation and weapon assignment system, used by most countries in order to better optimize their resources. RTSGE with its external interface, it can easily incorporate with running TEWA systems.

3. HLA Compliant Framework for Real Time Simulation Generator: Issues and Challenges

HLA is designed to incorporate different simulation models transparently, without any developer efforts. It is used to provide interoperability among existing simulations. Designing real time HLA based man-in-a-loop simulation framework required effective management of issues and challenges. Challenges in incorporating real time data with simulated environments in HLA includes:

- Time management
- Real time federate generation (RTFG)
- Latency (delay)

These capabilities must be handled automatically and transparently. In HLA environment RTI is responsible for time advancement based on tick (). This is a conservative approach of distributed simulation, in which ticks are not granted until all federates requested for tick (). Simulation models where data disseminate in a hierarchical fashion, from one federate to another required effective time management strategy. In traditional RTI, events are transmitted based on tick (); incorporating man-in-a-loop simulation requires delivery of real time data in one tick () instead of multiple ticks, as this mechanism adds additional delay. We designed and implemented Real Time Management Module (RTMM) to handles time management issues.

Real time federate generation is a complicated task, as it is responsible for generating federate at run time based on real time data input. The overhead in generating federate must be minimized. Real time data is incorporating through external interfacing with a delay of nano-seconds. This delay affects the performance of simulation model. Latency is a third important factor in handling real time data, adding multiple layers generate additional overhead. To handle these different modules RTMM, RTFG are incorporated in a single layer.

Figure 1 shows the layering approach used to developed Real Time Simulation Generator Engine (RTSGE). Different layers are communicating with central RTI. Initially only the Module generator is an active federate, it is responsible for dynamic generation of federates based on information from external source. Module generator provided an interface to other applications for communication through Real Time Object Exchange proxy (RTOE). This interface provided simple API calls to send and receive data from simulation engine. The next important functionality of Module generator is tracking of existing federates, it maintained a list of federates currently active. This list is build using discover object function call provided by RTI. Module Generator is acting as a federate. The main idea of making it as a federate is to efficiently re-route the data to the desired end federate through central RTI.

4. Real Time Simulation Generator Engine (RTSGE)

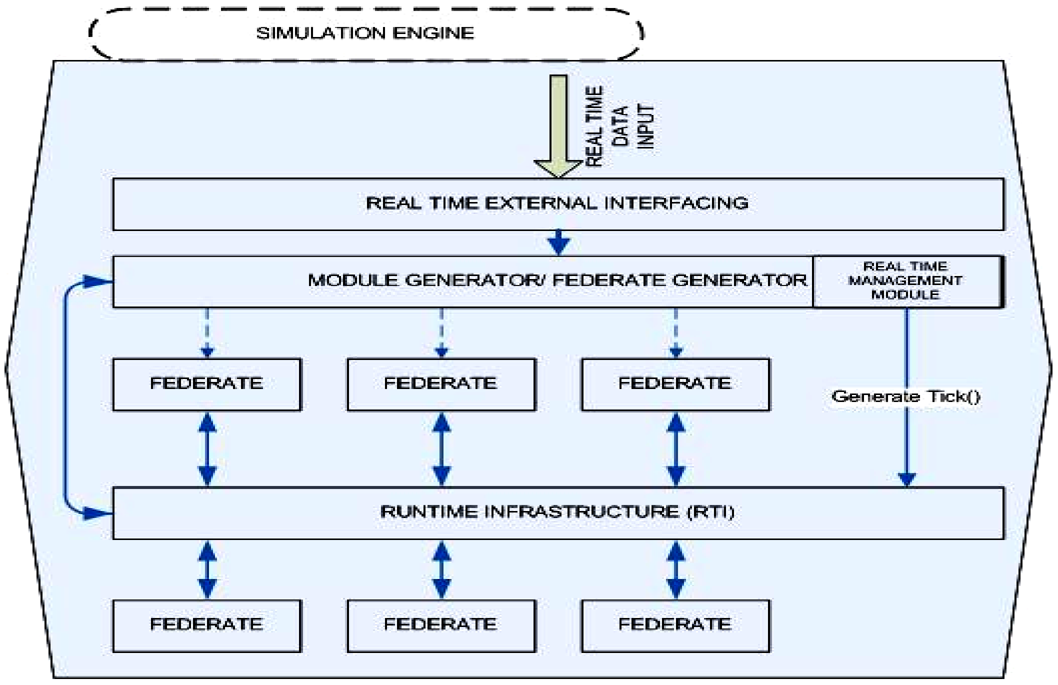
SIMULATION ENGINE
REAL TIME DATA INPUT
REAL TIME EXTERNAL INTERFACING
MODULE GENERATOR/ FEDERATE GENERATOR
REAL TIME MANAGEMENT MODULE
FEDERATE
FEDERATE
FEDERATE
Generate Tick()
RUNTIME INFRASTRUCTURE (RTI)
FEDERATE
FEDERATE
FEDERATE

