Abstract—In developing multi-UAV(Unmanned Aerial Vehicle) system, a simulation environment is essential to verify the functionalities of the whole system with higher productivity and reduced risks of accidents. Simulations of multi-UAV systems are usually accomplished by using distributed systems where multiple standalone simulators are connected via a network. Thus, in order to achieve an improved reality in such distributed simulation environments, the whole multi-UAV simulator should be guaranteed real-time and synchronized behaviors among the connected systems. In this paper, we discuss about the necessity of two important real-time features, i.e., the periodicity of computations and the clock synchronization among standalone simulators. We also propose an architecture based on TMO(Time-triggered and Message-triggered Object) model as a multi-UAV simulation platform with our preliminary experimental results showing its potential feasibilities.

Keywords- Multiple UAV simulation; TMO HILS; Periodicity guarantee; Global time synchronization;

I. INTRODUCTION

The flight control system for an unmanned aircraft is very complex and hard-to-develop system because it has to integrate fundamental functions for such as control, navigation, communication and etc. Because those aircraft system has extreme risk and financial loss by the system fault, a lot of financial and timing costs to develop and test are required. Most absolute way to test the control system is to embed the system on the real fuselage and run it actually, but it is too dangerous to operate unreliable system that is not fully verified. Thus the testing process using simulation is essential to reduce development cost.

The simulation technique, named HILS (Hardware-In-the-Loop Simulation) is usually used for development and verification of the UAV systems. The HILS is an environment that simulates the motion of target fuselage by using inputted signal from target control system, hence the reliability and productivity could be increased [3][4][6][7].

Recently, as well as the single UAV, multi-UAV that a number of systems interact and cooperate is issued [9][10][11][12]. Typical multi-UAV simulation system has a form of a distributed computing structure that consists of multiple simulators and one monitoring system connected via a network link. Each simulator simulates one target system, and every simulation result is synchronized and mashed-up onto one virtual environment to let engineers test the whole multi-UAV system with reality[1][2][9][10].

Thus, in order to achieve an improved reality in such distributed simulation environments, the whole multi-UAV simulator should be guaranteed real-time and synchronized behaviors among the connected systems. In this paper, we discuss about the necessity of two important real-time features, i.e., the periodicity of computations and the clock synchronization among standalone simulators. We also propose an architecture based on TMO(Time-triggered and Message-triggered Object) model as a multi-UAV simulation platform with our preliminary experimental results showing its potential feasibilities.

In the next chapter, we introduce the simulation system for the multi-UAV. Then, real-time issues for the reality of simulation are discussed in the chapter 3. The chapter 4 is about detailed design and implementation of the multi-UAV simulation system, named TMO HILS. The measurement of real time performance of the TMO HILS is shown in the chapter 5, and finally we conclude and state our future works.

II. SIMULATION FOR MULTI-AGENT UAVS

2.1. HILS and Dynamics model

In developing hard-to-test systems such as missile, satellite, and unmanned aircraft, dynamic characteristics of fuselage of the system is mathematically analyzed and simulation model is derived. In the HILS, a computer receives control signals from a real target control system and executes the mathematically derived model of dynamics. The Fig. 1 describes the concept of the HILS.

The model of dynamics embedded in the HIL simulator contains equations that describe motions of a system as a function of time. This equation of motions is derived by physical and dynamic analysis of influence of a force.

The HIL simulator periodically receives control signal from the target system, inputs the received data into model, operates the equation of motion, and feed the output data to the target system. This loop is kept during simulation operation.

978-0-7695-4038-2/10 $26.00 © 2010 IEEE
DOI 10.1109/ISORCW.2010.23
2.2. Earlier multiple simulation researches

Many of existing researches related to the HILS are mainly focused on verifying the single control system's stability, analyzing dynamics, and optimizing the equation of motion. Recently, many efforts have been conducted to apply HILS to multi-UAV systems as well as single UAV system[9][10]. Because of the sophisticated structure and high cost of development process in the multi-UAV systems, the HILS for multi-UAV systems became necessary for efficient developing processes and the reliable implementations.

