Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR-TP-2002-470

Performance of a component-based flight simulator architecture using the HLA paradigm

A.J.J. Lemmers, P.J. Kuiper and F.R. Verhage

Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR-TP-2002-470

Performance of a component-based flight simulator architecture using the HLA paradigm

A.J.J. Lemmers, P.J. Kuiper and F.R. Verhage

This report is based on a presentation held at the AIAA Modeling & Simulation Technologies Conference, Monterey, CA, USA, 5-8 August 2002.

Customer:	National Aerospace Laboratory NLR
Working Plan number:	V.1.d.2
Owner:	National Aerospace Laboratory NLR
Division:	Flight
Distribution:	Unlimited
Classification title:	Unclassified
	September 2002

Summary

Simulation components are the building blocks for simulators and can be used in more than one type of simulator. To enable cost-effective simulator development through reuse and exchange of simulator components, component and distributed simulation technologies can play an important role. Based on the principles of the High Level Architecture (HLA), a component-based simulation architecture for a training system has been proposed by Dutch Space, TNO-FEL and the National Aerospace Laboratory NLR. The federate is composed of various simulation components that interact using the same architecture and infrastructure as well as the overall HLA federation. Verification of the improved architecture performance has been performed by the NLR. The NLR Pilot Station has been restructured in accordance with the component-based concept and an experiment has been set up and carried out to measure the performance with an emphasis on latency issues. Comparison of the component-based simulation architecture with the performance of the validated NLR flight simulation facilities shows the capabilities.

Abbreviations

AIAA	American Institute of Aeronautics and Astronautics
CGF	Computer Generated Forces
CPU	Central Processing Unit
DIS	Distributed Interactive Simulation
DMSO	Defense Modeling and Simulation Office
DR	Dead-Reckon
FOM	Federation Object Model
FPS	Fighter Pilot Station
HLA	High Level Architecture
HMI	Human Machine Interface
Hz	Hertz
IG	Image Generator
I/O	Input/Output
JSA	JSF Simulation Architecture
JSF	Joint Strike Fighter
M&S	Modelling & Simulation
OTWV	Out of The Window View
PFD	Primary Flight Display
RAM	Random Access Memory
RCI	Run-time Communication Infrastructure
RPRFOM	Real-time Platform Reference FOM
RTI	Real-Time Infrastructure
SOM	Simulation Object Model

Contents

(23 pages in total)

This page is intentionally left blank

1 Introduction

Simulation components are the building blocks for simulators and can be used in more than one type of simulator. Examples of these components are: a dynamic model component, an Out-of-the-Window visual component or a flight controls component. These components can be developed by different companies or institutes, which all have defined their own application-specific interfaces. To enable cost-effective simulator development through reuse and exchange of simulator components, component and distributed simulation technologies can play an important role.

Based on the principles of the High Level Architecture, a component-based simulation architecture for a training system has been proposed by Dutch Space, TNO-FEL and the National Aerospace Laboratory NLR. The HLA concept has been extended to the level of simulation components. The federate is composed of various simulation components that interact using the same architecture and infrastructure as the overall HLA federation. The architecture is outlined in Figure 1. The potential capabilities of this component-based simulation architecture were demonstrated in a project in August 2000. Feedback on the demonstration revealed that latency is considered as a possible threat to the application range of a HLA-based component architecture. Therefore the next phase of the project focuses on performance improvement especially concerning the delay and latency issues. Verification of the improved architecture performance is performed by the NLR.