Fig.1 Shows Real Time Simulation Engine framework

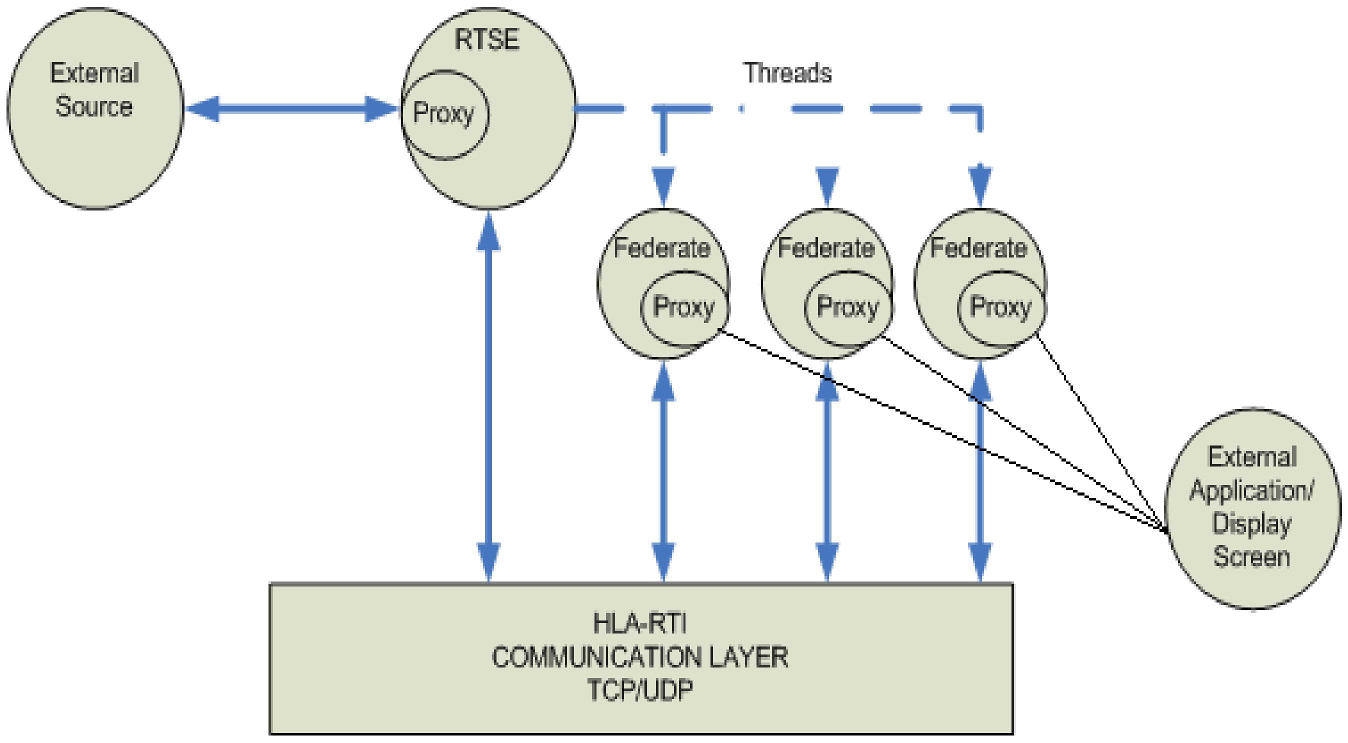

Fig. 2 Complete scenario showing communication model of RTSGE

Timer is an important module in distributed simulation, successful completion of simulation is entirely based on timer. Different approaches are used to efficiently handle timer, i.e. conservative, optimistic and hybrid approach [4] [11]. Lower Bound Timestamp (LBTS) is the minimum time among all federates; different approaches used to calculate LBTS, or Global virtual time (GVT) [12]. In this novel framework the major issue is time synchronization, as there are some federates running simulation time whereas other federates are sending real time data from some external source. Theoretical analysis shows that there two different approaches can be adopted for time synchronization. First is that a dedicated federate running a simulation time can generate tick messages whereas RTI should waits for the real-time federate response. Secondly a dedicated real-time federate is use for time management. The next major issue was to establish a relationship between simulation and real-time clocks. In this framework we incorporated a second approach where a dedicated real-time running federates are responsible for time increment. To cater the difference between time generated by real-time federate and simulated time, we incorporated a self scheduler module called Real Time Management (RTM).

Real Time Management (RTM) is an important concept we introduced in a RTSGE. Time need to be advanced for simulation to progress, having integration of real world with simulated environments leads a time advancement and synchronization problems. RTM module is responsible for handling time advancement and synchronization issues. In HLA-RTI based simulations all active federates need to request for time advancement, after that RTI grant the smallest requested time; having some federate waiting for the data from external source without issuing a time advancement request might lead to a deadlock situation. In particularly to handle this problem, federates are arranged in a hierarchy, based on their publication and subscription policies. Thus it forms different level in a hierarchy table; RTM generate the time advancement tick based on number of levels in a hierarchy. This will enforce delivery of real time object to the destination federate in one simulated time tick. During implementation phase we observed that if every federate including Module Generator issue tick (1) and messages are flow from one federate to another in a hierarchy than after first couple of packets, new packets starts overriding the previous packets before reaching the destination. This can lead to erroneous simulation results. In order to handle this problem, tick

```
Connect RTSGE to RTI
Publish all attributes
Subscribe all attributes
Start RTSGE
While (simulation not end)
     If (receive object from external link)
       Check destination federate exists
            If (! exists)
                 Create thread
                 Pass federate address space to start its normal execution
                 Update active federate list
            End if
            If (federate exists)
                 Call update attribute value for forwarding received data
            End if
            Request for time tick based on number of levels
     End if
End while
Destroy RTSGE
```

