Appropriate Synchronization Time Allocation for Distributed Heterogeneous Parallel Computing Systems

***Biruk Yirga Nidaw, *§Myeong-Hoon Oh, *§Young Woo Kim**
*Department of Computer Software, University of Science and Technology (UST),
217, Gajeong-ro, Yuseong-gu, Daejeon, 34113, KOREA
§SW Contents Research Laboratory, Electronics and Telecommunications Research Institute (ETRI),
218, Gajeong-ro, Yuseong-gu, Daejeon, 34129, KOREA
(biruky, mhoonoh, bartmann)@etri.re.kr
*Corresponding author: Young Woo Kim

Received January 6, 2019; revised April 25, 2019; accepted May 22, 2019; published November 30, 2019

Abstract

Parallel computing system components should be harmonized, and this harmonization is kept existent using synchronization time. Synchronization time affects the system in two ways. First, if we have too little synchronization time, some tasks face the problem of harmonization, as they need appropriate time to update and synchronize with the system. Second, if we allocate a large amount of time, stall system created. Random allocation of synchronization time for parallel systems slows down not only the booting time of the system but also the execution time of each application involved in the system. This paper presents a simulator used to test and allocate appropriate synchronization time for distributed and parallel heterogeneous systems. The simulator creates the parallel and heterogeneous system to be evaluated, and lets the user vary the synchronization time to optimize the booting time. NS3-cGEM5 simulator in this paper is formed by HLA-RTI federation integration of the two independent architecture and network simulators - NS3 and cGEM5. Therefore, nodes created on these simulators need synchronizations for harmonized system performance. We tested and allocated the appropriate synchronization time for our sample parallel system composed of one x86 server and three ARM clients**.**

Keywords: Distributed systems simulations, CERTI, HLA, GEM5, NS3, Synchronization

1. Introduction

Recently, with an improvement in the number and performance of complex machines in distributed and parallel processing systems, solving computationally intensive tasks for efficient resource utilization with improved performance has become easier. In the era of technology improvements, repeated system tests are indispensable. That is, the need for simulation tools is inevitable, almost in all aspects of science and arts, especially in the case of testing and expermenting with grand ideas or non-existing systems. Parallel computing technique was developed from serial computers to overcome the single state of work and to exploit multitasking. Performance improvement by application partitioning and scheduling tasks [1]-[2] on interconnected nodes to execute concurrently is the contemporary approach. With an increase in the number of processes/processors, the work done by each process/processor decreases [3]-[4]. Alternatively, an increase in the share among processes is an increase in its data usability and speeds up the performance.

The existence of parallel and heterogeneous systems simulator enables to select appropriate synchronization time for distributed and parallel heterogeneous systems. Simulation tools used to simulate the architecture structure of standalone computers in GEM5 [5] and CPUSim [6] have been proposed, and network performance analysis and simulation have been proposed in NS3 [7]. However, these simulation tools cannot be used in parallel and distributed heterogeneous computing system simulation as they have scalability issues. For instance, GEM5 [5] can run a maximum of 2 parallel homogeneous systems at a time.

In regard parallel system simulation, Mohammed Alian et al. [8] have proposed Dist-GEM5 simulation tool to simulate the architectural structure and network behaviour of parallel and distributed heterogeneous computing systems. Dist-GEM5 in [8] creates communication channels for GEM5s installed on standalone devices to communicate via the host network system and to address network simulation tool's (GEM5's) scalability problem over parallel systems. COSSIM [9] can run distributed and parallel heterogeneous system models by integrating cGEM5 with OMNET++. It has applied customized GEM5, and customized OMNET++, which they call it, cGEM5 & cOMNET++ respectively. In [10] Anis and et al. gave **Table 1** that makes a comparison on NS2, NS3, OMNET++ and others. To address the scalability issues raised in standalone simulators and to use the capabilities of NS3 over other network simulators, we propose an NS3 and cGEM5 integrated simulation tool. The proposed simulator used cGEM5 to simulate architectural structures such as CPU, memory, storage and input/outputs of parallel and distributed heterogeneous systems, and NS3 used to simulate the network devices, connection links, communication protocol suits and network loads of the nodes created by cGEM5.

The integration of the two independent simulators is the most essential part of the integrated simulator [11]. The isolation complexity of operation and ease of the controllability for the integrated simulator are achieved by using HLA–RTI system for integration. The HLA–RTI system lets the two simulators operating independently, and this leads the integrated simulator to isolate problems and operations of each simulator. The HLA–RTI system is an open-source based standard system, and it is easy to understand and modify [12]. In addition to these, the HLA–RTI system does not interfere with the measured result of tests; thus the integrated system becomes more robust to any problem caused by individual simulators. Distributed system simulators are used in design optimization and in design mistakes reductions [13].

Table 1. Performance comparison among network simulators [10]

TEST	NS2	NS3	OMNET++	GloMoSim
Memory Usage	Highest amount	Lowest amount	Average amount	Average amount
CPU Usage	Higher	Higher	Lowest	Lowest
Speed	Slowest	Fastest	Fast	Fast
Computation Time	Highest	Lowest	Low	Low

The proposed NS3-cGEM5 integrated simulator is used for the synchronization measurement test and for finding the appropriate synchronizing time. The results of the proposed simulator are compared with other simulators and found a comparative result. This research work has an important role and a big hand for designers and system engineers for distributed and parallel heterogeneous systems simulation. As computing system evolve, systems become more complex, and the distributed and parallel system becomes common. The computing system is now evolving to newer architecture – like, memory oriented computing architecture, processing in the memory, and so on. The application of an integrated simulator gives more efficient, effective, and methodical means than applying multiple separated single simulator for parallel and heterogeneous systems. The contributions of this work are an alternative integrated simulator for distributed and parallel computing systems, an integration of existing NS3 with GEM5 for better-performing simulation performance than separated simulators, and fast, flexible and precise simulation by using NS3 features for parallel and distributed heterogeneous systems.

The rest of this paper is presented in the following manner. The next section surveyed the related works and followed by details of the methodology for the proposed integrated simulator, that describes details on the implementations of the proposed simulator. Section 4 gives the experimental tests and results of the proposed integrated simulator, and finally, section 5 outlines the conclusion drawn as well as future works.

2. Related Work

Simulators in computing are classified into architecture (processors) simulators and network simulators. As examples of architecture simulators, we can mention simulators like MikroSim, CPUSim, HASE, Sniper, GEM5 and Zsim. On the other hand, in case of network simulators that simulate the communication flow of a system, we can mention simulators like NS2, NS3, OMNET, OPNet and OMNET++ [14]. In the following parts of this section, literature regarding this architecture, network and improved simulators will be discussed.

