CAPITAL UNIVERSITY OF SCIENCE AND TECHNOLOGY, ISLAMABAD

SIM-Cumulus: An Academic Cloud for the Provisioning of Large-scale Network-Simulation-as-Service

by

Muhammad Ibrahim

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Faculty of Computing
Department of Computer Science

2018

SIM-Cumulus: An Academic Cloud for the Provisioning of Large-scale Network-Simulation-as-Service

By

Muhammad Ibrahim

(PC121003)

Foreign Evaluator 1

University/Institute

Foreign Evaluator 2

University/Institute

Dr. Muhammad Azhar Iqbal

(Thesis Supervisor)

Dr. Nayyer Masood

(Head, Department of Computer Science)

Dr. Muhammad Abdul Qadir

(Dean, Faculty of Computing)

DEPARTMENT OF COMPUTER SCIENCE

CAPITAL UNIVERSITY OF SCIENCE AND TECHNOLOGY

ISLAMABAD

2018

This PhD thesis is dedicated to all my family members, especially my father, mother, and wife who supported me during all the stages of my PhD.

Author's Declaration

I, **Muhammad Ibrahim** hereby state that my PhD thesis titled "**SIM-Cumulus: An Academic Cloud for the Provisioning of Large-scale Network-Simulation-as-Service**" is my own work and has not been submitted previously by me for taking any degree from Capital University of Science and Technology, Islamabad or anywhere else in the country/abroad.

At any time if my statement is found to be incorrect even after my graduation, the University has the right to withdraw my PhD Degree.

(Muhammad Ibrahim)

Registration No: PC121003

Plagiarism Undertaking

I solemnly declare that research work presented in this thesis titled "*SIM-Cumulus: an Academic Cloud the provisioning of large-scale Network-Simulation-as-Service*" is solely my research work with no significant contribution from any other person. Small contribution/help wherever taken has been dully acknowledged and that complete thesis has been written by me.

I understand the zero tolerance policy of the HEC and Capital University of Science and Technology towards plagiarism. Therefore, I as an author of the above titled thesis declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred/cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after award of PhD Degree, the University reserves the right to withdraw/revoke my PhD degree and that HEC and the University have the right to publish my name on the HEC/University website on which names of students are placed who submitted plagiarized work.

(Muhammad Ibrahim)

Registration No: PC121003

List of Publications

It is certified that following publication(s) have been made out of the research work that has been carried out for this thesis:-

1. **M.Ibrahim**, M.Azhar Iqbal, M. Aleem and M.Arshad Islam, "SIM-Cumulus: an Academic Cloud for the provisioning of Network-Simulation-as-Service," *IEEE Access*, 2018. (IF = 3.244)

2. M.Azhar Iqbal, **M.Ibrahim**, M. Aleem and M.Arshad Islam, "Amazon Cloud computing platform EC2 and VANET simulations," *Int. Journal of Ad hoc and Ubiquitous Computing*, in Press. (IF = 0.705)

3. **M.Ibrahim**, M.Azhar Iqbal, M. Aleem and M.Arshad Islam, "MAHA: Migration-based Adaptive Heuristic Algorithm for Large-Scale Network Simulations," *Cluster Computing*, under Review. (IF = 2.040)

Muhammad Ibrahim
(PC121003)

Acknowledgements

I am very grateful to my supervisor, Dr. Muhammad Azhar Iqbal (Department of Computer Science), for his timely advice, and support during the whole course of my PhD. His advice, encouragements and thorough knowledge positively impact my PhD research work. All members of PCN research group were very supportive and open in discussing ideas.

I would also like to thank Dr. Muhammad Aleem and Dr. Arshad Islam for their helpful guidance and reviews on my research papers and Dissertation. I recognize their encouragement and continues support during my research and thesis write up.

The Vice Chancelor of the University, Prof. Dr. Muhammad Mansoor Ahmed, Dean Faculty of Computing, Prof. Dr. Muhammad Abdul Qadir and Head of Department, Dr. Nayyer Masood are specially acknowledged. I would not forget to acknowledge my family members, colleagues, friends and all teachers of the department for their encouragement and helpful attitude.