Fig. 3 Pseudo code of RTSGE framework

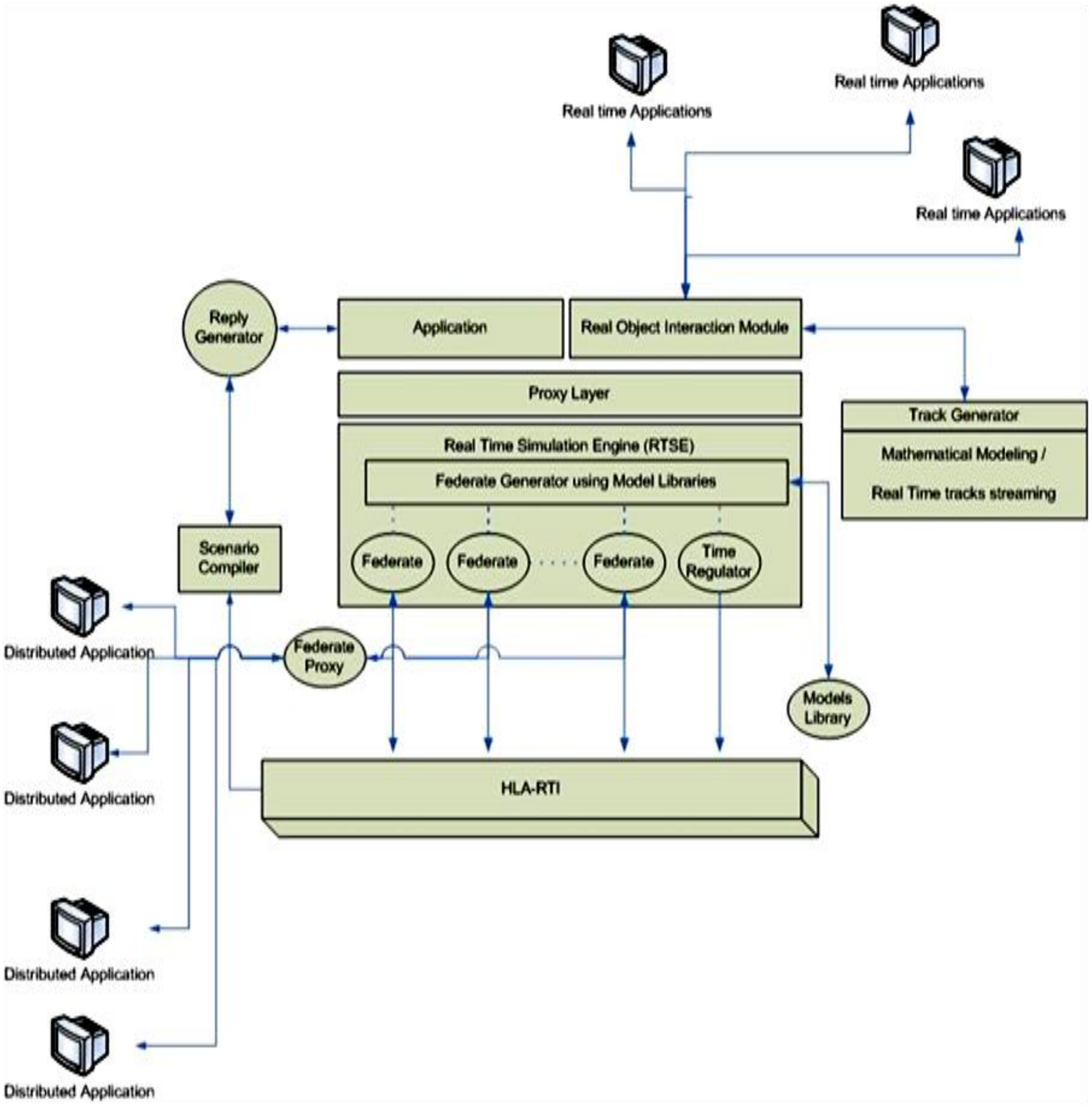

Fig. 4 Architectural view of distributed RTSGE

generated by RTM is based on number of levels in a hierarchy. This allows real time data to reach the destination federate within a single Module Generator tick. Real Time Simulation Engine is useful for developing war games, where large number of real objects like radar, jet etc interacts with each other. RTSGE allows interfacing between different RTSGE through real time object exchange server or through RTI.

When RTSGE receives object data from external source it first check that the federate exist, this is done by maintain a list of active federates. If the federate does not exist, it create a thread federate which act as an independent federate. RTSGE provides a container interface to its created federates. Each generated federate also have a RTOE proxy, which is used to communicate with external applications as shown in Fig. 2. RTSGE is independent of application layer, its independent of application model used. Using RTOE proxy server different GUI's can be incorporated in to the existing system.

Fig.4 demonstrate the complete working scenario, where RTSGE generates different federate threads that act as separate independent objects, communication with HLA-RTI. Each federate thread has an external interface for communication with other distributed applications. These distributed applications could be geographical display or some processing model taking input, processing and analyzing information. Scenario compiler is a special module used to record all the simulated data with respect to federates and local time. Replay generator used this scenario compiler to produce the same results; it provides replay option for analysis purpose. Track generator is a special module used to generate tracks of any moving object. It generates using mathematical model or it can also acquire from other streaming applications. All these modules are carefully designed and implemented keeping space for future enhancements.

5. Implementation

Fig. 5 shows the GUI used to simulate the track of fighter jets and radar system, map based GUI was developed using Map Window API's. GUI allow user to develop scenarios which is simulated by RTSGE, RTOE proxy servers are used to send updated information to GUI.

Series of experiments were performed to measure the delay introduced by Real Time Simulation Engine (RTSGE) by performing its federate creation and management functions. The experimental configuration uses Intel Core 2 Quad CPU Q8200 @ 2.33 GHz with 2 GB of RAM. The systems are connected together using Gigabit Ethernet; for centralize communication we used Portico RTI.