2.1 NS3

The NS3 is a network simulator that applies a discrete-event network simulation [15]. it is predominantly targeted for researchers working on network communications and for network system courses. NS3 licensed under the GNU GPLv2, which is available for researchers and network system developers for free. It outlines a model of the proper working flow of packet data and provides a mechanism for modelling and simulation. NS3, unlike some other

network simulators, can model with and without internet connections, but most researchers are using NS3 to model a system without a connection of internet.

NS3 uses two principal languages C++ and python [7], and scripts written in C++ or python can be used for execution. NetAnim is used for the animation to visually display the results. Both programming languages provide a robust library, which is helpful for the user requiring less effort to edit it for their specific need. NS3 provides models for wired technology, models of a simple network of Ethernet, which uses CSMA/CD as its network protocol. NS3 also delivers a set of 802.11 models to provide precise MAC-level employment of the specifications 802.11 and a PHY-level 802.11a model [16]. It reduces the simulation memory footprint and allocates no memory for the virtual zero byte values. In NS3, mobility model is not required as the node position of the simulated network does not need to wired devices.

Moreover, NS3 is capable of producing packet trace files for debug purpose using PCAP (packet capturing mechanism). Protocol units in NS3 are designed to be nearly the same as that of real computers. Additional resources based on its open-source are supported in NS3 networking software and there reduce the need to rewrite models for simulation, but it is not able to simulate the architectural structure of computing systems.

Table 2. Network simulation tools [17]

Simulator	Interface	Emulation	Source	Prog. Lang.	Platform (OS)
NS2	CLI	Yes	Open	C++, OTcl	Windows, Linux, Mac OS, Free BSD
NS3	CLI	Yes	Open	C++, Python	Windows, Linux, Mac OS, Free BSD
OMNET++	GUI	Yes	Open for education	C++	Windows, Linux, Mac OS
NetSim	GUI	Yes	Open	C, C++, Java	Windows
OPNET	GUI	Yes	Open	C, C++	Windows
J-Sim	GUI	Yes	Open	Java, TCL	Windows, Linux

NS3, OMNET++, and OPNet are capable of carrying out large-scale network simulations. Note that NS3 is the fastest simulator [18], among the mentioned simulators in **Table 1** regarding computation time [19]. **Table 2** shows comparesion among networksimulators in variation points of view. NS3 has varieties of modules which show its modular capabilities. NS3 is publicly available for academic and non-academic use. It encourages the community contribution in the development of simulation models to be sufficiently realistic to permit NS3 to be used as a real-time network [20].

2.2 GEM5

Researchers in different areas of fields may need their systems for test and require a flexible simulation system framework that can assess a wide diversity in designs and support rich OS services including input-output and networking. GEM5 is an open source software with BSD-based license, and the code is accessible to all researchers without any legal limitations [5].

Among architecture simulators, we selected and compared some familiar architecture simulators. We found GEM5 is the more capable architecture simulator. **Table 3** shows a comparison among architecture simulating tools [21].

Table 3. Architecture simulation tools comparison

Simulator	Prog. Lang.	Sim. Type	Source Type	Compatible OS	Remark
GEM5	Python C++	Microarch. Full & SE	Open	Linux Ubuntu	Wide Variety of capabilities & flexibility
CPU Sim	JAVA	Full	Open	MS-Win, Linux, Mac	Support Comp. Arch. Education
Slack Sim	POSIX Thread Prog. Model	CMS*	Open	Linux	Cycle Acc. Sim. and Check pointing

*Chip Multiprocessor Simulator

GEM5 is composed of M5 and GEMS, which have their impacts on architecture simulation history. It has various capabilities that outperform on other architecture simulators. GEM5 full system simulator supports many ISAs with various CPU models, and is possible to test different applications on system emulation base. In regard, the CPU type GEM5 acquired detailed CPU modelling from M5 of its components which are 'AtomicSimple', 'TimingSimple', 'InOrder' and 'O3' (Out Of Order), and the simulation.

Table 4. GEM5 simulation capability [22]

GEM5 Capability					
ISAs	**Execution Mode**	**CPU Mode**	**Cache Coherence Model**	**Interconnection on Network**	**Devices**
Alpha ARM MIPS SPARKS x86 POWER	Full Mode System-Call Micro-architecture	Atomic Simple Timing Simple InOrder Out of Order	Slice Invalidation Based Logic Form Granularity	Simple Network Garnet	Several IO

Furthermore, the GEM5 provides a flexible and modular simulation that can evaluate a broad range of systems [23]. It is widely available to all researcher's simulator that overcomes limitations of modularity and poor coding problems by other simulators. This flexibility is achieved by offering a varied set of CPU models, mode of executions, and a variety of memory system models. **Fig. 1** shows flexiblity and accuracy comparison in design proceses. Based on flexibility comparison, the programmers design is the most flexible than others. And concerning the accuracy, the RTL representation is the most accurate mechanism in testing. However, GEM5, it is located on the appropriate position in between the programmers flexibility and the RTL accuracy position as shown in **Fig. 1**.

Nonetheless, GEM5 does not support parallel network modelling. If we want to run a full system simulation for a parallel system, only a pair of the similar system will be simulated with the same image files using "--dual" together with the GEM5 full system simulation command.

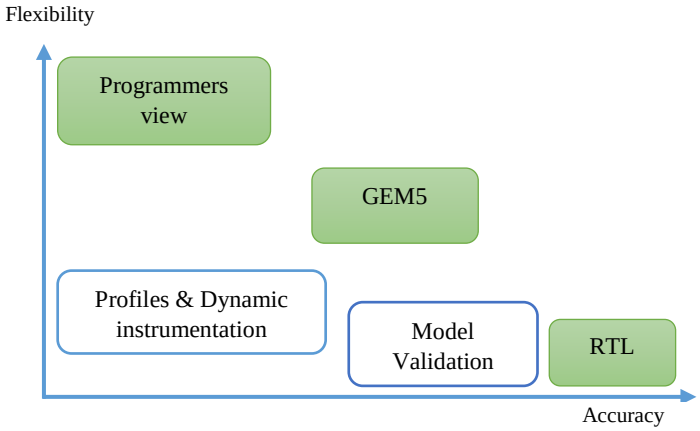


Fig. 1. GEM5 a flexible tool for architecture simulation [24]

2.3 Dist-GEM5

Dist-GEM5 combines two autonomous development methods, the pd-GEM5 together with the multi-GEM5, which is considered as a GEM5 distributed version. Dist-GEM5 tried to simulate several nodes using multiple simulation systems. This simulator uses TCP sockets as a channel for transfer of synchronization and data messages between a switch node and a full-system node, which enables to prevent data messages from avoiding synchronization messages (due to the strict ordering between TCP packets) [8].