Earlier researches since 1990’s in multi-simulation were tightly related with distributed computing field because the multi-simulation basically consists of several computers operating concurrently [8]. Many multi-simulations has been structured as distributed computing system where several simulation computers are connected via network and data are simply collected into one node [9][10]. Fig. 2 describes concept of the typical multi-HILS environment for multi-simulations.

The multi-simulation, however, has some inherent issues related to the timeliness among the engaged systems.

III. REAL-TIME ISSUES IN MULTI SIMULATION

There are two important real-time issues in the multi-HILS system; 1) guaranteeing the steady periodicity of operations of the equations of motions, and 2) global synchronization among the engaged systems.

3.1. Periodicity of computations for simulation

Equations of a motion, the core logics of simulation, are basically ordinary differential equations. Thus operations of the HIL simulator should be executed at a steady cycle to be same with theoretical time period. If a series of process of the HIL simulator is not timely completed, the results may not be fed back to target system at expected timing, and this causes a low quality of reality in the simulation. This is the reason that most of the HIL simulators is based on the real-time systems [4][5].

Fig. 3 illustrates a scenario of timely ideal reactions of a simulation. Both of the signal inputted at 20 milliseconds and the result calculated at 10 milliseconds are inputted into simulation model and a new simulation result is calculated at about 20 milliseconds. The equations should be given new control values and previous simulation results as input parameters.

Fig. 4 shows the HIL simulator's scenario that doesn't guarantee real time. Non-real-time HIL simulator causes a
drift of the completion time, thus its simulation is not the same with ideal model and the reality is disturbed.

3.2. Global time synchronization among concurrent operations

In multi-simulation environments, every calculation in each computer is expected to be operated at same time and with same cycle. But once the timing is not properly synchronized in real time, reality of the relationship between simulators would become unreliable. Therefore every temporal factors of the simulation should be properly controlled, and synchronization of the global time is fundamental requirement. Following paragraph describes the reason that the global time synchronization is essential.

The Fig. 5 describes a scenario of synchronizations between results from two simulators. (a) Simulator #1 and simulator #2 are expected to share the same global time and to execute simulations at the same cycle. Although those operations are completed at the same time, some delay could occur during transmission due to unexpected network jitters. The scenario (b) presents delayed transmission where deadline is not violated. The scenario (c) shows that the deadline is violated due to network jitter and two simulation results are not properly synchronized at the same time. This fault causes inaccuracy of the simulation and also unreliability in simulated target systems which share one virtual simulation space. In other words, it could bring out errors, for example, in computing conflict avoidance among multiple aircrafts. However, if the global time synchronization is not guaranteed, then some of conflicts could be missed.

There were some efforts to meet the synchronization issue with respect to the network scheduling and control [8]. The network scheduling is one of approaches to control packet transmission for timely synchronization. And there have been a lot of methodology about distributed computing for the past several decades. However, unfortunately, establishing the distributed system still requires too many efforts of designers.

In the next chapter, we introduce an object-oriented distributed real-time scheme to overcome above issue, and also previously explained multi-simulation architecture using this scheme, too.

3.3. TMO scheme and RMMC

As we discussed formerly, the system architecture of the multi-HILS based on distributed real-time object model named TMO (Time-triggered Message-triggered Object), was previously proposed to construct reliable real-time multiple simulation system. The TMO is object-oriented programming scheme to reduce complex design efforts and improve the productivity of the development of the real time systems. Using this scheme, the designer can save the cost of design to guarantee the services of the networked and time coordinated systems effectively [13][14].

The TM0 contains two types of methods, time triggered (TT) methods (spontaneous methods or SpMs), which are clearly separated from the conventional service methods (SvMs). The SpM executions are triggered whenever the RT clock reaches time values determined at the design time. On the contrary, SvM executions are triggered by calls from clients that are transmitted by the execution engine in the form of service request messages.