To demonstrate the effectiveness of our RTSGE, we developed a toy application of war game, which consist of radars, fighter jets and air defense missile system. Fighter jet is responsible for generating tracks i.e. its real time coordinates along longitude, altitude and latitude. Radar system is responsible of detecting the motion of fighter jet

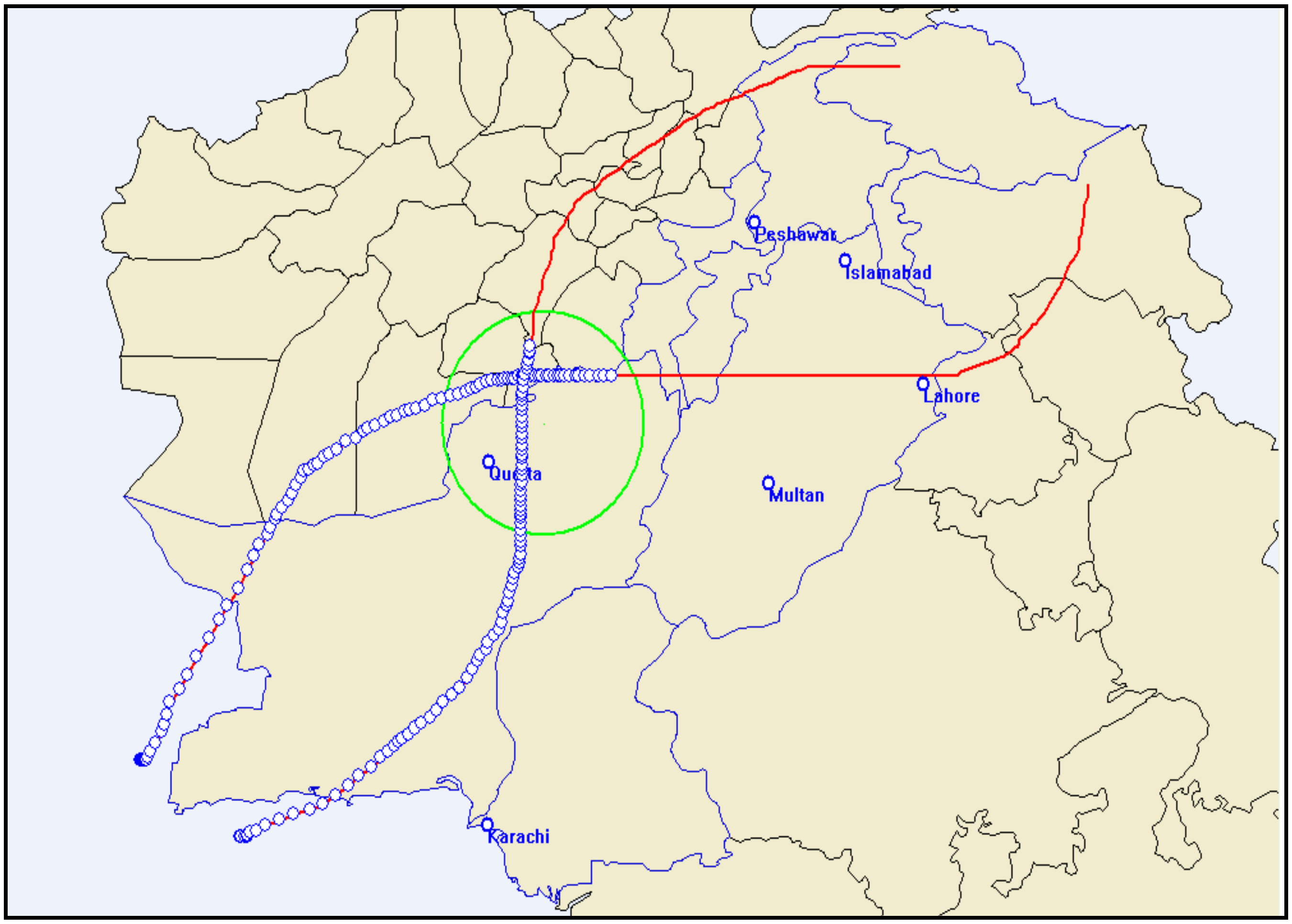

Fig. 5 Shows GUI interface for simulator

under its area of surveillance. We are interested in measuring the delay incur by RTSGE for federate creation, management and in routing real time data through its external interface.

In this experimental setup a Fighter Jet federate sends its location to the RADAR federate. Data is send through real time external interface which is detected by RTSGE and route the data to the destination federate. Destination federate remove the header and calculate the delay. The difference between the source and destination wall clock time is the delay incur by RTSGE. Delay can be given by:

Time delay = Processing delay + Transmission delay + Propagation delay.