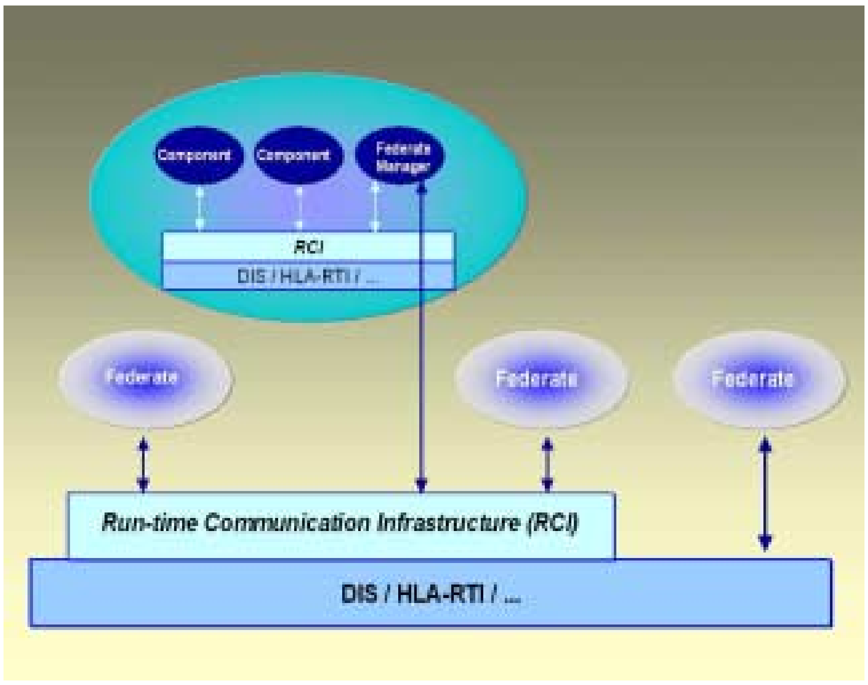

Figure 1: Component-based Architecture

Traditionally the NLR Flight Simulation Department is strong in high fidelity human-in-the-loop flight simulation. In recent years a high-performance network has been created between NLR's military flight simulations in combination with computer generated forces using ITEMS and a connection between this high performance network and "external" simulations has been established based on DIS and HLA. Comparison of the component-based simulation architecture with the performance of the validated NLR flight simulation facilities shows the capabilities. The NLR Pilot Station has been restructured in accordance with the component-based concept and an experiment has been set up and carried out to measure the performance with an emphasis on latency issues.

In this paper first the principles of the component-based architecture will be explained. It will be described how the Pilot Station has been divided into components. The capabilities of the Pilot Station components and their operational requirements will be discussed. Next the design and set-up of the validation tests will be outlined. The validation test will concentrate on three application domains: Human Machine Interface, Modelling & Simulation techniques and implementation-technical. Finally the results of these tests will be discussed and conclusions will be drawn to what extent the component-based concept is validated and in what areas it will have its advantages

2 Consistency in a disturbed environment

A major requirement for distributed simulations with real-time man-in-the-loop applications is to create a common and consistent presentation of the virtual environment among each federate and component. Due to message transmission delays, clock asynchronies among different simulation nodes and simulation time step of each node, together with the timeliness requirements of this kind of simulations, various inconsistencies may occur. This means that for example different participants may see the same entity located at different positions at the same (wall-clock) time. This phenomenon is called Time-Space Inconsistency and may cause inconsistent judgements among participants about the situation at wall-clock time t, what can lead to inconsistent actions. According to [Zhou01] and [Zhou02] these inconsistencies can cause the following problems in a distributive interactive application: Expectation Violation, Causal Order Violation and Divergence.

- Expectation Violation: After a participant takes an action, the participant expects the relevant outcome to be congruous with its real-life experience. This expectation can be so strong that violation of it will greatly degrade the application.
- Causality Violation: In real world, events happen according to their causal orders. Without the cause, the effect can never happen. Thus, the cause must happen before the event. If this

causal order is violated in the virtual world, participants will feel the virtual world incorrect and in contradiction with its real-life experience.

- Divergence: The same action may result in different interpretations at different sites.

Besides the transmission delay and clock asynchrony, time-space inconsistency is also dependent on some fundamental parameters of the simulation application, such as the threshold and average update rate of DR algorithm, the dynamic properties of the moving entities, and human factors like minimum discernible distance and human reaction time.

3 Evaluation aspects of application domains

Based on the time-space inconsistency problems discussed in the previous paragraph and the hardware aspects of the distributed implementation we can distinguish three different application domains which each have their own demands:

- Human Machine Interface (HMI) domain
- Modelling & Simulation (M&S) domain
- Technical Implementation domain

<u>HMI domain</u>

An essential part of flight simulation is the generation and display to the pilot in the simulator of a simulated perspective view of the outside world. The visual system receives position and attitude inputs from the dynamic model and must provide the out of the window view appropriate to each position and attitude. The same principle is valid for the avionics display except the fact that this display provides the pilot with a radar view and a Primary Flight Display (PFD). These displays stimulate the pilot to control inputs, and these control inputs lead to simulator responses, which will finally be represented in the displays. Delays in this closed loop must be as low as possible and are critical to the controllability of the simulated aircraft as given in [Rolfe86].

The following elements are identified for this domain.

- controls to visual latency
- controls to avionics display latency
- difference in latency between avionics display and visual

<u>M&S domain</u>

Within a distributed simulation environment the effects of data transmission delays and data losses can have a major impact on the differences between the true and perceived positions of the distributed simulation entities [Smith98]. This can be especially a problem for high performance entities interacting with each other (e.g. aircraft/munition). From this modelling and simulation point of view for interactions the following elements are of importance:

- the latency of remote entities (from a CGF) to visual (for targeting)
- the latency of remote entities to avionics (for detection by avionics systems)
- the latency of actions (target lock, hit etceteras) to remote entities

Technical Implementation domain

Loads imposed on the network and on the CPU in general depend on scenario, (dead-reckon) algorithms and (communication) protocols. Network loading is based on increasing numbers of digital entities and their performance characteristics to investigate scalability. Recent years many research projects were performed to investigate the network load and to predict the bandwidth and CPU-load requirements for both configurations based on the DIS protocol and HLA/RTI based configurations ([Purdy98], [Menzler99], [Bertin01], [Givens01]). With the results of these projects and a baseline of latency and performance characteristics of the simulation architecture it will be possible to identify requirements for network and CPU capacity and predict the simulation performance. From this implementation/technical point of view the following elements can be a constraint depending on the resources available:

- the I/O throughput (network load)
- the calculation need (CPU load)
- the memory size (memory use)

For all of these elements not only the (average) value is important but the minimum, maximum and variance also have a major influence on the usability.