This simulator improves the checkpointing mechanism of its previous work pd-GEM5 and is strongly coupled with the Ethernet protocol. Dist-GEM5 can deliver a fast, scalable and detailed infrastructure of simulation for modelling and evaluating large computing groups [8].

Getting the network from the host system enables to create parallel distributed system. Server-1 from host one will connect with Client-2 of the next host. This will proceed until it gets the last connection from the end host. Nodes per given system and heterogeneity are the central lack of Dist-GEM5. Enhancement in the network performance of GEM5 was achieved with COSSIM. Note here that Dist-GEM5 is considered as an extension of GEM5 and no combination with other simulation is made with GEM5 in the formation of Dist-GEM5.

2.4 COSSIM

COSSIM delivers the necessary hooks to security testing software, making it possible to determine vulnerabilities and inspect the toughness of the system under design. It is the first integrated solution that can give the mechanism of simulation for the actual system of systems, network dynamics and energy aspects. The goal is to provide a solution that offers functionality greater than using each component separately. COSSIM applied GEM5 for

designing a simulator for general purpose applications, to simulate a various kinds of nodes. COSSIM is a system simulator that is cycle-accurate, ISA independent, configurable, able to boot real-world operating systems and capable of executing software compiled for those systems [9]. Further more, it applies a dedicated network simulator that handles all network related modelling from the physical layer of an NIC and beyond. For such purpose, OMNET++ is chosen.

In COSSIM, integration of GEM5 and OMNET++ is made with the help of High-Level Architecture (HLA). HLA is a general-purpose software architecture specifically designed for the development and implementation of a distributed simulation applications [25], defining the functional attributes, design rules and interfaces for simulation systems and specifying the communication between individual components. The cGEM5 in COSSIM is a customized GEM5 for lightweight and fast booting behavior of the image file run on GEM5. For this reason, the proposed simulator directly uses cGEM5 [9] for the distributed and parallel hetrogrnous simulation.

Based on the above related works and other references, we have designed the proposed simulator approach to tackle the simulation problems observed in distributed systems and parallel heterogeneous computing systems. We designed the NS3-cGEM5 integrated simulator to test and allocate synchronization time among simulated nodes. The next chapter deals with the design approach for the NS3-cGEM5 integration.

3. Method

3.1 HLA background

The set HLA federation input-output depends on formulator's attributes and objects that make federation using HLA-RTI tool [26]. HLA-RTI federation formation is well stated in detail on 'Improving the HLA-CERTI framework' [27]. HLA represents varieties of RTIs, and CERTI is the selected RTI for the proposed integrated simulator. It is an open source HLA runtime infrastructure that supports HLA 1.3 specifications [28] and uses C++ and Java programming languages for the processing.

3.2 NS3-cGEM5 integration Components

The use of either the architecture simulator or the network simulator alone, for simulating parallel heterogeneous and distributed system will not give a precise simulating mechanism of communication systems and architectural structures. The proposed alternative integrated simulator uses a network simulator and architecture simulator in a combined form to solve the simulation problem on distributed and parallel hetrogeneous system. The customized GEM5 (cGEM5) for architecture simulation, and NS3 for network representation, and communication facilities of distributed and parallel systems are selected for the proposed simulator. From the comparison tables, we can get a bit of information that NS3 has better features over other network simulators. These NS3 properties presented in **Table 5** and in the previous sections are the main reasons to select it as a component for the proposed integrated simulator. Details of NS3 and GEM5 are given in the previous section of related works.

Table 5. Network simulation tools comparison

Criterion	NS2	NS3	OPNET	OMNET++	QualNet
Interface	C++ OTcl	C++ Python	C, C++	C++	Parsec
GUI support	No	Limited	Yes	Yes	Yes
Parallelism	No	Yes	Yes	Yes	Yes
Documentation	Excellent	Excellent	Excellent	Good	Good
Scalability	Small	Large	Medium	Large	Very Large
Emulation	Limited	Yes	Not Direct	Limited	Yes
License	Open Source	Open Source	Commercial	Educational (Limited)	Commercial

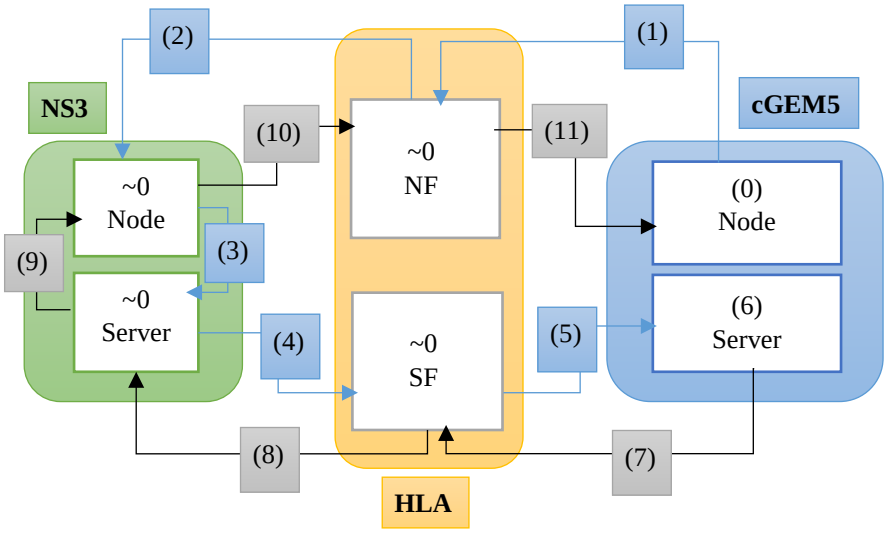
3.3 HLA CERTI Architecture

In CERTI, each federates process interacts locally with an RTIA (Run Time Infrastructure Ambassador) through a Unix-domain socket [29]. RTIA processes on exchange messages over the network, in particular with the RTIG (RTI Gateway) process, through TCP and/or UDP sockets. A specific role of RTIA is to immediately satisfy some federate requests, while other requests require network message sending or receiving. RTIA manages memory allocation for the message FIFOs (First In First Out) and always listens to both the federate and the network (RTIG). It has a significant role in the implementation of the tick function.

The RTIG (RTI Gateway) is a centralization point in the architecture. It has an essential role in managing the creation and destruction of federation executions and the publication/subscription of data. It plays a crucial role in message broadcasting which has been implemented by an emulated multicast approach. When a given message is received from a given RTIA, the RTIG delivers it to the interested RTIAs, avoiding true broadcasting.

