ภาคผนวก ข บทความเรื่อง A Simulation of the HLA Infrastructure Based on the ns Network Simulation Tool

The 2004 European Simulation Interoperability Workshop (Euro-SIW), Edinburgh, United Kingdom, 28th June - 1st July 2004.

A SIMULATION OF THE HLA INFRASTRUCTURE BASED ON THE *NS* NETWORK SIMULATION TOOL

Atinat Palawan
Pichaya Tandayya
Suntorn Witosurapot
Computer Engineering Department, Prince of Songkla University, Thailand
S4412081@maliwan.psu.ac.th, pichaya@fivedots.coe.psu.ac.th, wsuntorn@fivedots.coe.psu.ac.th

Keywords:
Network Simulation Tool, Federation Performance, Distributed Simulation, HLA, RTI, Object Management, Time Management

ABSTRACT: *The development of a network simulation tool for prediction of the performance of HLA federations using ns, a network simulator, is to be proposed in the paper. Although many studies have been conducted, few reports on HLA federation performance in terms of networking and infrastructure have been published due to the complexity involved in real situations. In other words, it is rather difficult to analyze the speed and reliability of HLA federations, especially in public networks. Tools on HLA are mostly focused on how to build federations. This paper will discuss the performance of HLA federations based on their designed infrastructure specification and demonstrate it with the HLA-extension of ns. We use ns because it is the most popular public domain software amongst researchers. However, ns has no special components that support the performance analysis of HLA federations. It only supports typical network protocols. We added the HLA components based on the interface specification into the ns. The HLA components include object management and time management. Our results show that the HLA extension on ns can adequately be used as a tool for studying the HLA RTI framework in simulated network environments.*

1. Introduction

1.1 Motivation

In creating distributed simulation systems using High Level Architecture (HLA) [1], developers often experience performance problems, especially in cases when the federates in the federation communicate with one another through public networks. On public networks, typical problems of the HLA are high consumptions of link bandwidth and processing time at simulation nodes. These are significant, and it is difficult to find the main causes [2]. HLA federations are usually very complex. Therefore, the symptoms of the performance problem usually appear far from the source of the problem.

Over the past few years, tools, techniques and models about the HLA have been developed for studying, understanding and solving the performance problems of HLA federations [3]. They focused on external and simple parameters of the actual distributed simulations, such as using constant formula for calculating the performance in various environments. However, none of the current HLA tools can assist network performance analysis. A simple example is shown in Figure 1.1. The upper network topology has only one router and the lower network topology has two routers. Obviously, transmitting data through the Run-Time Infrastructure (RTI) [1] node on the upper network topology has a better performance than on the lower network topology. Our work proposes a tool that allows users to look at the network's factors on the HLA federation and RTI such as different topologies and number of objects for distribution, and to use the network simulation tool as a means to HLA federation performance prediction.

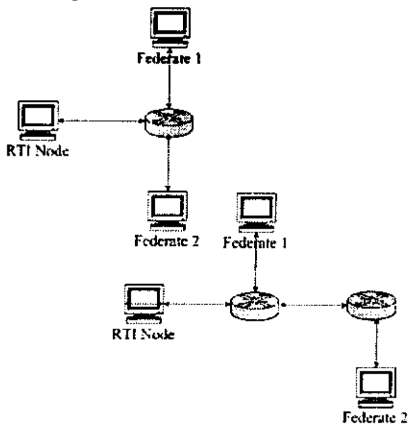


Figure 1.1 Different network topologies cause different performance results

1.2 HLA Module on a Network Simulation Tool

From our observation, an HLA federation is composed of three kinds of components: the RTI component, network components, and simulation components. This leads to our design of the simulation of HLA models on a network simulation tool that consists of a network model, an RTI model and simulation models.

The network model includes a representation of the network topology, link delay, queue types, protocols and so on. The simulation model includes a group of simulation entities that represents the statistical characteristics of each entity. We can consider a simulation entity as a random message generator that produces data or events [3]. The RTI model includes a representation of the RTI service functions such as creating and joining a federation, publishing and subscribing objects and attributes, updating and reflecting attributes, etc.

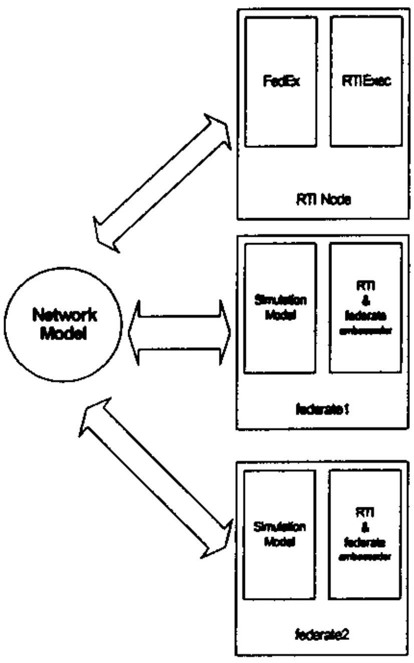


Figure 1.2 Concept of an HLA federation

The following subsection explains why we use *ns* as a base tool, and the overview for how we add the HLA module into *ns*.