Nothing is possible without the prayers and assistance of parents. I feel lucky to be at a stage that we were waiting for years and that is possible because of continues support of my parents.

Abstract

Large-scale network simulations are resource and time intensive tasks due to a number of factors i.e., setup configuration, computation time, hardware, and energy cost. These factors ultimately force network researchers to scale-down the scope of experiments, either in terms of Simulation Entities (SEs) involved or in abridging expected micro-level details. The Cloud technology facilitates researchers to address mentioned factors by the provisioning of Cloud instances on shared infrastructure. In this thesis, an academic Cloud SIM-Cumulus targeting the research institutions is proposed. The thesis is divided into three parts, each part discussing the contributions achieved in thesis.

The first part of this thesis discusses the design and implementation of SIM-Cumulus academic Cloud framework for the provisioning of Network-Simulation-as-a-Service (NSaaS). SIM-Cumulus provides the framework of Virtual Machine (VM) instances specifically configured for large-scale network simulations, with the aim of efficiency in terms of simulation execution time and energy cost. The performance of SIM-Cumulus is evaluated using large-scale Wireless Network simulations that executed sequentially as well as in parallel. Simulation results show that SIM-Cumulus is beneficial in three aspects i.e., (i) promotion of research within the domain of computer networks through configured Cloud instances of network simulators (ii) consumption of considerably fewer resources in terms of simulation elapsed time and usage cost and (iii) reduction of carbon emission leading towards sustainable IT development. The execution of simulation in parallel involves the partitioning of simulation model into several components and each component is assigned to separate execution units (Logical Processes (LPs)). Each LP is comprised of a set of SEs that can interact with local as well as remote SEs. However, the remote communication among SEs and synchronization management across LPs are the two main issues related to the parallel and distributed executions of large-scale simulations. A number of migration techniques are used to mitigate the problem of high remote communication and lead to a reduction in remote communication among SEs. However, most of the existing migration

strategies results in higher number of migration which lead to higher computation overhead. The second part of the thesis contributes Migration-based Adaptive Heuristic Algorithm (known as MAHA). MAHA provides dynamic partitioning of the simulation model based on runtime dynamics of the wireless network simulations. The proposed algorithm uses an intelligent heuristic for migration decision in order to reduce the number of migrations with an ultimate goal to achieve better Local Communication Ratio (LCR). The proposed algorithm is better in terms of achieving optimum LCR with reduced number of migrations as compared to the existing technique(s). The third contribution of this thesis is related to implementation of adaptive SIM-Cumulus (A-SIM-Cumulus) that integrates Advanced RTI System (ARTIS) and Generic Adaptive Interaction Architecture (GAIA) with the SIM-Cumulus framework. To obtain an insight into the performance gain, the simulation has been performed multiple times with different configurations and execution environments. The obtained results assert that the proposed algorithm significantly reduces the number of migrations and achieves a good speedup in terms of execution time for parallel (i.e., both multi-core and distributed) simulations on the A-SIM-Cumulus Cloud.

Contents

List of Figures

List of Tables

Abbreviations

ARTIS	Advanced Runtime Infrastructure System
CFI	Cloud Front-end Interface
CIML	Cloud Instance Management Layer
CIRM	Cloud Instance Resource Management
EC2	Elastic Computing Cloud
EHA	Existing Heuristic Algorithm
GAIA	Generic Adaptive Interaction Architecture
HeuC	Heuristic Cost
IC	Interaction Cost
LCR	Local Communication Ratio
LComm	Local Communication
LIR	Local Interaction Cost
LP	Logical Process
LRR	Local to Remote Communication Ratio
MAHA	Migration-based Adaptive Heuristic Algorithm
MF	Migration Factor
MigC	Migration Cost
$\mathbf{Mig}_{CPU}$	Migration CPU
$\mathbf{Mig}_{Comm}$	Migration Communication
MM	Middleware Management
M-VPL	Modified Virtual Platform Layer
OEC	Overall Execution Cost
PADS	Parallel And Distributed Simulation
PDES	Parallel Discrete Event Simulation