HLA is a standard for distributed simulation and used when creating a simulator by combining (federating) several simulators. HLA was developed in the ’90s with US Department of Defense, later transitioned to become an open international IEEE standard [30].

In general, the independent nodes created in cGEM5 will communicate through HLA with the network communication help of NS3. Those nodes in cGEM5 have different architectural behaviour (heterogeneity), and these independent nodes need synchronization for harmonized tasks. The transaction flow of the proposed simulator is given in **Fig. 2**, and its detailed communication structure is depicted in **Fig. 5**.



NF: Node Federation, SF: Server Federation
~0: the operation is assumed to be performed in no time
(0) T_RqG: Request Generation Time
(1) T_GH: GM5 to HLA Communication Time
(2) T_HN: HLA to NS3 Communication Time
(3) T_NN: NS3 to NS3 Communication Time
(4) T_NH: NS3 to HLA Communication Time
(5) T_HG: HLA to GM5 Communication Time
(6) T_RsG: Response Generation Time
(7) T_GH: GM5 to HLA Communication Time
(8) T_HN: HLA to NS3 Communication Time
(9) T_NN: NS3 to NS3 Communication Time
(10) T_NH: NS3 to HLA Communication Time
(11) T_HG: HLA to GM5 Communication Time

Fig. 2. The tranaction flow and federation integration of the proposed simulator

3.4 Synchronization

After node creation occurs, synchronization among paired nodes will be done [31]. In order to have the synchronized simulation, we have allocated a waiting time till all nodes are ready to communicate federates, involved in the created federation and repeat the same pattern of execution periodically with Δt time step.

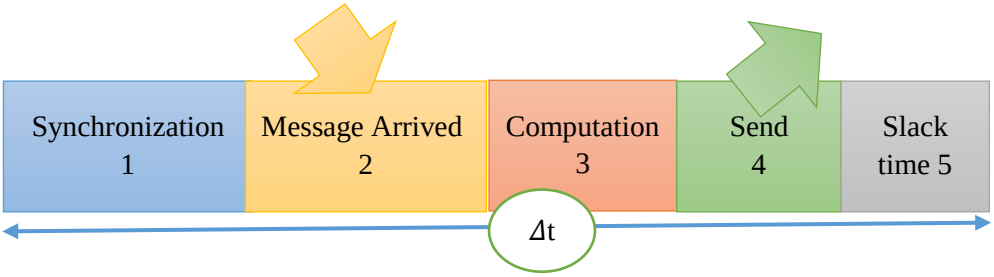


Fig. 3. Synchronization addition in periodic federate scheme

During each time step for the repeated execution, federates carry four phases: a reception, a computation, a transmission and a slack time phases. It is important to execute explicitly adding a synchronization phase to ensure the global coherent run time of the whole simulation [32].

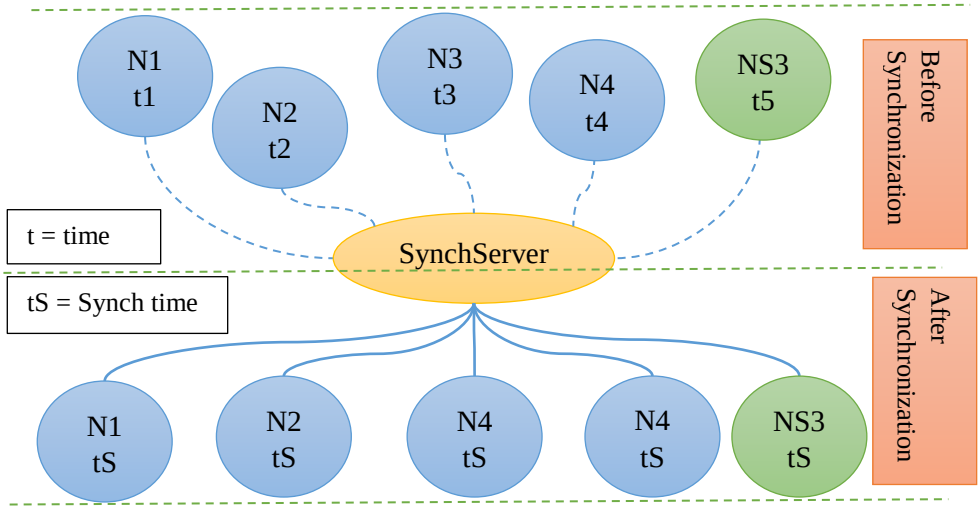


Fig. 4. Synchronization of NS3-cGEM5 integration for distributed system

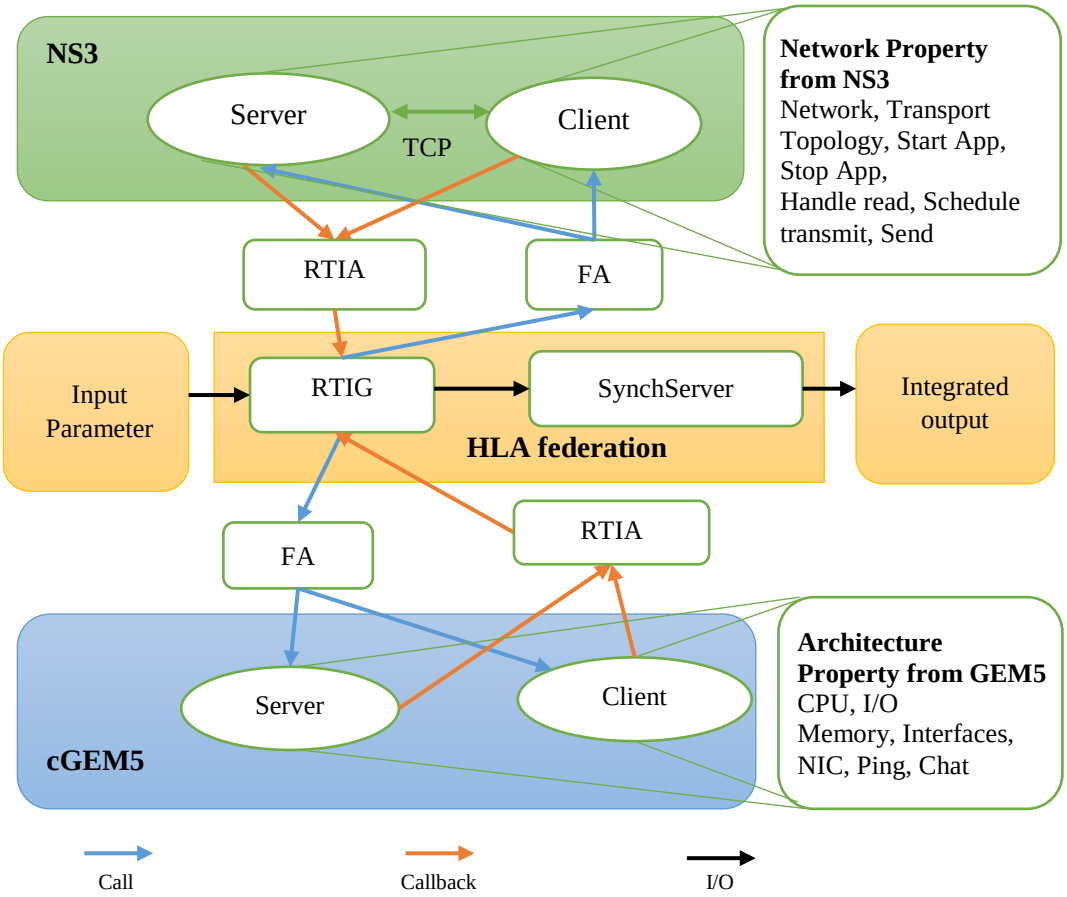
In the proposed NS3-cGEM5 integration, if we consider four independent cGEM5 nodes (N1, N2, N3, N4), then we can consider NS3 simulation node as an additional independent node, which gives us a total of five independent nodes (see **Fig. 4**) having different arrival time for synchronization. The synchronization server (SynchServer) creates a waiting point to make sure that all the five nodes have arrived. It allocates a waiting specified synch time until all nodes are available. When it gets information about all nodes arrival, it will release the nodes for their execution [33]. Then, it waits for the same period for the next synchronization and execution. Synchronization time (Synch time) of the system is allocated by the user.