1.3 *ns* Extension

The *ns* network simulator has been chosen as our base simulation tool because it is well known and popular in the computer network community. It is also well documented and built for network analysis[4]. *ns* is an efficient network simulation tool that can be used to represent a complex network topology including essential parameters such as link delay, queue types and so on. The HLA simulation models and RTI models can be added into *ns* in the same way that existing *ns* network modules are represented. *ns* was designed with an object-oriented style that the user can extend or add in functions by deriving existing components in order to create a new protocol or a new network entity. In this case, as HLA is located at the OSI application layer, we derived our HLA module from Class *Application* in the same way Class *Telnet* and *FTP* do (Figure 1.3). The object derived from class *Application* can apply *ns* objects such as *Connector*, *Delay*, and *Agent* in connection and communication.

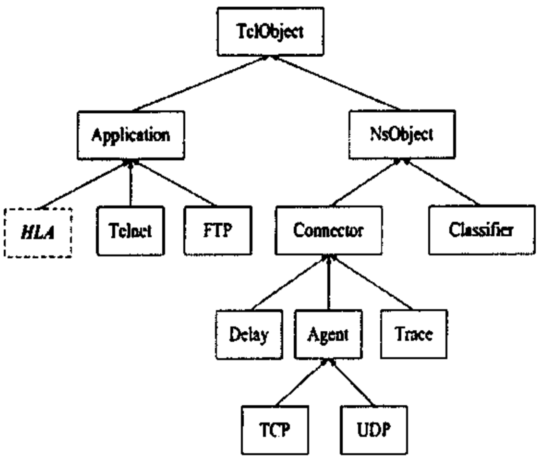


Figure 1.3 The HLA module is extended into *ns* in the same way the other application layer protocols are built

Generally, other analyses on performance of the HLA focus on the simulation and RTI models, and do not really focus on the simulation of the HLA network model. The network model was represented by only constant parameters or formula [2, 5]. Our work combines all three models and allows simulating an HLA federation with a complex and changeable network topology.

This paper discusses the design, development and test of the HLA federation simulation tool on *ns* in the areas of object management and time management. The sections are arranged as follows: Section 1 explored the overview for how we add the HLA module into *ns*. Section 2 talks about the design of the HLA extension of the *ns*. Section 3 shows the case study and theory used to support the results, such as the queueing theory for object distribution and the graph result that proves the distributed time management algorithm used in the time management. The final section is the conclusion.

2. Design of the HLA Network Model

2.1 Design Overview

As the HLA framework performs on the application layer, the HLA extended module derives the characteristics from Class *Application* of *ns* as shown as dashed boxes of *Simulation Object* and *Federate Agent* in Figure 2.1.1 and 2.1.2. *Simulation Object* (Sim Obj) simulates the behavior of locally simulated objects and remotely simulated virtual objects of other federates at a federate. It is designed to operate with Class *Random Variable* that allows the configuration of different types of data distribution.