The SpM time properties are defined in AAC (Autonomous Activation Condition) which allows designing timely behavior of SpM easily. The AAC parameters consists of "From"(start time), "Until"(end time), "Every"(cycle of execution), "EST"(Earliest Start Time), “LST” (Least Start Time), and “By” (deadline of SpM execution). Once designer just define SpM(s) with the AAC and SvM(s), every methods would operate respecting the time constraints if there is not any violation of timeliness.

To offer great flexibility in concurrency control, there is shared data container called the object data Store (ODS). The ODS segment (ODSS) is a basic unit of data storage, a group of variables, that can be reserved for exclusive access by a real-time object method [21]. The variables grouped into ODSS are locked for exclusive use by a TMO method in execution. Scheduling and access control rule, named BCC (Basic Concurrency Constraint), protect mutual exclusion and prevent potential conflicts between SpMs and SvMs.

Basically, because the TMO scheme is designed to provide convenience of designing distributed systems, there is a gate section for distributed computing, named RMMC (Real-time Multicast and Memory-replication Channel). Multiple TMO objects interact via remote method calls using the RMMC [15].

It is regarded as all TMO objects initially share same global time, and every remote call and data transmission
between TMO objects are especially accomplished with a time constraint called ORT (Official Release Time). The ORT is a time to judge whether the messages or remote calls are officially acceptable state or not. By synchronizing the global time, the ORT could be reliably accepted by other distributed object. Although some messages (or remote call request) are transmitted to the other objects, the receiver side can only accept the messages when the message’s ORT is passed. By this timing control, multiple distributed objects could be timely synchronized at accurate timing.

3.4. Review of TMO HILS architecture

Applying above features of TMO, we previously proposed TMO HILS architecture for multiple simulations supporting multi-agent system [1][2]. As described in Fig. 6, the architecture consists of a number of HIL simulators which are simulating each target control system, virtual environment server, and a visualization tool. Although extra financial cost is proportionally increased to expand the number of target system, we allocate one simulator per one control system because of following reasons.

1) MIMO (Multi-Input Multi-Output)

Basically, the aircraft control system requires many I/O ports for communication with various devices such as sensors, actuators, and wireless networking modules. Hence there is a limitation to connect many signal port of the control systems to single unified simulator.

2) Expandability of multiple simulators

Although one simulator can have enough ability and resources to operate multiple simulations, the number of simulations is limited within the allowed computing power. Therefore, a flexible expansion by using additional computers is required, as well as increasing real-time tasks in single computer is necessary. In order to achieve these two contradictory goals simultaneously, a network-centric structure could be useful.

On the other hand, guaranteeing both of real time synchronization in distributed system and flexible expandability is a very sophisticated issue. Designer should consider many factors which might affect on real-time violation and scalable design. We believe that the TMO scheme is one of effective approaches to address this issue. The detailed design and implementation are explained in the next chapter.

IV. DESIGN AND IMPLEMENTATION

4.1. Object design in TMO HILS

The previously proposed an object design for the TMO HILS is illustrated in Fig. 7[1]. EnvTMO object is to describe a virtual environment of a simulation, which defines environmental factors for target objects. Virtual environment data, for example, virtual ground and air environment, are common information that is shared by every simulation object. The Update_Env_SpM object periodically broadcasts the change of environment data to each SimTMO object. The SimTMO object composed of some SpMs and RMMC gate for data synchronization executes main real-time tasks in the HIL simulator. The ReadFromCS_SpM is a task to analyze the control signal from a target control system and to save the data into the SimODSS which is a shared storage for SpMs and SvMs within the SimTMO object. The Dynamics_SpM uses an updated virtual environment data
and control signal as parameters for the equations of motion. After calculation of equations, the result data are broadcasted to the EnvTMO and SimMonitorTMO through the SimRMMC, and each transmitted data is stored in the message queue within the EnvRMMC and SimMonitorRMMC.

4.2. Real-time task scheduling design for HIL simulator

As discussed in chapter 3.1, the periodicity, which is one of the most important goals of real-time scheduling, is deeply related to the reliability of simulations because the execution cycle of tasks to calculate the equations of motion should be maintained steadily.