3.5 Federation creation

In the federation creation processes, one of the federates is responsible for federation creation [34]. Then the federate itself joins the federation. After this process, the next federates joins the federation and the predetermined task execution will be done. This execution is followed by the release of the lately joined federation, and the creator will kill the federation. In the case of NS3-cGEM5 integration, NS3 is the federation creator, and cGEM5 will join the federation and get released first after completing the task assigned for the federation.

In **Fig. 5** there are ambassadors that play an impotrant role in HLA system. They are the federate ambassador and the RTI ambassador, these ambassadors are found beween HLA and the two federates side. Communications among client nodes, the server node and HLA created with the help of these ambassadors. The system call from HLA to federates passes through federate ambassadors and system callback from federates to HLA interaction returns through the RTI ambassador. The RTI ambassador is responsible for the communication of the federates and RTIG, that is, federates reach the RTIG through the RTI ambassador. On

the other hand, the federate ambassador is responsible for the communication of RTIG and the federates, that is, RTIG reaches to the federates through the federate ambassador. **Fig. 5** depicts these transaction in detail.



FA: Federate Ambassador, RTIA: Runtime Infrastructure Ambassador, RTIG: Runtime Infrastructure Gateway.

Fig. 5. Detailed node communication that shows the synchronization of NS3-cGEM5 integration for a distributed system. Ambassadors are responsible for communications between federates and HLA, and TCP applied for communications between NS3 nodes.

4. Experimental Tests & Results

In the integration of NS3-cGEM5, we used the following specified platform: Desktop CPU processor Intel ® Core™ i5-3570k CPU @3.40GHz processor speed, Linux Ubuntu 14.04, RAM 6GB, Storage size 1TB, NS3 (ns-allinone-3.19), and cGEM5 from COSSIM. After setting up the platform, we run the following experiments and got results.

To measure and compare the performance of the proposed integrated simulator, the following metrics are defined and used throughout this paper:

- Booting time: the time taken for each node created by NS3-cGEM5 simulator to be ready to operate after the execution command has been executed.
- Federation (Fed.) time: the very short time that takes for nodes from the two separate simulators, NS3 and cGEM5, to create a unified node through HLA-RTI system and

be ready to boot.

- Synchronization (synch) time: in the HLA system, synch time is defined as the time required for updating each federates updates and synchronizing to the system.
- Starting time: the maximum time taken for the node created by NS3-cGEM5 simulator to be ready to operate after the execution command has been executed. Nodes may have different starting time based on their ISA and other factors; as a parallel system, starting time determines the time of system communication.

4.1 Experimental Tests

We had set up the NS3-cGEM5 integration and performed the experimental tests to check synchronization time test. Synchronization time in HLA defines the time that each federates updates and gets synchronized to the system. Synchronization time is set by the user together with the architecture to run on the integrated simulations. The primary objective of this test is to know the impact of synchronization time on booting time and to figure out the optimal synch time that gives the best synchronization and minimum booting time.

Table 6. Experimental test result for synch time vs. booting time test

Synch time vs. booting time test							
Synch time [ms]	**Fed. time [s]**	**x86 Node 0 [s]**	**ARM Node 1 [s]**	**ARM Node 2 [s]**	**ARM Node 3 [s]**	**Avg. Total time [s]**	**Starting time [s]**
10	11	539	469	469	469	497.5	539
100	10	515	446	446	446	473.25	515
1,000	10	652	506	506	571	568.75	652
10,000	10	194	194	519	933	470	933
100,000	10	793	1,873	1,873	17,826	5,601.25	17,826

On these experimental tests, we allocted varied synchronization time, and measured the federation time, and booting time. Having measured values, we calculated the average booting time, and determined the total time required for booting a system. Note here that we considered the slowest booting time plus the federation time as a minimum booting time for the system, as a system it should have all architectures started.

We varied the synchronization time from 10 to 100,000 ms, which means that they will update to the SynchServer based on these synchronization times. In most simulators 10 ms synchronization time is considered as the default synchronization time. Based on our measurements, we had an analytical discussion as follows.

The experimental test focused on the synchronization of heterogeneous system of x86 and ARM combinations. Synchronization time is very vital for a distributed system. Regarding this test, **Fig. 6** shows an analytical property of federation time variation from 10ms to 100,000 ms of synch time. From **Fig. 6**, we have seen that an increase in synchronization time will not affect the federation time - which nearly 10 sec thorough out the synchronization time (almost identical). The federation time has negligible impact on booting time determination, so it is not a means to reduce booting time. **Fig. 7** depicts the the booting time of ARM and x86 architectures as a function of Synchronization. In the **Fig. 7**, we can observe a reduction in booting time at some point near 10,000 ms of synch time. This experimental result shows that repeated experiments and measurements are required to

determine the synchronization time. Synchronization time is varied on x86 and ARM image operating systems. Generally the booting time of ARM is expected to be lower than that of x86, and also this means that we can expect that the ARM operating system image is faster in booting time than that of x86. The experimental results showed faster booting time of ARM operating system image than x86 as expected, and some specific time of synchronization time (10,000 ms in **Fig. 7**) showed much faster booting time over all other sync time. And we can consider 10,000 ms as an optimal synchronization time for booting time.