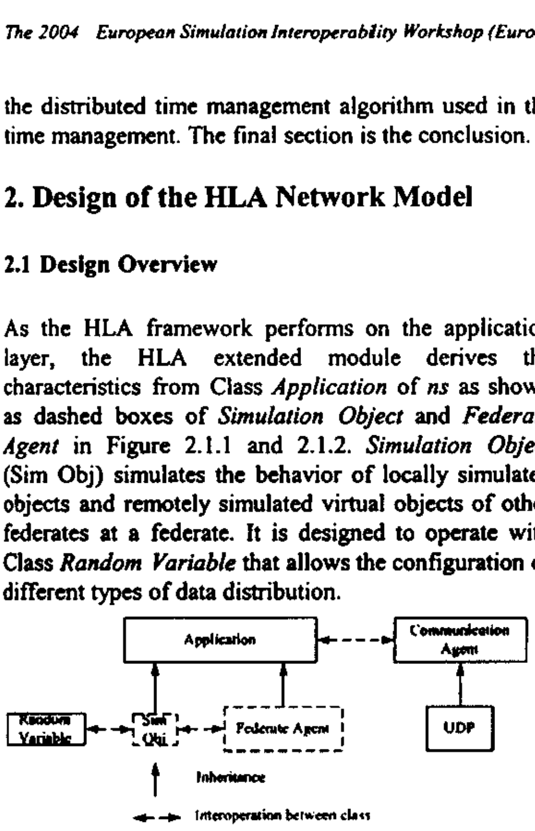


Figure 2.1.1 Attribute Distribution Extended Module

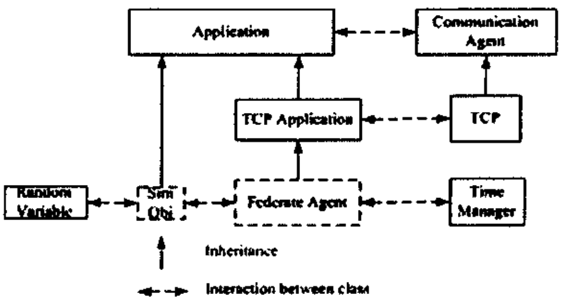
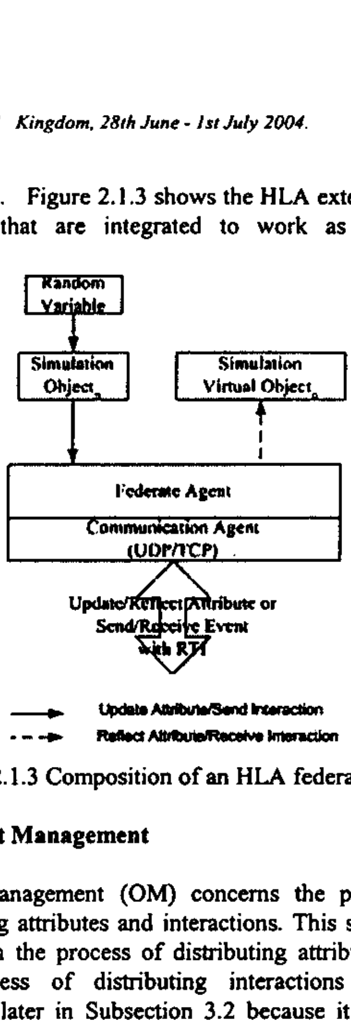


Figure 2.1.2 Interaction Distribution and Time Management Extended Module

Federate Agent is designed to simulate the behaviors and work as an agent of the Federate Ambassador and RTI Ambassador. In Figure 2.1.1, the *Federate Agent* works with the UDP protocols in order to transmit and receive data. In Figure 2.1.2, on the other hand, the *Federate Agent* works with the *Communication Agent* using the TCP protocols to ensure the reliability of transmitting and receiving events and time control messages. Additionally, the *Federate Agent* works in cooperation with the *Time Manager* module that calculates the Lower Bound on the Time Stamp (LBTS) [6] of the simulation system or the considered federation. Figure 2.1.3 shows the HLA extended sub-modules that are integrated to work as an HLA federate.



Random Variable
Simulation Object
Simulation Virtual Object
Federate Agent
Communication Agent (UDP/TCP)
Update/Reflect Attribute or Send/Receive Event with RTI
Update Attribute/Send Interaction
Reflect Attribute/Receive Interaction

Figure 2.1.3 Composition of an HLA federate in *ns*

2.2 Object Management

Object Management (OM) concerns the process of distributing attributes and interactions. This subsection focuses on the process of distributing attributes only. The process of distributing interactions will be discussed later in Subsection 3.2 because it concerns time management.

In general, to distribute attributes, the User Datagram Protocol (UDP) is used in order to have the best effort transmission. If a reliable protocol is needed, the Transmission Control Protocol (TCP) will be used. However, in practice, UDP is mainly used for updating and reflecting attributes in most RTIs due to less complexity.

In distributing attributes in the HLA, there are two functions or services, namely, *Update Attribute Value* and *Reflect Attribute Value*, that are defined as follows.
Update: a simulated object as a source of information transmits data to the RTI.
Reflect: the RTI looks for the virtual objects that require updates from the source objects and then transmits the object attributes to the destinations where the virtual objects are located. The structure of this is shown in Figure 2.2.1.

Figure 2.2.2 shows an example of updating and reflecting attributes. Entity A is a simulated object at Federate 1 that publishes attributes. Federate 2 subscribes attributes from Entity A. Therefore, we create a virtual object for Entity A in order to work as a