PH	Placeholder
PIL	Physical Platform Layer
RComm	Remote Communication
REST	Representational State Transfer
RDP	Remote Desktop Protocol
RIC	Remote Interaction Cost
SAL	System Accessibility Layer
SE	Simulation Entitiy
SIM	Simulation
SMF	Standard Migration Factor
SUC	State Updating Cost
SynC	Synchronization Cost
UWSNs	Under Water Sensor Networks
VANETs	Vehicular Ad-hoc Networks
VIL	Virtual Infrastructure Layer
VPL	Virtual Platform Layer
V2I	Vehicle to Infrastructure
V2V	Vehicle to Vehicle
WCT	Wall Clock Time
WLANs	Wireless Local Area Networks

Symbols

α	Set of Logical Processes (LPs)
β	Set of Simulation Entities (SEs)
γ	Number of SEs on each LP
Γ	Remote to Local Communication Ratio
LP_i	Set of SEs on a specific LP
LP_{PH}	Set of PHs on a Specific LP
N	Number of LPs
n	Number of SEs
SE_{LocInt}	SEs Remote Interacion
SE_{RemInt}	SEs Local Interaction
ω	Standard Migration Factor

Chapter 1

Introduction

1.1 Overview

The successful deployment of a new (i.e., wireless, mobile, and ad hoc) communication network always relies on the predicted behavior of a system's performance already obtained through the use of analytical, experimental, or simulation [1] techniques. Experimental techniques are expensive (both in terms of *time* and *resources*) whereas analytical methods are unable to fully grasp the characteristics associated with communications in wireless networks. Therefore, simulation techniques are extensively used during the design, implementation and performance analysis of a large number of wireless networks i.e., Mobile Ad hoc Networks (MANETs), Vehicular Ad-hoc Networks (VANETs), Urban Mesh Networks (UMNs), and Wireless Sensor Networks (WSNs) etc. Numerous simulation tools such as `NS-2` [2], `NS-3` [3], OMNeT++ [4] [5], OPNET [6], QualNet [7], DRMSim [8], Artery [9], JiST/SWANS [10] [11], DIVERT [12], NCTUns [13], iTETRIS [14], Castalia [15], SURE [16] etc.) are being employed by computer network researchers to quantify the performance of their proposed protocols for infrastructure-based and Ad hoc networks. However, the selection of an appropriate network simulator involves certain considerations, such as: ease in configuration, learning curve of the corresponding programming language, type of

communication system to simulate, provisioning of GUI environment, and support for scalability. These factors have also been acknowledged by the researchers in a survey (titled as *Working with Network Simulators* [17]). This survey analyzes the trends associated with the selection and usage of network simulators. The in-depth analysis in the survey has revealed two factors predominantly influencing the selection of a network simulator i.e., configuration complexity and scalability. Figure 1.1 shows an excerpt of the results on the time consumption for executing some standard tasks in network simulations as surveyed along more than 100 researchers from renowned universities. The survey results are shown in Figure 1.1. Figure 1.1 (a) is related to the researchers experience (i.e., in number of years) on any tool that they used for the network simulation. Time required for simulation environment setup is also important in the selection of a particular network simulator. The second part of survey is related to the time taken for initial installation and environment configuration as shown in Figure 1.1 (b). After initial configurations, the next important consideration is the time taken for the execution of available simulation example(s). Figure 1.1 (c) shows the survey results pertaining to the time taken by the researchers to execute simple network simulations. Most of the simulators are extended by patches to provide the functionality of certain scenarios. Figure 1.1 (d) is related to the time taken by researchers to install and configure customized patches for their required simulation. The obtained survey results reveal few important points: The time required for the initial simulator configuration and setup is quite high and is normally ignored. Users of the simulation often face difficulty in executing the very basic example. The last and very important point is that the researchers often face difficulty and waste sufficient time in configuring customized patches and running their required simulation.