Transmission delay is calculated as packet Length (bits) divided by link band width (bps). Since we are using Gigabit Ethernet with only 56 bytes of payload/ update, so transmission delay is close to zero, which is negligible. In order to measure the delay between distributed nodes, time synchronization is required, we used network time protocol (NTP); it measures the drift between times. Drift is measure before any transmitted update. The detailed results after correction are shown in Fig. 6 X-axis represents the wall clock time and y-axis represents number of reading; different tracks have been drawn on GUI for RTSGE testing, results are shown in Fig. 5. It was observed that delay between the systems is 18 ms on the average. It was also observed that delay varies between 0 ms and 32 milli sec. Experiment were repeated number of times to verify the results. The experiment was performed with multiple fighter jets and a radar system. It was observed that delay varies between 0 ms and 78 ms with average value of 15 milli sec. Different factors involved in this variation like scheduler, priority and preemption mechanism used by different operating systems. This investigation is in its initial stage, more detail analysis will be presented in a extended version.

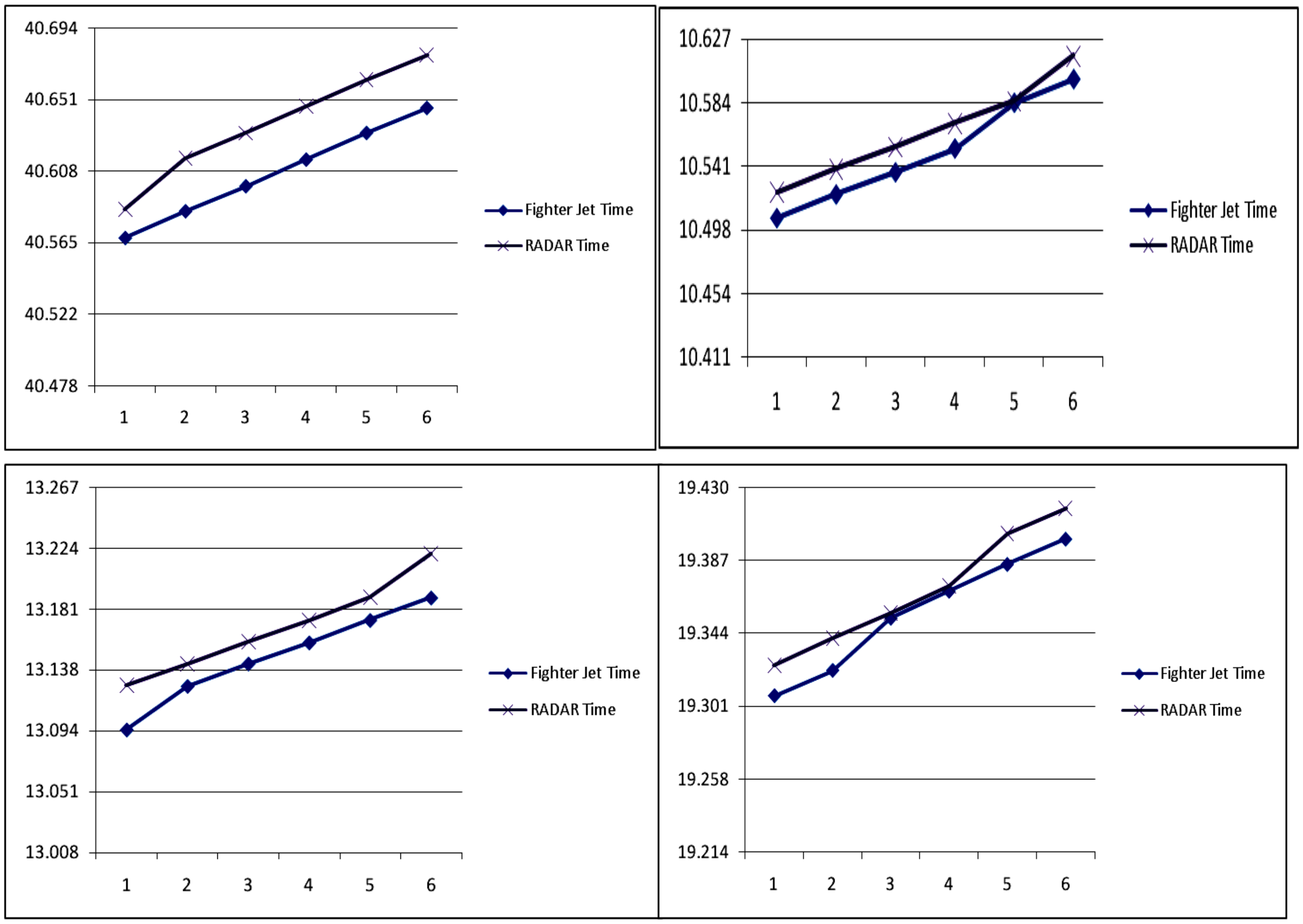

Fig. 6 Results of various tests, x-axis represents the wall clock time (minutes/secs) and y-axis shows the no. of observations

6. Conclusions

Time synchronization between real time and simulation time clock is difficult and it varies from application to application. In this paper we presented a complete framework for real time applications. This framework provides interface to real world systems. The interfacing is simple enough to incorporate in any existing system. Generating federate on-the-fly is another effective way to analyze the real word simulations. The experimental section demonstrates the effectiveness of our approach and measured the time taken by RTSGE to create and activate the federate is convenient. This is the preliminary work shows some satisfying results; in-depth analysis is required to conclude some remarkable achievement in integration of real time simulation engine. This is ongoing project and more detail analysis of each and every layer of RTSGE will be added in a journal paper. In future we are particularly interested in providing interface to other HLA compliant RTI's that could be helpful for large scale simulations.