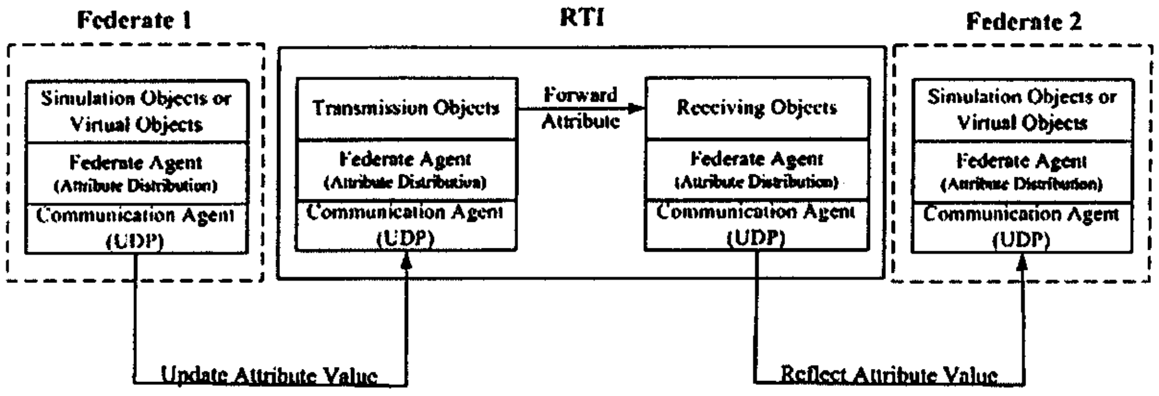


Figure 2.2.1 Simulation layers of the HLA framework on object management

target for the RTI model to reflect data to. In other words, we can think of the RTI model as a pure data forwarding engine. Figure 2.2.3 shows the mechanism inside the RTI module including publication and subscription mechanism using lookup tables for forwarding data to update virtual objects on the subscribing federates.

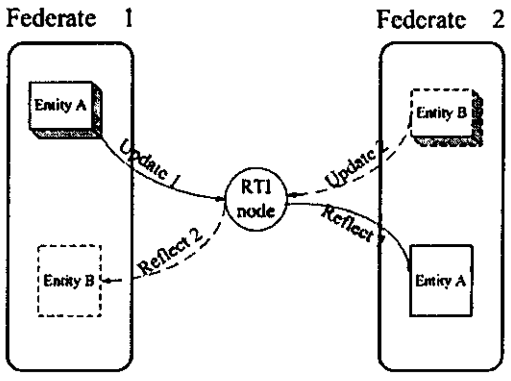


Figure 2.2.2 Attribute distribution in an HLA federation

After the functions for updating and reflecting data on the HLA model were developed, we then used the principle of queueing theory as a standard for comparing results from a simulation tool and the expected results from the theory. The results of the case study and queueing theory used are shown in Section 3.

2.3 Time Management

Time Management is the HLA service concerning the control of time advancement so that logical times of participating federates synchronize with one another in order to prevent disorder of messages, especially the time stamp order message that contain a time stamp which is the time to activate an event. The RTI acts as the time advancement controller in this context.

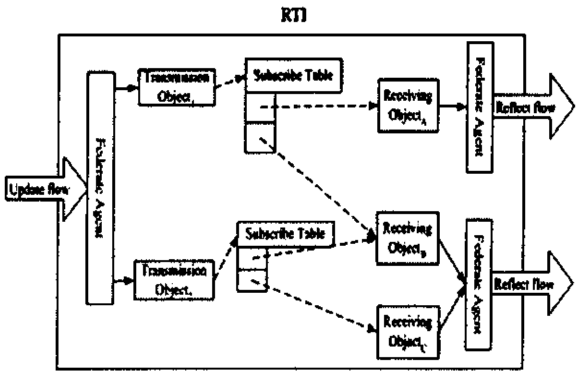


Figure 2.2.3 RTI mechanism on forwarding messages

The management process of time synchronization consists of two main steps, namely, a request from a federate for advancing its logical time and the acceptance granted by the RTI to the request. This is explained in details below [7].

- There are two patterns of requests for advancing the logical time, namely, Time Advance Request (TAR) and Next Event Request (NER).
- The Time Advance Grant (TAG) service actually supports both conservative and optimistic synchronization. In this paper, however only the RTI module applies the conservative policy that the Lower Bound on the Time Stamp (LBTS) is computed in order to prevent the disorder of event processing. Equation 1 is used for computing the LBTS from the minimum among regulating federates participating in the considered federation. [6].