The intended network simulation can be small-to-medium as well as large-scale, having large number of nodes. The computation complexity of the simulation is dependent on the granularity of details (i.e., macroscopic, microscopic or mesoscopic) involved in the simulation modeling [18]. The macroscopic details highlight the big picture of the system to be simulated whereas the microscopic approach considers more details about the simulation model. The mesoscopic simulation

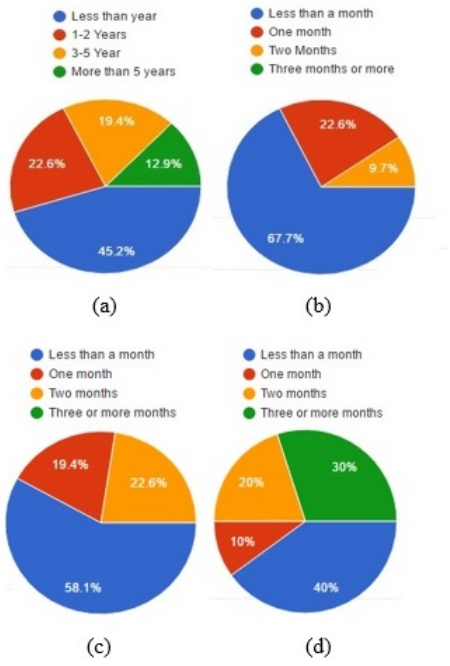

FIGURE 1.1: (a) Researchers' experience with network simulators, (b) Time taken to configure network simulator, (c) Time taken to execute basic example, (d) Time taken to execute simulation with available frameworks/patches.

analyzes intermediate details about simulation model. For example, in VANETs, macroscopic, microscopic, and mesoscopic details are related to complete road flow, individual vehicle details and transportation elements in small groups, respectively. An inadequate amount of details can lead to incorrect results during performance evaluation [19]. The correctness of the simulation results highly depend on the amount of details required to represent the simulated model [20]. However, simulations with microscopic parameters can lead to high memory and computation requirements to simulate a modeled system. Moreover, the highly dense simulation with granular details also attribute to more time required for simulation execution.

Simulations (either microscopic or macroscopic) can be executed in two different ways (i.e., sequential and parallel). The sequential approach for network simulation execution is based on monolithic design, where a single execution unit is in-charge of managing all parts of the simulation. The monolithic approach is more realistic for small and medium scale network simulations [21]. On the other

hand, scalable wireless network simulations, with microscopic details are impractical to execute on monolithic systems. Generally, microscopic details of a wireless network simulation model ascertain the accuracy of simulation results and are essential for the successful deployment of the proposed model. Concerning the monolithic systems, memory scarcity and low computing power are two important factors that limit the scale of network simulations [22]. This is due to the fact that memory requirements increase exponentially (as representation of each nodes state requires a certain amount of memory) and simulations takes long time span to be completed. Moreover, numerous interactions (i.e., event, packets) are required to be created, stored and delivered instantly [21], [23].

Alternatively, Parallel And Distributed Simulation (PADS) [24] approaches effectively deals with the execution of large-scale network simulations. In PADS, the simulation is segregated into several parts and each part is assigned to specific logical unit (LP). These LPs may reside on same or different Physical Execution Unit (PEU). The PADS has the capability to employ a huge amount of memory and computation resources from different PEUs. For precise simulation execution using PADS, all LPs are required to be synchronized during the entire span of the simulation. The initial employed partitioning scheme is based on the static allocation of SEs on a number of LPs. The goal of this scheme is two folds: the load on different LPs should be balanced, keeping the remote communication (among SEs located on different LPs [25]) as minimum as possible. However, the static partitioning approach embroils two issues that are: latency overhead of remote communication (inter-LP communication for parallel execution) and synchronization management among the LPs. The nodes in the wireless networks are mobile in nature and frequently move due to high mobility involved. This leads to high remote communication and thus every form of static partitioning may not be an optimal choice. The partitioning problem become more complex due to a number of factors that must be taken into account. The LPs in the simulation may not be homogeneous [23] in nature and the communication pattern may vary during the course of the simulation as the transmission of message by the SEs is random in wireless networks. Moreover, the execution architecture may comprised of several

components that can be heterogeneous (i.e., both in terms of hardware used and performance).