4 Fighter Pilot Station set-up

The Fighter Pilot Station (FPS) used for the project consists of five relevant major software components:

- Controls input component which acquires pilot inputs (actions)
- Aircraft dynamics component which calculates the aircraft movements
- CGF component (Computer Generated Forces) which simulates “other” aircraft
- Avionics component which incorporates the instruments and display logic
- Visual component which drives the visual system

Both the FPS (undivided) as the FPS based on components have the same basic data flow. There is no difference in connection with the hardware systems (controls, visual and avionics display). With the undivided FPS the data is distributed among the software modules on one host computer through Global Simulation Data which resembles shared memory. Also the software modules are triggered sequentially within the real-time simulation program PROSIM. Because of the fast shared memory communication the whole process is scheduled real-time at a high frequency of 50 Hz.

For the component-based set-up the software components are not triggered sequentially but run independently of each other and they have their own frequency (free running). Each component is scheduled real-time within its own simulation program PROSIM. To communicate between the software components on the separate host computers the component based FPS uses a middle-ware layer based on HLA, called the Run-time Communication Infrastructure (RCI). The RCI currently uses DMSO's RTI through Ethernet.

There is no difference in connection with the hardware systems (controls, visual and avionics display).

Each software component produces data and/or uses data from other components as indicated by the numbered arrows in Figure 2 and Figure 3. The data elements that are distributed among the components are:

1) Controls inputs divided into:
   a) Stick and throttle position (used by the aircraft dynamics)
   b) Switch positions (used by the avionics system)
2) The ownship Aircraft State (position, orientation, appearance) used by the visual module for eye-point movement, by the CGF module for interaction with other aircraft and by the avionics module for display indications.
3) The states of "other" aircraft (position, orientation, appearance) used by the visual module for the movement of visual models in the outside world scene and by the avionics module for display indications (radar display).