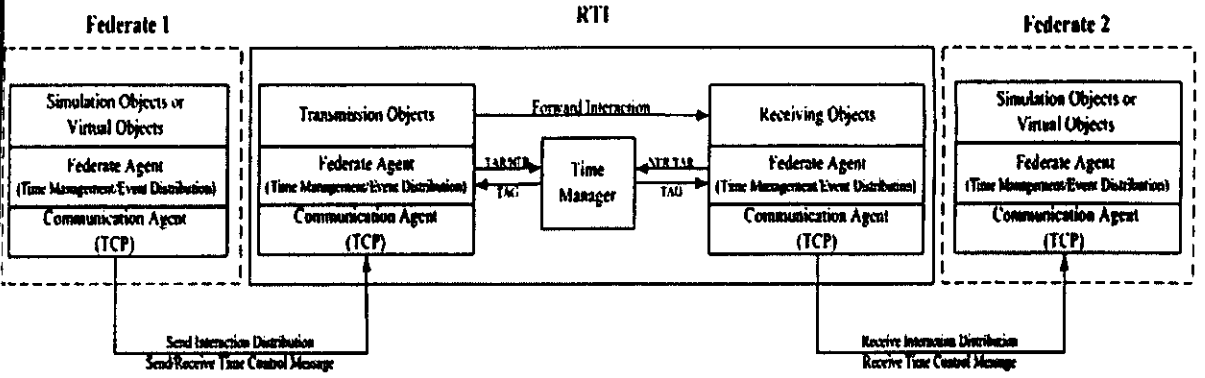

Federate 1
RTI
Federate 2
Simulation Objects or Virtual Objects
Federate Agent (Time Management/Event Distribution)
Communication Agent (TCP)
Transmission Objects
Forward Interaction
Receiving Objects
TAR/NER
TAG
Time Manager
NER/TAR
TAG
Send Interaction Distribution
Send/Receive Time Control Message
Receive Interaction Distribution
Receive Time Control Message

Figure 2.3.1 Simulation layers of the HLA framework on time management

$$LBTS = \min(T_{REG} + L_{REG}) \;\ldots\ldots\ldots\ldots\ldots\; (1)$$

when

T_{REG} is the requested time for advancement of regulating federates.

L_{REG} is the lookahead of regulating federates.

Before time advancement is granted by sending the TAG message, the RTI will distribute all messages for activating events that have the time stamp lower than or equal to the time being granted in order to ensure that no event activating messages arrive late and to prevent an event disorder. From this point, the RTI is not only a data forwarding engine but also a controller that controls scheduled events and ensures the correct order of events.

Figure 2.3.1 shows the communication overview of the HLA layer on *ns* focusing on Time Management. The *Time Manager* module is created to calculate the LBTS and to work in cooperation with the *Federate Agent* according to the HLA time advancement. The *Federate Agent* requests to advance its logical time by sending a TAR or NER message. The LBTS will then be computed before granting a corresponding TAG message to the requesting *Federate Agent*. The *Communication Agent*, in this case, applies the TCP protocol in order to ensure the reliability of event distribution.

Figure 2.3.2 shows an example of the HLA time management. In this case, the logical time is initialized at 0 and the two federates have the same value of lookahead which is 1. A federate that subscribes events and has its logical time is constrained by other federate is called a constrained federate, and a federate which determine the logical time of an other federate is called a regulating federate[8].

At Point A, the current logical time of constrained federate is 0. The constrained federate sends the TAR(2) message to the RTI in order to advance time to 2.

At Point B, at the same logical time 0, the regulating federate sends a time stamp order message or an event with the time stamp 1 (E(1)) and then the TAR(1) message.

At Point C, the RTI receives the TAR(2) message from the constrained federate. Then, the LBTS is calculated, from the values T_{REG} of 0 and L_{REG} of 1, as min(0+1) or 1. The requested time is 2 which is greater than the computed LBTS. Therefore, the RTI postpones the TAR(2) message and puts the request into the waiting queue.

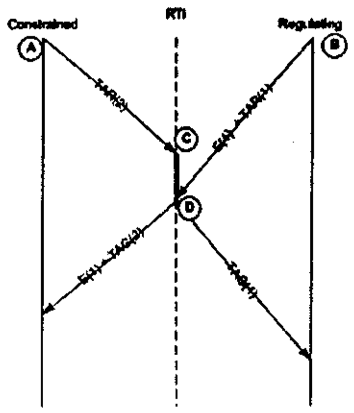


Figure 2.3.2 Synchronization among the RTI, constrained federate and regulating federates in time management

At Point D, the TAR(1) message from the regulating federate arrives. Then, the RTI takes all TAR messages from the waiting queue to recalculate the LBTS. Now, the LBTS is min (1+1) or 2, consequentially, the RTI grants TAG(2) to the constrained federate and for the regulating federate TAG(1) is immediately permitted.

Unlike attribute distribution, the event or interaction distribution process needs to be done before allowing the logical time advancement. No lost event messages are allowed during transmission. Therefore, we use the TCP is used as a base protocol in this case.

In the next section, for the verification of the HLA time management module extended on the *ns* network simulator, we used a time advancement graph as a measuring tool.

3. Performance Measurement and Verification

3.1 Object Management

In order to test if the HLA extended module on *ns* works according to the theory, we used the topologies shown in Figure 1.1 as a fundamental case study. Given that Federate 1 publishes Object A and Federate 2 subscribes Object A, it will lead to the processes of updating and reflecting the attributes of Object A. Object A was set as follows:

- The Communication Agent was derived from the UDP module.
- The update rate was 1 time per second, the average size of update data was 110 bytes, and the exponential distribution was used.
- Every link has a communication bandwidth of 1 kbps and a delay of 10 ms.