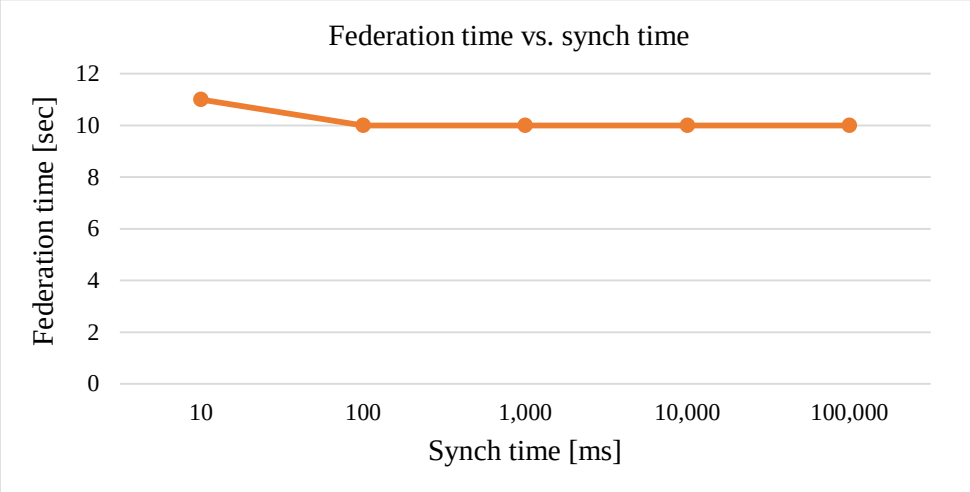


Fig. 6. Federation time as a function of synch time.

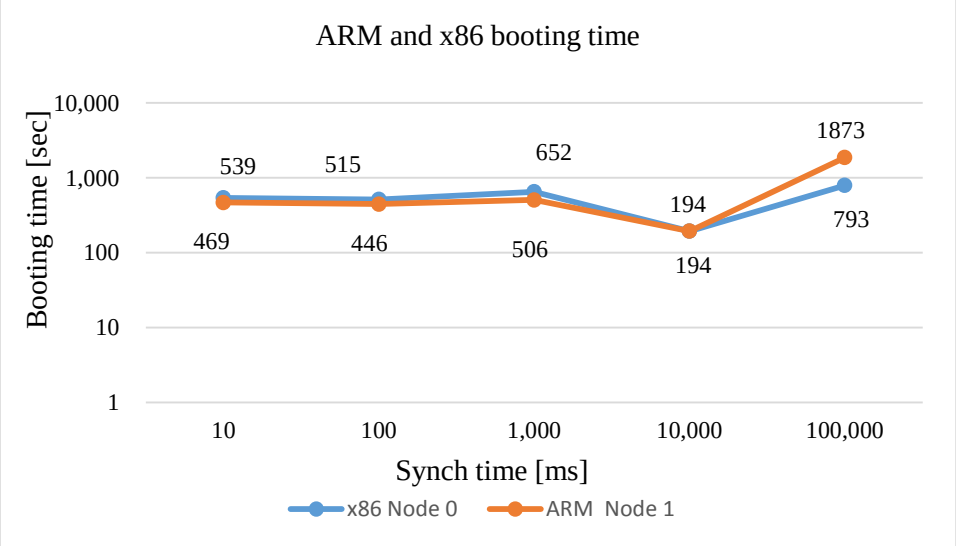


Fig. 7. ARM and x86 booting time vs. synch time

Based on our measurements, we have also plotted graphs about federation time, average total booting time, and starting time according to various synchronization time for x86 and

ARM. **Fig. 8** shows a collective view about measured time. In **Fig. 8**, we can note that the starting time of the integrated system simulation is significantly increasing for 100,000 ms and above synchronization time.

In the following analysis, we focused on the time window of synchronization time between 10 to 10,000 ms, and put aside the results for the time above 100,000 ms for future analysis, because synchronization time of 100,000 ms and above are too large and impratical for a real simulation run. Within the time window of interest, 100 ms and 10,000 ms of synchronization time is more appropriate time of synchronization for use in starting time (100 ms) and average total booting time (10,000 ms) point of view. Here, the federation time has negligable impact as compared to booting and starting time, because it is too small and almost constant for various synch time variations.

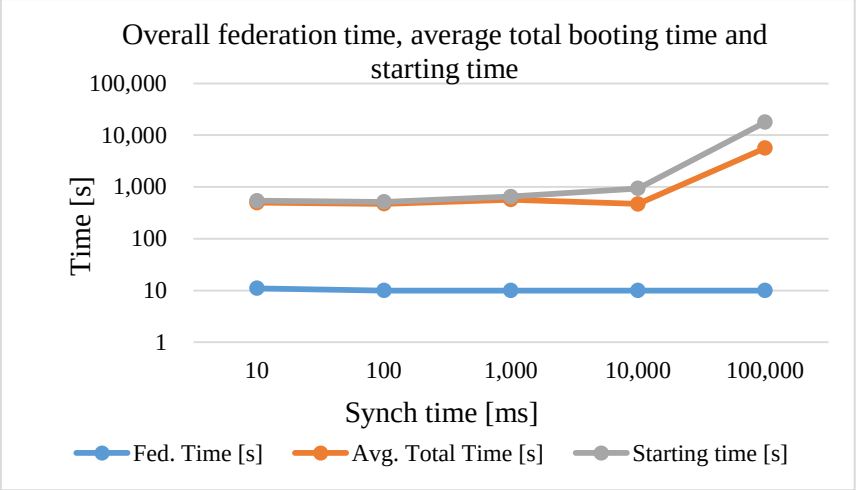


Fig. 8. Federation time, average total booting time and starting time as a function of synch time

We compared the proposed simulator with COSSIM and the result shows nearly similar behavior, especially for lower synchronizattion time. **Table 7** shows the measurement results of variation in booting time as a function of synchronization time between COSSIM and the proposed simulator. And the result is depcited in **Fig. 9** that shows the comparison between the two simulators.