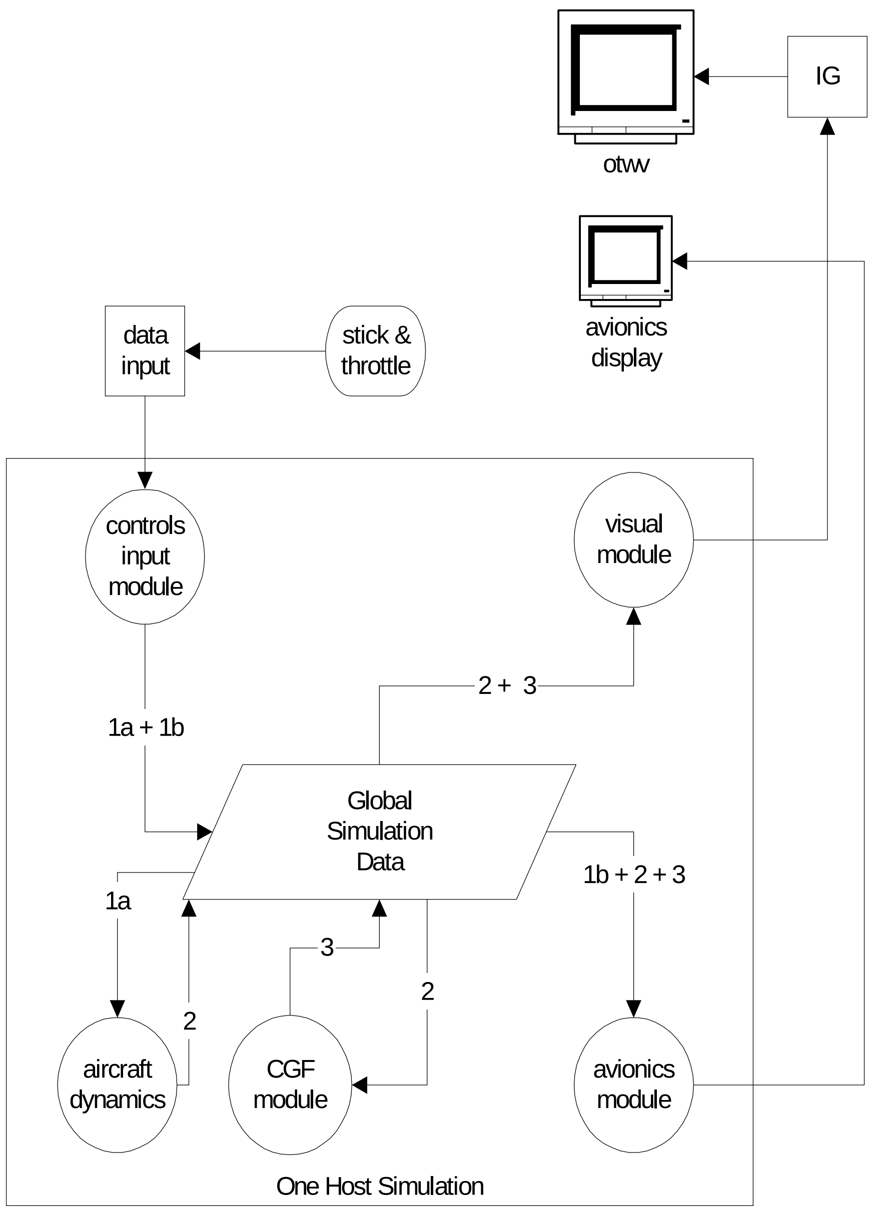

Figure 2: Models in Fighter Pilot Station, undivided set-up

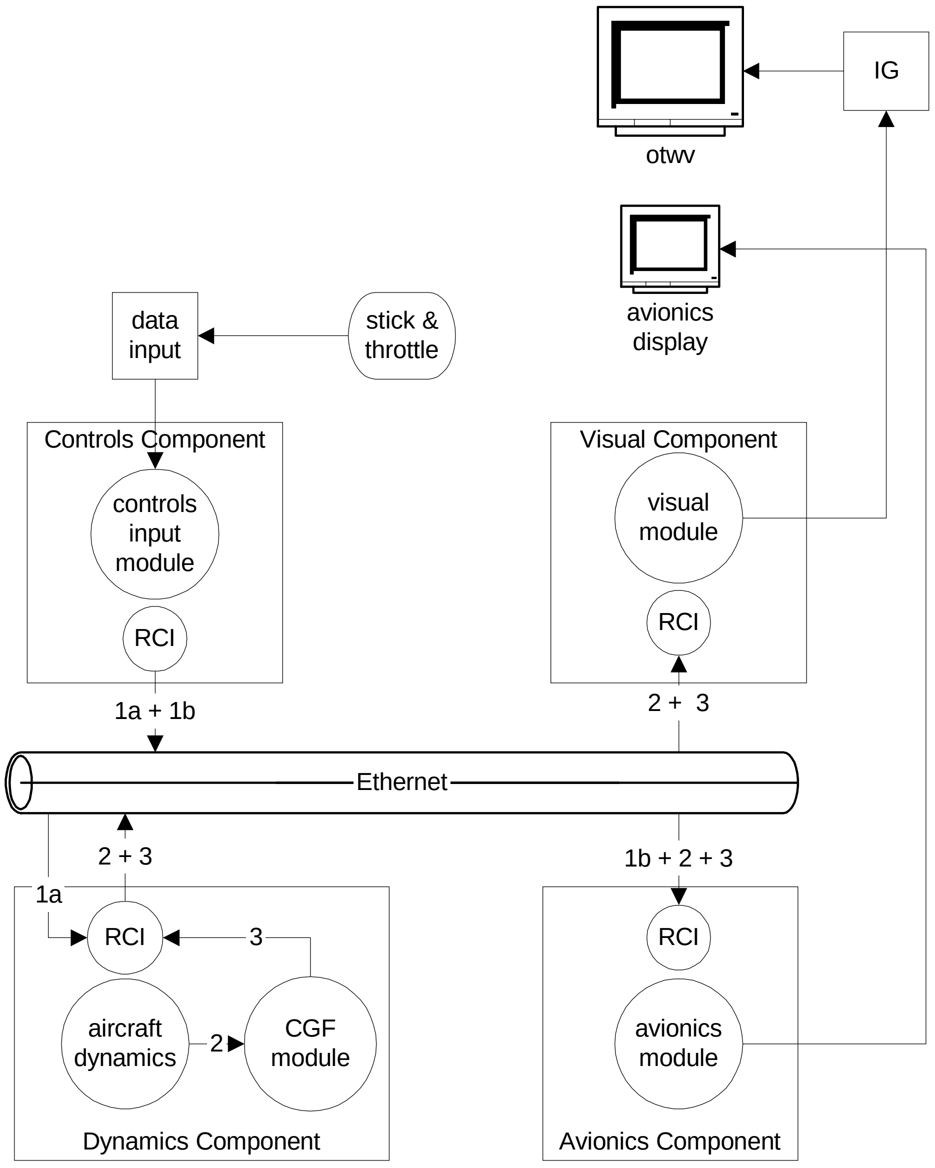

Figure 3: Models in Fighter Pilot Station, Component-based set-up

5 Method

The latencies can be divided into the following categories:

- Component hardware to software latency or vice versa
- Processing time (CPU time to process the data)
- Software module to software module latency

The first category is the same for both the undivided FPS as for the component based FPS because they both use the same hardware connections and are not influenced by the dividing of FPS into components. These hardware latencies have been analysed and measured in the past [Kuiper00] and therefore these have not been measured again.

The average processing time is determined by multiplying the average processor load (%) by the cycle time (milliseconds). The average memory load is measured and no-growth is monitored using the system tool *top*.

The last category (s/w to s/w latency) is the essential point in a component-based set-up. This is measured by sending a timestamp together with the data flow and calculating the difference in time between this time stamp and the time this data is received. Also the time stamp increase with each sample is measured as an indication of the smoothness of the data.

A counter is used for the controls data flow to detect lost updates or out of order updates. The latency of this data is influenced by the network throughput and load. It is not possible to determine the momentary network load, only the average network load is measured (in packets/s) with system tools such as *top* and *netstat*.