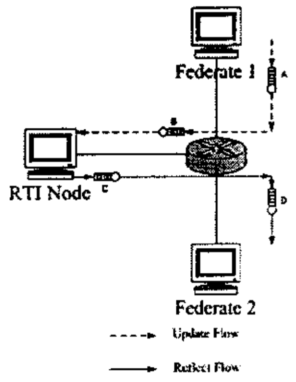


Figure 3.1.1 Flow of queues for Topology 1

Figure 3.1.1 shows the flow of queues for Topology 1 which has only one router in between Federate 1 and 2. From observation, the lengths of queues from Queue A to Queue D were 6.27, 2.90, 3.22 and 3.45 sequentially.

In theory, the number of packets in a queue can be computed by Equation 2 [9]. The limitation of this equation is that the flow of input data of queues must be independent or randomly generated. Therefore, the lengths of Queues B, C, D can not be computed. For example, the input flow to queue B always passes through Queue A. This is also the reason why the performance estimation for a complex computer network requires a simulation and can not be done by theory only. However, if the first queue, Queue A, works properly, other queues that follow Queue A can be assumed to work correctly, too.

From the above parameter configuration for testing the HLA extended module, λ was set to 1 packet per second, and μ was set to 1000/(8*110) or 1.136 packets per second. Applying Equation 2, N_Q, the average number of data packets in theory, is 6.49. The actual length of Queue A simulated in *ns* is 6.27. The difference is 3.38% which is acceptable.

$$N_Q = \lambda^2/\mu(\mu-\lambda) \qquad (2)$$

when

N_Q is the average number of data packets in queue
λ is the average arrival rate of packets
μ is the average service rate of a simulation node or router

Comparing Topology 1 and 2, Figure 3.1.1 and 3.1.2, the lengths of Queue A to Queue D resulted the same. However, Topology 2 additionally had Queue E with the queue length of 3.62.

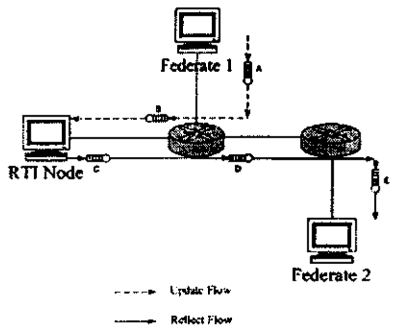


Figure 3.1.2 Flow of queues for Topology 2

From the lengths of queues, the packet dropping rate can be estimated. In other words, we can estimate the reliability of attribute or interaction distribution of a federation that can normally be done only after an actual federation is created. From the experiment, the simulation result supports the assumption that

Topology 2 in Figure 3.1.2 has less reliability than Topology 1 in Figure 3.1.1 due to the additional Queue E.

3.2 Time Management

The time advancement was verified by comparing the logical time and physical time. Using Topology 1 in Figure 1.1, parameters were configured for testing the time advancement of the HLA extended module as follows:

- Federate 2 was a regulating federate and Federate 1 was a constrained federate.
- The lookaheads of both federates are equal to 1.
- The delay between Federate 2 and the closest router was set to 100 ms (Figure 3.2.1) and 50 ms (Figure 3.2.2). The delays of other communication links were set to 5 ms.
- The bandwidth for each link was 2 Mbps.
- The Communication Agent was derived from the TCP module.
- TAR was used for regulating time advancement.

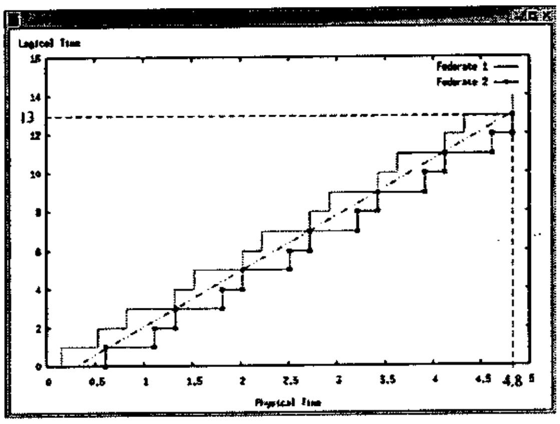


Figure 3.2.1 Time advancement when the delay is 100 ms

Figure 3.2.1 and Figure 3.2.2 show the results of the simulation when the delay between Federate 2 and the closest router was set to 100 ms and 50 ms, respectively. Figure 3.2.1 shows that for the duration of 4.8 seconds (physical time), the logical time advanced to 13. The speed of the topology was 13/4.8 or about 2.7. Figure 3.2.2 shows that the speed of the topology was 23/4.75 or 4.84 which is faster. It shows that the higher communication link delay causes a slower time advancement, or that the simulation advances slowly if the communication link delay is high.