The HIL simulator should complete within deadlines for every real-time task for such as calculating equations of motion, accepting control signal, and synchronizing with other objects. In order to design real-time task scheduling for target HILS, the measurements of execution time for each task are needed. For our experiments, we measured the execution times of two major tasks; 1) task for accepting control signal, and 2) task for calculating equations of motion. The platform used for our experiments consists of a 2.13GHz dual core CPU and 2Gbytes-sized main memory as hardware, and Windows XP as OS. Following is measurement of average execution time of each task.

1) Task for accepting control signal (testing 1,000 times)
- Average execution time: 234 µsec
- Maximum execution time: 1457 µsec
- Minimum execution time: 162 µsec

2) Task for Calculating equations of motion and feeding back (testing 1,000 times)
- Average execution time: 400 µsec
- Maximum execution time: 478 µsec
- Minimum execution time: 243 µsec

According to above measurement, we defined the AAC parameters of the ReadFromCS_SpM and Dynamics_SpM for real-time guarantee as shown TABLE I. The ReadFromCS_SpM and the Dynamics_SpM are designed to be executed at every 20 milliseconds steadily. The MonitorSimTMO_SpM in SimMonitorTMO object is designed to accept simulation results from each simulator and analyze those data to detect conflict at every 20 milliseconds. And the Export_result_SpM periodically sends the results to an external visualization tool. In order to serialize execution sequences of SpM, Dynamics_SpM's "From" parameter has been tuned to 5 milliseconds later than ReadFromCS_SpM. The execution time is set as under 2 milliseconds, and each deadline of SpM is set as 5 milliseconds by considering above measurement records.

4.3. Synchronization among distributed objects

The Fig. 8 illustrates a scenario of synchronization among distributed objects of our TMO HILS for the multi-UAV simulation. One of messages transmitted through RMMC gate is the “SimResult” message containing a simulation result data, and another is the “VirtualEnv” message containing a virtual environment data.
Each SimResult message has been designed to be synchronized at 50Hz (every 20 milliseconds as Dynamics_SpM’s execution cycle), and the ORT (Official Release Time) has been set as 20milliseconds. The ORT is to properly synchronize every message although some packets are delayed due to network jitters or simulator violates real time guarantee, hence the receiver object would wait for delayed packets fairly (Ref. chapter 3.3). The decision of ORT boundary will be explained later. In every RMMC gate, there are message queues to save transmitted message, and only ORT-released messages are allowed to be accessed by a SpM.

The MonitorSimResult_SpM accepts simulation results from each SimTMO object and judges whether there is any risk of conflict between virtual objects representing UAV’s. ORT of the VirtualEnv message has been set as 0 to be released as quickly as possible.

V. EXPERIMENTS

5.1. Experiment environment

To measure the accuracy of message synchronization and execution periodicity in TMO HILS, we constructed a multi-simulation system that consists of four simulators. The hardware platform specification is listed in TABLE II, and we used TMO toolkit XP v4.2.3 which is a TMO support middleware developed by Dream Lab, UC Irvine [20].

5.2. Periodicity of HIL simulator operation

The real-time tasks of HIL simulator are implemented as SpMs which is periodically triggered. The TABLE III lists execution time measurements of the start time and end time of ReadFromCS_SpM and Dynamics_SpM. By the AAC configuration in TABLE I, two SpMs are expected to be alternately executed at every 20 milliseconds.

The Fig. 9 graphically shows the periods of two SpMs. The Y axis is a time difference between start time and end time of SpM. Each average period of ReadFromCS_SpM and Dynamics_SpM by testing 1,000 times is 20.00367 milliseconds and 19.99959 milliseconds. Namely the most of execution cycles are near 20 milliseconds steadily. This means that the accuracy of periodicity of our TMO HIL simulator is near enough to the designed simulation model. Hence, this allows the simulator system more reliable with high reality.