6 Configuration

All components have been run on IRIX 6.5 using DMSO's RTIv1.3NG3.2 and TNO's RCIv1.0beta2. The computer and network hardware used is summarised in the following table:

Table 1: Computer system specification

Federate	Controls & Visual	Aircraft Dynamics	Avionics
Computer type	SGI Origin	SGI Challenge	SGI O2
Processor	4 x R12K	4 x R4400	R10K
Processor speed (MHz)	270	100	225
Memory (MB)	512	64	64
OS	IRIX 6.5.6	IRIX 6.5.5	IRIX 6.5.5

The data that has been exchanged between the components is defined in a Federation Object Model (FOM). This FOM is based on the RPR FOM v1.0 Draft 2. The Control Component SOM was added for specific Stick and Throttle objects and interactions.

For the simplicity of the simulation model code the object attributes were updated every simulation cycle, they were not dead-reckoned nor were changes detected to send changes only. This was also preferred to have a as much of a known steady load during measurements and not a varying load. A Stick and Throttle update from the Controls Component resulted in 15 attribute updates each simulation cycle. For the Aircraft state update by the Aircraft Dynamics Component 37 attribute updates are done each simulation cycle. It is important to realise that this can lead to a higher load than strictly needed.

7 Experiment execution

The experiment has been executed with a NLR employee with simulator experience as pilot. The pilot has flown a square pattern in the runs. On each leg of the square he has performed one of the following manoeuvres: S-turn, pop-up, 4 points roll or 360 with 30 degrees roll angle.

The runs have been carried out with all components running at the same frequency of 10, 25, 40 and 50 Hz respectively. The number of players has been varied during the experiment. Several runs have been executed with the ownship only, while in other runs upto 5 CGF players have been added.

8 Experiment results

During the experiment the amount of memory used by the RTI/RCI communication was high and also the CPU load for the RTI/RCI was heavy for the computer available as indicated in figure 4. This system was stable only if a strict start-up sequence (one by one) for the federates was followed. The communication process was approximately 160 MB large in memory. With some computers having only 64 MB of main memory this had a large impact on the start-up time. This was 2.5 minutes for the Avionics Component (64 MB RAM), 1.5 minutes for the Aircraft Dynamics Component (64 MB RAM) and 10 seconds for the Controls and Visual Component (512 MB RAM). This resulted in a total start-up time of more than 5 minutes.

Overall the amount of memory was sufficient to run fast enough most of the time. But as soon as some extra “work” had to be done the memory shortage led to memory swapping which worsened the situation and from which recovery usually became very difficult if not impossible.
The network bandwidth (10 Mbit/s) on the other hand was not a bottleneck. The total network load was below 200 packets per second. Assuming “full” packets of 1500 bytes this is approximately 2.3 Mbits/s which is 23 % of the bandwidth.
The CPU load of the HLA communication process is given in figure 4.

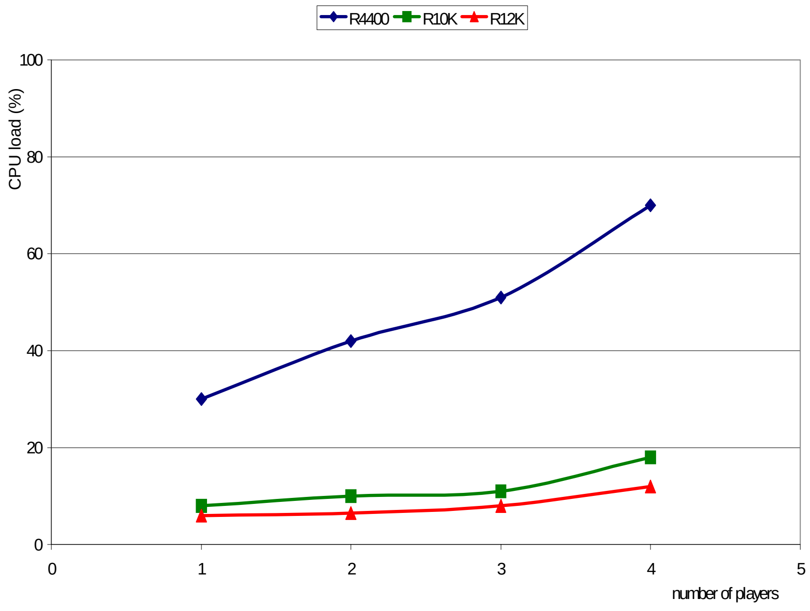

Figure 4: CPU load on different processors

During the experiment it was found that each attribute value update has a (CPU) overhead [Murray00]. Performance wise it would be better to group the attributes that are always sent together in one complex data type. The 15 attribute updates each simulation cycle for a Stick and Throttle update from the Controls Component, could be reduced to two attribute updates (one for each object).
The 37 attribute updates each simulation cycle, for the Aircraft state update by the Aircraft Dynamics Component, could be reduced to one for positional and orientation data together with an occasional attribute update for the others.
In the next version of RPR-FOM (v2) the positional and orientation attributes have already been grouped together (spatial attribute). The other attributes are intended to be sent on change basis only.