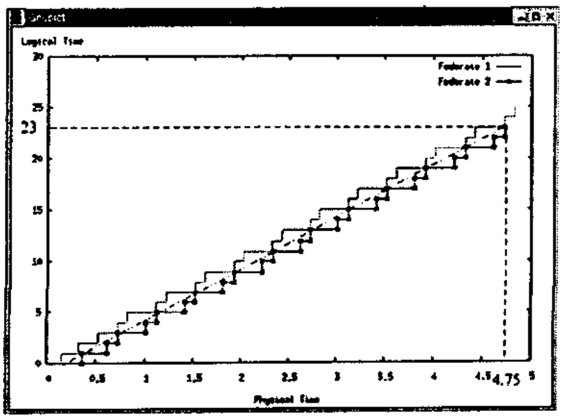


Figure 3.2.2 Time advancement when the delay is 50 ms

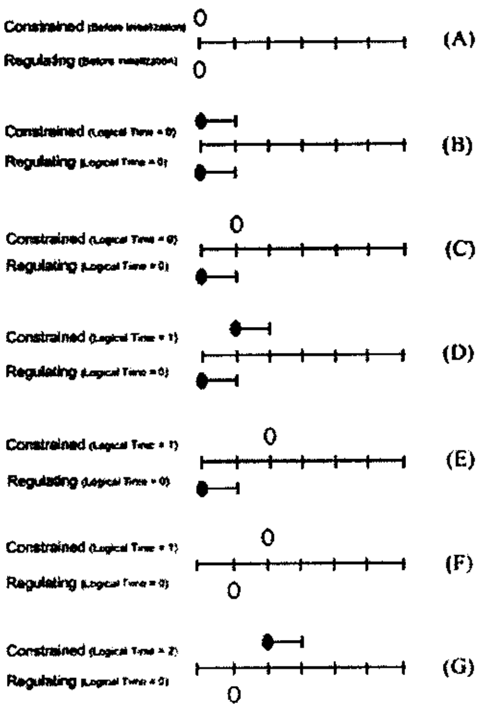

Constrained (Before initialization)
Regulating (Before initialization)
(A)
Constrained (Logical Time = 0)
Regulating (Logical Time = 0)
(B)
(C)
Constrained (Logical Time = 1)
Regulating (Logical Time = 0)
(D)
(E)
(F)
Constrained (Logical Time = 2)
Regulating (Logical Time = 0)
(G)

Figure 3.2.3 The time advancement graph

From the results, we applied the time advancement graph to ensure the correctness of the simulation. The graph uses the following symbols:

- A white circle shows that the considered federate is waiting for the TAG message after sending the TAR message.
- A black circle shows that the considered federate has already received the TAG message to advance its logical time. The line after the black circle shows the length of the lookahead that guarantees that the federate

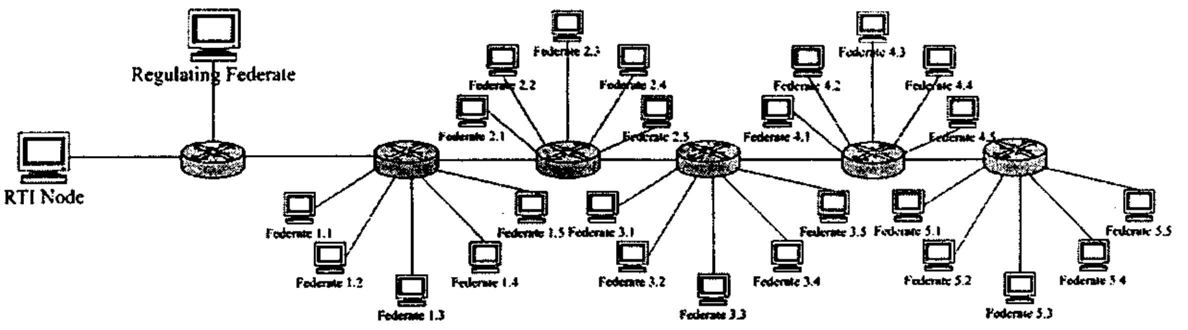


Figure 3.3.1 25-node HLA federation for scalability test.

will not generate events with a time stamp within that line as well as events with a timestamp lower than its current logical time.

In Figure 3.2.3, we considered the time advancement of the regulating and constrained federates from Figures 2.3.2 and 3.1.2 in order to check if our simulation was correct. From the experiment, the constrained federate tries to advance time but the logical time of the constrained federate will not be able to advance much beyond the logical time of the regulating federate (E and F). Figure 3.2.3 (E) shows that the constrained federate was at the logical time 1 and had already sent the TAR(2) message to the RTI. The regulating federate was at the logical time 0. When the TAR(2) message arrived at the RTI, the LBTS was calculated as min(0+1) or 1. Therefore, the TAR(2) request was not allowed. The constrained federate needed to wait until the TAR(1) message from the regulating federate arrived at the RTI and the LBTS could be calculated again. At the end (G), the constrained federate was permitted to advance the logical time to 2.