Table 7. Starting time as a function of synchronization time in COSSIM and NS3-cGEM5 integrated simulators for ARM and x86 architectures

Synch time [ms]	COSSIM x86 [s]	COSSIM ARM [s]	NS3-cGEM5 x86 [s]	NS3-cGEM5 ARM [s]
10	565.12	487.095	539	469
100	468.26	413.3	515	446
1,000	432.53	364.52	652	506
10,000	389.34	180.57	194	194
100,000	411.91	193.26	793	1,873

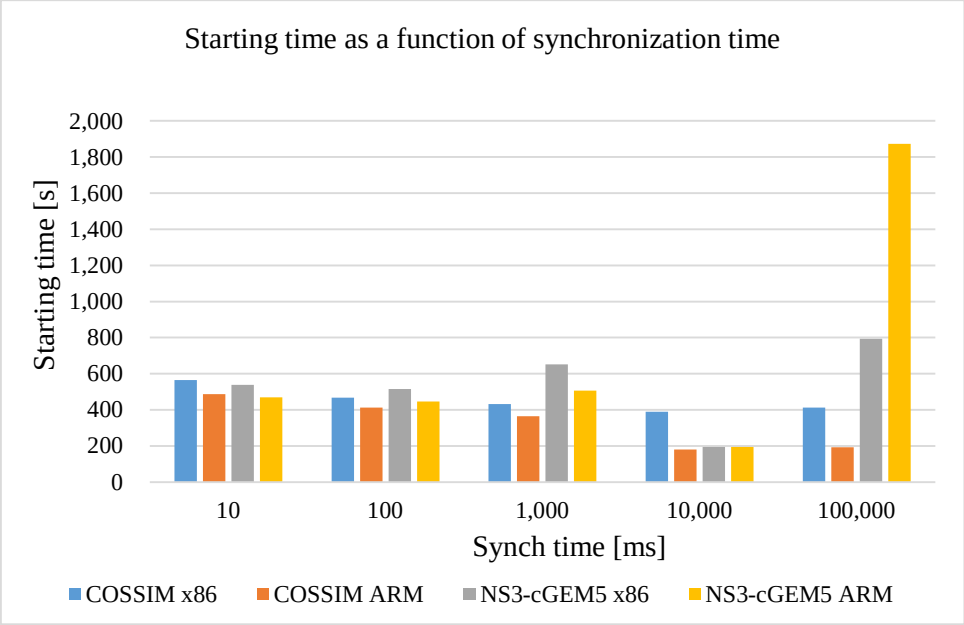


Fig. 9. Starting time as a function of synchronization time in COSSIM and NS3-cGEM5 integrated simulators

5. Conclusion and Future work

To tackle the challenges of synchronization time allocation in the integrated simulation of distributed and parallel heterogeneous systems, we have proposed an integrated simulator composed of NS3 and cGEM5 for appropriate synchronization time allocation.

From our tests, we simulated heterogeneous and homogenous systems with the variation of synchronization time to determine of booting time. We showed that integrated simulator is more capable of simulating heterogenous systems. And we also showed that a repeated simulation test is required to determine an appropriate synchronization time parameter for better and faster simulation.

From our measured tests we can conclude that the distributed system simulation is not easily achieved with single system simulator, like GEM5 and NS3 which are simulating architecture and network systems only. With the proposed NS3-cGEM5 integrated simulator, we performed various different architectural simulations (including homogenous and heterogeneous systems), and tested synchronization time effect on booting and execution time of parallelly coupled systems.

Generally, we can conclude that the proposed simulating system is capapble to allocating appropriate synchronization time for homogenous and heterogeneous systems. For parallel processes and distributed systems, the synchronizing server which depends on synchrionization time is vital in achieving synchronization for distributed system simulation of harmonized system communication. Our test results reveals that the minimum synchronization time does not always corrspond to the fastest in the booting time; there is an even smaller booting time and faster starting time in between. As a results, the selection of

the synchonization time parameter for integrated simulator needs frequent tests to decide the optimal synchronization time for the integrated simulator.

While testing and mesuring the integrated simulator, we found an unexpected increase in booting time in some nodes over 100,000 ms of synch time. Even the application of 100,000 ms of synch time in real simulation is not practical, the reason and mechnism of the behavior will be analyze in a future study.

ACKNOWLEDGEMENT

This work was supported by the Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2018-0-00503, Researches on next generation memory-centric computing system architecture).

References

[1] H. Singh and G. Singh, "Task scheduling in cluster computing environment," in *Proc. of 2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, Feb. 2015. Article (CrossRef Link)

[2] R. Arokia Paul Rajan, F. Sagayaraj Francis, "Dynamic Scheduling of Requests Based on Impacting Parameters in Cloud Based Architectures," in *Proc. of the 48th Annual Convention of Computer Society of India, Advances in Intelligent Systems and Computing, Springer International Publishing*, vol. 1, no. 248, pp. 513-521, 2014. Article (CrossRef Link)

[3] Matlab Documentation, "Resource Contention in Task Parallel Problems," 2018.

[4] Q. Kalim, M. Babar, H. K. Jawad and A. M. Sajjad, "Task partitioning, scheduling and load balancing strategy for mixed nature of tasks," *The Journal of Supercomputing*, vol. 59, no. 3, pp. 1348-1359, 2012. Article (CrossRef Link)

[5] N. Binkert, B. Beckmann, G. Black, SK. Reinhardt, A. Saidi, A. Basu, & R. Sen, "The GEM5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp 1-7, May 2011. Article (CrossRef Link)

[6] D. Skrien, "CPU Sim 3.1: A tool for simulating computer architectures for computer organization classes," *Journal on Educational Resources in Computing (JERIC)*, vol. 1, no. 4, pp. 46-59, Dec. 2001. Article (CrossRef Link)

[7] NS3 documentation, "NS3 Network Simulator," June 2018.[Online]. Available: Article (CrossRef Link)

[8] A. Mohammad, U. Darbaz, G. Dozsa, S. Diestelhorst, D. Kim, & N. S. Kim, "Dist-GEM5: Distributed simulation of computer clusters," in *Proc. of Performance Analysis of Systems and Software (ISPASS), 2017 IEEE International Symposium*, pp. 153-162, Apr. 2017. Article (CrossRef Link)

[9] A. Brokalakis, N. Tampouratzis, A. Nikitakis, S. Andrianakis, I. Papaefstathiou, D. Pau, E. Plebani, M. Paracchini, M. Marcon, I. Sourdis, P. R. Geethakumari, M. C. Palacios, M. A. Anton, & A. Szasz, "COSSIM An Open-Source Integrated Solution to Address the Simulator Gap for Systems of Systems," in *Proc. of 2018 1st Euromicro Conference on Digital System Design (DSD)*, pp. 115-120, Aug 2018. Article (CrossRef Link)