The amount of time the HLA communication process needed each simulation cycle varied as can be seen in figure 5, even though the process was given real-time priority.

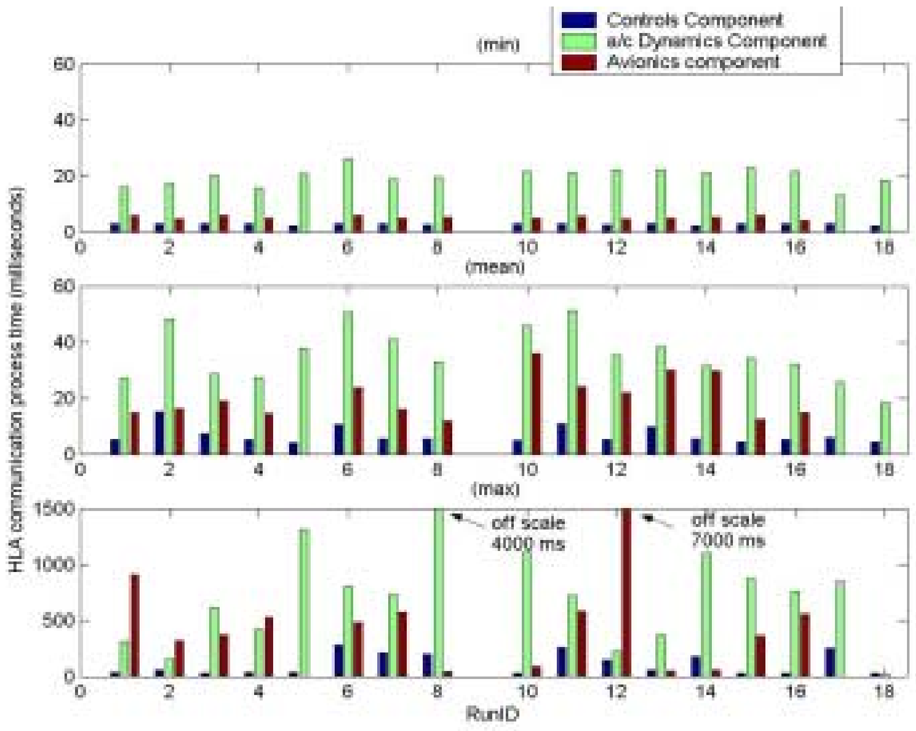

Figure 5: HLA communication time per simulation cycle

The average HLA communication time per simulation cycle was 5-10 milliseconds for the Controls component, 15-30 milliseconds for the Avionics component and 27-50 milliseconds for the Aircraft dynamics component. The computer on which the Aircraft dynamics component ran was indeed the slowest of the three and the average time per simulation cycle for the HLA communication process limits the simulation frequency to 20-40 Hertz. Besides the average time the maximum time and the variation are important for real-time applications. In the experiment set-up the federates did not wait for the HLA communications process to complete when this took longer than the simulation cycle, but kept its own real-time pace. This should be considered as an undesired situation. The number of times this took place is logged as the "communication not ready" counter in figure 6.

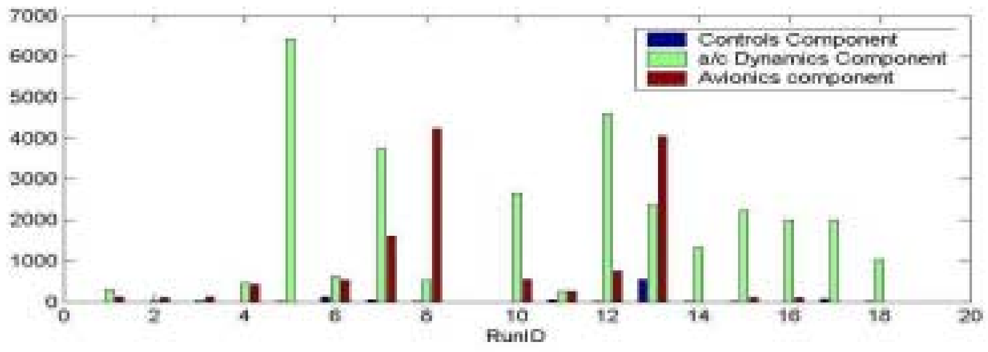

Figure 6: HLA communication not ready counter

Due to the occurrence of "communication not ready" and due to the fact that the federates run independently, the updates for Stick&Throttle and Aircraft state data do not arrive at fixed intervals, but vary. This can be seen in the time difference or interval between between aircraft state updates for the Visual component shown in the figure 7. The order in which the state updates arrived was the same order as they were sent out.