From the graph results, the logical time of the constrained federate can not be different from the logical time of the regulating federate more than a threshold of 2. Additionally, this verifies that the time management of our HLA extended module on *ns* works according to the HLA rules and RTI specification.

3.3 Scalability of the HLA Extended Module on *ns*

Scalability is one of the most important issues in distributed simulation and performance analysis. However, the analysis on the scalability of the HLA extended module on *ns* requires complicated experiments and takes quite some time. Currently, we have completed some brief experiments to observe that our extended module supports scalability to some extent.

In Figures 3.1.1 and 3.1.2, we focused on the simulation experiments on *ns* of a 2-node federation. In this subsection, we carry out experiments on an HLA federation with more node members. This time the total number of routers in the federation is 6. Our router connects to the RTI node, a regulating federate and one other router. Each of the other 5 routers has 5 or 10 constrained federates connected to it as shown in Figure 3.3.1. The HLA federation built on *ns* ran on a personal computer with the microprocessor Intel Celeron 1.7 GHz, memory of 128 Mbytes, and the Linux OS set for personal use. The details of the tested federation were as follows:

- The lookaheads were 1.
- The delay of each communication link was set to 5 ms.
- The bandwidth for each link was 2 Mbps.
- TCP was used for regulating time advancement.

For the simulation of a physical time of 5 seconds, the task was completed within 16 seconds by the 25-node HLA federation and within 52 seconds by the 50-node HLA federation under the multitasking environment of the Linux OS. The results showed that the HLA extended module was scalable and behaved according to the expectation.

4. Conclusion

This paper has presented the design and development of a federation performance prediction tool based on the network simulator, *ns*, that will help distributed simulation developers studying, understanding and solving the performance problems of building an HLA federation over a simulated network before creating the actual federation. The case study proved that the results from the simulation are according to the theory and that the HLA extended module on *ns* is verified. More complex federations can be tested on the *ns*. The *ns* itself has no limitation of the number of network entities created on the simulator. Although the simulation speed decreases when the number of entities increases, the distributed simulation of *ns* using PDNS [4] can also be done. We can configure the network parameters such as the dropping rate, mean of delay in communication links and types of links (wired or wireless). The network simulation results can be used for estimating the performance and efficiency of the considered HLA federation.

Further works could include enhancing the tool to support the simulation of data distribution and ownership management, modifying the RTI model to be able to support optimistic algorithms, or conducting research on scalability.

5. References

[1] Defense Modeling and Simulation Office Web Site, http://hla.dmso.mil.

[2] Duncan C. Miller, Steven B. Boswell: "A General Framework for Modeling Federation Performance" Spring Simulation Interoperability Workshop, 2001.

[3] Stephen R. Kolek, Steven B. Boswell, Harry M. Wolfson: "Toward Predictive Models of Federation Performance : Essential Instrumentation" Fall Simulation Interoperability Workshop, 2000.

[4] ns manual, The VINT Project Website, http://www.isi.edu/nsnam/ns.

[5] Jenifer McCormack, Cristl Weckenman, Gene Lowe: "Development of a HLA Constructive Performance Model" Spring Simulation Interoperability Workshop, 1999.

[6] R. M. Fujimoto: "Conservative Synchronization Algorithms" Parallel and Distributed Simulation System, John Wiley & Sons Inc, USA, 2000.

[7] R. M. Fujimoto, R. M. Weatherly: "HLA Time Management and DIS" Simulation Interoperability Workshop, 1996.

[8] F. Kuhl, R Weatherly, J, Dahmann: "Synchronizing the Federation" Creating Computer Simulation Systems, Prentice Hall PTR, NJ, 2000.

[9] D. Bertsekas, R. Gallager: Data Network, Prentice-Hall, Inc , New York, 1992.

Author Biographies

ATINAT PALAWAN graduated in Computer Science from Prince of Songkla University (PSU), Thailand. Now, he is pursuing a master's degree in Computer Engineering at the Faculty of Engineering, PSU. His thesis topic is *A Simulation of High Level Architecture (HLA).*

PICHAYA TANDAYYA graduated in Electrical Engineering (Communications) from Prince of Songkla University (PSU) in Thailand in 1990. She obtained her PhD in 2001 from the University of Manchester on distributed interactive simulation. She is a lecturer in the Department of Computer Engineering, PSU. Her current research work is concerned with distributed interactive simulation, High Level Architecture, and a Braille keyboard and display unit for the visually-impaired.

SUNTORN WITOSURAPOT works as an assistant professor at the Department of Computer Engineering, Prince of Songkla University (PSU). He received his bachelor's and master's degrees from PSU in 1985 and 1990 respectively. He is waiting for his PhD award from Swinbourne University. His research areas are computer networks, differential services, 3rd generation mobile phones, and wireless communication.