Acknowledgement

This work is sponsored by National University of Science and Technology, through its research grant fund.

References

[1] R. M. Fujimoto, "Parallel discrete event simulation," *Communications of the ACM archive,* vol. 33, pp. 30-52, 1990.

[2] "Defense Modeling and simulation, High Level Architecture, RTI 1.3-Next Generation programmer's guide" vol. version 3.2, 2000.

[3] "IEEE standard for modeling and simulation M&S High level architecture (HLA) Framework and rules IEEE 1516-2000 " pp. 1-22, Sep.

[4] P. T. Bui, S.-D. Lang, and D. A. Workman, "A New Conservative Synchronization Protocol for Dynamic Wargame Simulation."

[5] J. H. Lim;, T. D. Lee;, and C. S. Jeong;, *Distributed Object Oriented Wargame Simulation on Access Grid* vol. 3516: Springer, 2005.

[6] S. Pawletta, W. Drewelow, and T. Pawletta, "HLA-based simulation within an interactive engineering environment," in *Distributed Simulation & Real-Time Applications, 2000. (DS-RT 2000). Proceedings, 4th IEEE International Workshop on*, 2000, pp. 97-102

[7] A. W. Malik, A. Basit, and S. A. Khan, "HLA compliant network enabled distributed modeling and simulation infrastructure design," in *Proceedings of the 4th WSEAS International Conference on Software Engineering, Parallel \& Distributed Systems* Salzburg, Austria: World Scientific and Engineering Academy and Society (WSEAS), 2005.

[8] C. Gang, X. Shang, J. GuanQun, and J. YiLong, "Network Attack-Defense Simulation Training System Based on HLA," in *Computer Modeling and Simulation, 2009. ICCMS '09. International Conference on*, 2009, pp. 303-306.

[9] B. Cho;, D. Y. Kin;, S. H. Kim;, and C. Youn;, "Real Time distributed Simulation Environment for Air Defense system using a Software Framework," *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology,* vol. 4, pp. 202-217, 2007.

[10] H. Naeem;, A. Masood', M. Hussain;, and S. A. Khan;, "A Novel Two-Staged Decision Support based Threat Evaluation and Weapon Assignment Algorithm, Asset-based Dynamic Weapon Scheduling using Artificial Intelligence Techniques," *CoRR,* vol. abs/0907.0067, 2009.

[11] A. Park; and R. Fujimoto;, "Optimistic Parallel Simulation over Public Resource-Computing Infrastructures and Desktop Grids," in *12th IEEE International Symposium on Distributed Simulation and Real Time Applications,* Vancouver, BC, Canada., 2008.

[12] D. R. Jefferson, "Virtual Time," *ACM Transactions on Programming Languages and Systems, V*ol. 7, pp. 404-425, 1985.