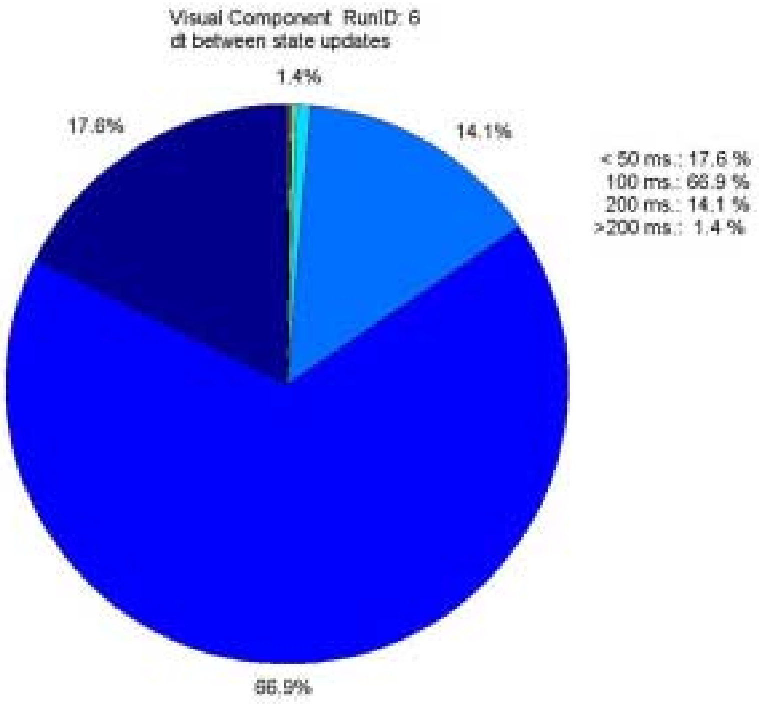

Figure 7: Time difference between state updates Visual component

The variation in time interval of Stick&Throttle and Aircraft State data update has an effect on the form of the data signals as seen in figure 8, pointed out by the arrows. Although this effect seems small, it has a non-deterministic effect on responses of for example accurate control systems. Also the “glitches” in pitch and roll can be seen on the visual as “hiccups”.

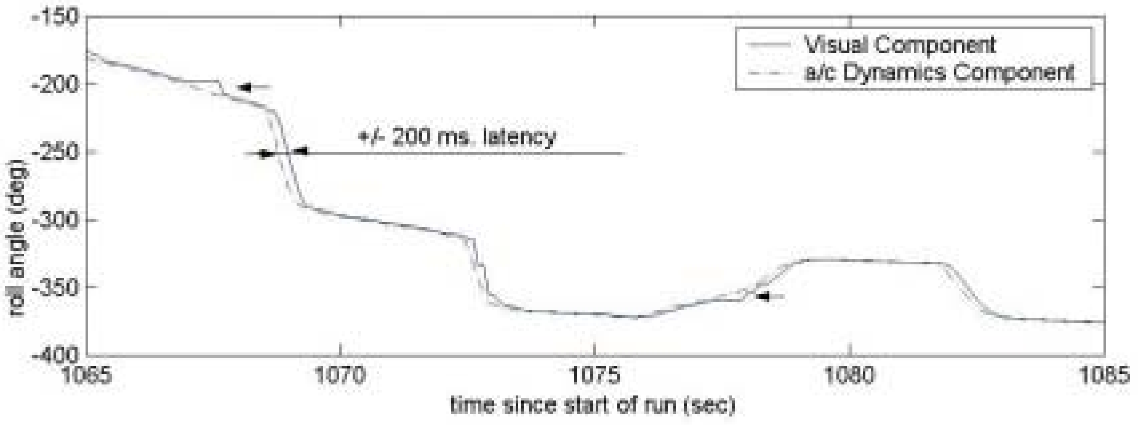

Figure 8: Roll angle in Dynamics and Visual component

9 Comparison of results with undivided FPS

All the measured latencies and variances in latency in this experiment are due to the components being distributed on separate computers. In the case of an undivided FPS, with all software modules running on one central computer, the latency or variance in latency is always less than one simulation cycle.

The measured latencies come on top of the component – hardware latencies (data acquisition hardware, out of the window display hardware etcetera). These component-hardware latencies are the same in both the undivided FPS and the divided FPS (this experiment set-up) and depending on the hardware system vary between one simulation cycle and 70 milliseconds [Kuiper00]. The total latencies are expressed in table 2. Combining the given latencies of the undivided FPS with the measured latencies during the experiment gives the total latency for the JSA set-up and is given in table 3.

The flexibility of independently developed components is paid for in added latency and variance in latency. Also expertise and time is needed to monitor the latency and often to tune the set-up to keep the latency as small as possible. Also the consistency must be looked at closely with each set-up. Especially with more complicated set-ups the prevention of expectation and causality violation or divergence needs extra attention.

For rapid prototyping and functional tests the advantages of a component-based set-up outweigh the trade-off in latency. But if deterministic low-latency is needed every added latency is to be avoided. The needed total Stick to Visual latency of 100 milliseconds (maximum) for high fidelity flight simulators is already hard to achieve.

The RCI also has the possibility to schedule components. This could eliminate a large part of the variance in latency that is inherent to free-running components as used in this experiment. The gain is dependent on the accuracy of this distributed scheduling mechanism. This can be an interesting and promising area for further research.