5.3. Global Synchronization

The TMO HILS implemented for our experiments consists of four simulators and one TMO HILS Server. Fig. 10 illustrates scenario of synchronization of SimResult messages from each SimTMO. Every TMO objects are expected to share the same global time and simultaneously started at the same time. Each SimTMO starts to send messages through the SimMonitorRMMC at the same period. The message transmitted at time T has ORT value (T+20msec), hence the received message in the SimMonitorRMMC is accessible after the ORT is passed.

We measured actual execution time of SpMs in the implemented TMO HILS to confirm the above scenario. The TABLE IV lists message transmission time and index of

<p>| TABLE I. AAC PARAMETERS OF SPMS IN SIMTMO |  |</p>
<table>
<thead>
<tr>
<th>AAC parameters</th>
<th>ReadFromCS_SpM</th>
<th>Dynamics_SpM</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td>0sec</td>
<td>5msec</td>
</tr>
<tr>
<td>Until</td>
<td>60min</td>
<td>60min</td>
</tr>
<tr>
<td>Every</td>
<td>20msec</td>
<td>20msec</td>
</tr>
<tr>
<td>LST</td>
<td>5msec</td>
<td>5msec</td>
</tr>
<tr>
<td>By</td>
<td>5msec</td>
<td>5msec</td>
</tr>
</tbody>
</table>

| TABLE II. HARDWARE SPECIFICATION OF HIL SIMULATOR |  |
| CPU | Intel Core 2 Duo 6600 |
| Main Memory | 1 Gbytes |
| OS | Microsoft Windows XP professional |
| TMO Middleware | TMO toolkit XP v4.2.3 (developed by Dream Lab. UC Irvine[22]) |

Figure 9 Execution cycle of ReadFromCS_SpM and Dynamics_SpM

Message transmission time: T_{send}
ORT of message at transmission: T_{send} + 20msec
Reading message time: T_{read}
T_{send} <= T_{send} + 20msec <= T_{read}
each SimTMO object (from #0 to #3). The TABLE V lists message access time of the SimMonitor_SpM and index of each message.

According to the first column of TABLE IV and TABLE V, the index #0 message from the SimTMO #0 is transmitted at 1,008,208 microseconds, and the SimMonitor_SpM reads the message at 1,040,820 microseconds. When the first cycle is passed, the message is not read because the ORT of message (transmission time plus 20 milliseconds) is not passed although it has been arrived already at the SimMonitorTMO object. Thus all messages from SimTMO’s are always read after 2 cycles are passed.

In the TABLE IV, we can also realize that each transmission time of four messages is similar but not accurately same enough. Nevertheless, due to the global time synchronization and the ORT, distributed data are gracefully accepted at the same time. As shown in the TABLE V, all indexes of messages synchronized in the SimMonitorTMO are same in every cycle.

Fig. 11 shows the time differences between message transmission time at each SimTMO and access time of the SimMonitor_SpM. The X axis means the index of message and the Y axis means the time difference. The range of time difference is from 0.03 to 0.04seconds.

The slight gaps between four graphs are by unpredictable latency of network or drift of global time synchronization. However we can conclude that the messages from the distributed simulators are timely retrieved by using ORT mechanism. The ORT boundary has been decided through a series of experimental measurements by starting from small value, e.g. 10 milliseconds, and increasing it up to the point of no ORT violations due to the increment of the number of TMO objects.

VI. ConClusion And fuTuRe Work

The multi-HILS consists of a number of distributed computers simulating each corresponding target system, a monitoring server, and a visualization tool. In this paper, we discussed about real time issues in developing multi-UAV simulation systems, and suggested multi-HILS architecture based on distributed real-time object model, i.e., TMO.
ACKNOWLEDGEMENT

This research was supported by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2010-C1090-1031-0003), and by grant No.R33-2008-000-10068-0 from the World Class University (WCU) project of the Ministry of Education, Science & Technology (MEST) and the Korea Science and Engineering Foundation (KOSEF) through Konkuk University.

REFERENCES