[10] A. Zarrad, & I. Alsmadi, "Evaluating network test scenarios for network simulators systems," *International Journal of Distributed Sensor Networks*, vol. 13, no. 10, pp. 1-17, Oct. 2017. Article (CrossRef Link)

[11] Z. Lei, Z. Ying, A. Chen and C. Liu, "A simulation platform for ZigBee-UMTS hybrid networks," *IEEE Commun. Lett.*, vol. 17, no. 2, pp. 293–296, Mar. 2013. Article (CrossRef Link)

[12] P. Gao, C. Jin, & G. Wang, "HLA-based distributed simulation model for multimodal operation system on container terminals," in *Proc. of System Simulation and Scientific Computing, ICSC 2008, Asia Simulation Conference-7th International Conference*, pp. 338-343, Oct. 2008. Article (CrossRef Link)

[13] R. M. Fujimoto, "Parallel and Distributed Simulation," *Handbook of simulation*, pp. 429-464. Article (CrossRef Link)

[14] NS3 Simulations, "List Of Network Simulators," June 2018. [Online]. Available: Article (CrossRef Link)

[15] Henderson, Thomas R., Mathieu Lacage, George F. Riley, Craig Dowell, and Joseph Kopena, "Network simulations with the NS3 simulator," *SIGCOMM demonstration*, vol. 14, no. 14, 2008.

[16] J. C. Gustavo, "NS3 module Network Simulator," June 2018. [Online]. Available: Article (CrossRef Link)

[17] P. Rajan, "Investigation of Network Simulation Tools and Comparison Study: NS3 vs NS2," *Journal of Network Communications and Emerging Technologies (JNCET)*, vol. 5, no. 2, pp. 137-142, 2015. Article (CrossRef Link)

[18] R. K. Atta ur, S. M. Bilalb, & M. Othmana, "A performance comparison of network simulators for wireless networks," in *Proc. of 2012 IEEE International Conference on Control System, Computing and Engineering*, 2012. Article (CrossRef Link)

[19] Research gate, "The difference between network simulators like NS2 NS3 Omnet Opnet and NETSIM," May 2018. [Online]. Available: Article (CrossRef Link)

[20] M. H Kabir, S. Islam, M. J. Hossain, & S. Hossain, "Detail comparison of network simulators," *International Journal of Scientific & Engineering Research*, vol. 5, no. 10, pp. 203-218, Oct. 2014. Article (CrossRef Link)

[21] J. Chen, L. K Dabbiru, D. Wong, M. Annavaram, & M. Dubois, "Adaptive and speculative slack simulations of CMPs on CMPs," in *Proc. of IEEE Computer Society. 43rd IEEE/ACM International Symposium on Microarchitecture*, pp. 523-534, Dec. 2010. Article (CrossRef Link)

[22] Y. N. Biruk, Z.Shin, H. Y. Kim, Y. W. Kim "Valuation of Microprocessor's and Network Simulation Technique," *Consumer Electronics-Asia (ICCE-Asia) sponsored by the IEEE Consumer Electronics (CE), Society and the Institute of Electronics and Information Engineers (IEIE)*, Nov. 2016. Article (CrossRef Link)

[23] GEM5 documentation, "The GEM5 Simulator A modular platform for computer-system architecture research," 2018. [Online]. Available: Article (CrossRef Link)

[24] B. Sascha & H. Andreas, "GEM5 Tutorial," 2016.

[25] T. W.Silva, D. C. Morais, H. G. Andrade, A. M. Lima, E. U. Melcher, & A. V. Brito, "Environment for integration of distributed heterogeneous computing systems," *Journal of Internet Services and Applications*, vol. 9, no. 1, Dec. 2018. Article (CrossRef Link)

[26] VT MAK, "The MAK RTI: HLA Run Time Infrastructure," 2017. [Online]. Available: Article (CrossRef Link)

[27] David Come, "Improving the HLA-CERTI framework," 2015. [Online]. Available : Article (CrossRef Link)

[28] CERTI documentation, "CERTI-Summary," June 2018. [Online]. Avalaible: Article (CrossRef Link)

[29] N. Eric, R. Jean-Yves, & S. Pierre, "CERTI, an Open Source RTI, why and how," in *Proc. of Spring Simulation Interoperability Workshop*, pp. 23-27, Mar. 2009 Article (CrossRef Link)

[30] B. Möller, KL. Morse, M. Lightner, R. Little, R. Lutz "HLA evolved–a summary of major technical improvements," in *Proc. of 2008 Spring Simulation Interoperability Workshop*, p. 1-7, 2009. Article (CrossRef Link)

[31] D. Rylan, "Synchronization in a Distributed System," Oct. 2013. [Online]. Available: Article (CrossRef Link)

[32] J. B. Chaudron, M. Adelantado, E. Noulard, & P. Siron, "HLA high performance and real-time simulation studies with CERTI," in *Proc. of 25th ESM-European Simulation and Modelling Conference*, pp. 24-26, Oct. 2011. Article (CrossRef Link)
[33] N. Neha Dalwadi, and C. Mamta Padole, "Comparative Study of Clock Synchronization Algorithms in Distributed Systems," *Advances in Computational Sciences and Technology*, vol. 10, no. 6, pp. 1941-1952, 2017. Article (CrossRef Link)
[34] HLA Group, "HLA RTI Synopsis," Nov. 1999.

Biruk Yirga Nidaw received B.S.in Electrical engineering from Adama University in Sep. 2010 Adama, Ethiopia. He is assistant lecturer in Adama Science & Technology University in Adama, Ethiopia. Currently he is a PhD candidate in an Integrated Program in Computer Software Department University of Science and Technology (UST) in Electronics and Telecommunications Research Institutions school (ETRI), Deajeon, Korea. His research interest focuses on system software, computer architecture and system interconnection design.

Myeong-Hoon Oh He received his PhD in information and communications engineering from Gwangju Institute of Science and Technology (GIST), Gwangju, Korea in 2005. He has been with ETRI, Daejeon, Korea since 2005 as a Principal engineer. He has been also an associate professor in University of Science and Technology (UST), Daejoen, Korea from 2006. His current research focuses on memory centric computing architecture, high-performance computing system design, cloud computing infrastructure and standardization. He also has been an editor for developing a Recommendation of cloud computing in ITU-T SG13.

Young Woo Kim received his BS, MS, and PhD in electronics engineering from Korea University, Seoul, Rep. of Korea, in 1994, 1996, and 2001, respectively. In 2001, he joined ETRI, Daejeon, Rep. of Korea. He is an associate professor at the University of Science and Technology (UST), Daejeon, since 2014. His current research interests are high-speed networks and supercomputing system architecture.