Table 2: Total Stick to Visual latency for Fighter Pilot station, undivided set-up

	Latency @ 10 Hz (milliseconds)			Latency @ 25 Hz (milliseconds)			Latency @ 50 Hz (milliseconds)		
	min	mean	max	min	Mean	max	min	mean	Max
1. Data input time	2	52	102	2	22	42	2	12	22
2. Controls input module processing time	100 (together less than one simulation cycle)	100	100	40	40	40	20	20	20
3. Controls to Dynamics transmit time									
4. Dynamics module processing time									
5. Dynamics to Visual transmit time									
6. Visual module processing time									
7. Visual system display time	49.99	58.3	66.7	49.99	58.3	66.7	49.99	58.3	66.7
Total Stick to Visual latency	152	210.3	268.7	92	120.3	148.7	72	90.3	108.7

Table 3: Total Stick to Visual latency Fighter Pilot station, JSA set-up

	Latency @ 10 Hz (milliseconds)			Latency @ 25 Hz (milliseconds)			Latency @ 50 Hz (milliseconds)		
	min	Mean	max	min	Mean	max	min	Mean	max
1. Data input time	2	52	102	2	22	42	2	12	2 2
2. Controls input module processing time	1	1	1	1	1	1	-	-	-
3. Controls to Dynamics transmit time	10	200	>400	10	100	>200	-	-	-
4. Dynamics module processing time	100	100	>100	40	40	>40	-	-	-
5. Dynamics to Visual transmit time	30	100	>200	30	20	>40	-	-	-
6. Visual module processing time	1	1	1	1	1	1	-	-	-
7. Visual system display time	49.99	58.3	66.7	49.99	58.3	66.7	49.99	58.3	66.7
Total	194	512	>870	134	242	>390	-	-	-

10 Conclusions and lessons learned

Due to the large size in memory of the HLA communication process (RTI/RCI, 160 MB on IRIX) it is strongly recommended to use computers with at least 256-512 MB of memory. The HLA communication process is also CPU hungry, lots of CPU power on all the participation systems is necessary to prevent it from creating a bottle neck. The use of older computer systems is therefore strongly discouraged.

Most of the latency build up and peaks as seen in the graphs could presumably be avoided by using more powerful computers (CPU and memory) and lowering the load by cutting down the number of attribute updates each simulation cycle. This could also give more room to increase the simulation frequency, which in turn has a positive effect on the latency (lower average), but on the other hand would lead to again a higher load.

The CPU load of the HLA communication process limited the simulation frequency to 25 Hz and the number of players to 4. Higher simulation frequencies are possible by using more powerful computers and by combining attributes to complex data types in the SOM/FOM. The mean latency can be lowered by rearranging the read and write order in the HLA communication process.

A large part of the added latency and variance in latency can be traced back to the fact that the components ran independently with a fixed simulation frequency. If the components could be triggered sequentially, this could be a large improvement. This trigger mechanism would need to be accurate and deterministic to get good results.

References

1. [Bertin01]
   Ronald Bertin, paper 01S-SIW-096, *Bandwidth, Latency, Jitter, and Synchronization: A High Fidelity Simulation Perspective*, Simulation Interoperability Workshop, Orlando, March 2001
2. [Givens01]
   Bret Givens et.al., *Lessons Learned in Virtual Strike Warfare Environment (VSWE) 7*, paper AIAA 2001-436, AIAA Modeling & Simulation Technology conference, Montréal, Canada, August 2001
3. [Kuiper00]
   Paul J. Kuiper and Arjan J.J. Lemmers, *Opening up full-mission flight simulation for networked simulation exercises; experiences with advanced distributed simulation,* AIAA 2000-4398, AIAA Modeling & Simulation Technologies conference, Denver, August 2000, NLR-TP-2001-357
4. [Menzler99]
   Hans-Peter Menzler, Hartmut Ufer, *PDU for DIS versus HLA-RTI: A Load Perspective*, paper 99S-SIW-076, Simulation Interoperability Workshop, Orlando, March 1999
5. [Murray00]
   Bob Murray and Adam Raier, *Defining FOMs for Performance*, paper 00F-SIW-116, Simulation Interoperability Workshop, Orlando, September 2000
6. [Purdy98]
   Lt. Stephen Purdy and Roger Wuerfel, *A Comparison of HLA and DIS Real-time Performance*, paper 98S-SIW-042, Simulation Interoperability Workshop, Orlando, March 1998
7. [Rolfe86]
   J.M. Rolfe and K.J. Staples, Flight Simulation, Cambridge Aerospace Series, 1986, Cambridge, UK
8. [Smith98]
   N. Smith, *Report on Architectures for Simulation Interoperability involving High Performance Flight Simulations (UC)*, DERA/AS/FMC/CR980386/1.0, August 1998
9. [Zhou01]
   Suiping Zhou, Wentong Cai, Francis B.S. Lee, Stephen J. Turner, *Consistency in Distributed Interactive Applications*, paper 01E-SIW-003, Simulation Interoperability Workshop, Harrow, UK, June 2001
10. [Zhou02]
    Suiping Zhou, Wentong Cai, Francis B.S. Lee, Stephen J. Truner, *Time-Space Consistency in Large Scale Distributed Virtual Environment*, School of Computer Engineering, Nanyang Technological University, Singapore

    http://www.ntu.edu.sg/home/asspzhou/ts.ps