In contrast to the static partitioning techniques, the dynamic partitioning are proposed that keep the load balanced on LPs and reduce high remote communication [26]. The dynamic partitioning techniques use the concept of SE migrations from one LP to another with the aim to reducing the high remote communication. The migration is triggered based on the condition specified by heuristics that are evaluated for each SE at their respective LPs. The local available information is used for the migration decision. The migration mechanism is an attempt to localize the communication among the SEs as much as possible and to keep the load balanced on different LPs. Moreover, the migration mechanism has a certain computation cost which is crucial for achieving good performance. The migration approach will be effective if the cost associated with employing the migration is lower than the remote communication cost. Another important consideration is related to the number of migrations which is required to minimize the remote communication among SEs located on different LPs. The type of network chosen in this research thesis is wireless networks. The nodes in the mobile wireless networks are in movement due to the mobility involved. This means that the communication pattern among the simulation nodes changes continuously during the course of the simulation. The required number of migrations also raise frequently due to the continues node mobility. Thus more computing and memory will be required to achieve optimal LCR (Local Communication Ratio). The LCR corresponds to the ratio of the total number of local communications by all SEs in a given LP, with respect to all interactions originating from this LP [23]. The suggested approaches [23], [27], [26] are appropriate to handle the networks with no or very low mobility nodes. The high mobility involved in these type of mobile wireless networks will lead to more frequent migration which attribute huge migration cost, ultimately leading to poor simulation performance.

PADS simulations could be implemented on multiple platforms, such as grid computing [28], High Performance Computing (HPC) [29] [30], Cloud computing [27] etc. In recent advancements, Cloud computing eliminates the barriers of resource

limitations for large-scale applications [31]. The implementation of PADS over Cloud computing provides an essential platform to address the problems encompassing scalable simulations with microscopic details.

1.2 Problem Statement

The required amount of memory and computation time (for each simulation run) are two important factors that limit the scale of packet-level simulations. In case of scalable network simulations, memory requirements increase exponentially as representation of each nodes state requires a certain amount of memory. Analysis of large-scale networks and realistic scenarios through computer simulations on stand-alone machines becomes a time consuming process due to prolonged elapsed-time of simulation. A number of schemes i.e., Grid computing [32], Agent-based solution [33], [34], [35], [36], [37], HPC [29], and Cloud computing [38–41] are being employed by the researchers to attain the demands of large-scale simulation. Grid computing is designed for large-scale distributed data processing/storage and works with assigning subtasks across different clusters [32], [42]. The grid computing environment is best suitable choice for specialized type of large-scale computing tasks and often inefficient for network simulations as the components of the grid are loosely coupled and thus the synchronization among them is quite complex. Agent-based simulation approaches [33–35, 43–45] are suitable to effectively deal with situations where each agent has been independently assigned different set of tasks. The agent-based approaches are not suitable for large-scale network simulation tasks due to the frequent synchronization required among different agents. In addition, agent-based approaches require HPC to support the network simulations. Nevertheless, independent task assignment in case of HPC environment is restricted and requires pre-training to be configured to meet the simulation requirements [29]. On the other hand, in recent times, Cloud computing provides solution encompassing large-scale problems including the domains of storage, computational intensive tasks, real-time applications, and simulation jobs [38–41] etc. Due to the complex nature of large-scale network simulations and

limitations of network simulators; to-date only few Cloud solutions (e.g., SEMsim cloud service, CloudSME etc.) are available that are specifically designed for large-scale simulation jobs. The problems related to the existing Cloud approaches for large-scale network simulations are two folds. First, for large-scale network simulations, the existing Cloud frameworks have considered specifically designed network simulators (encompassing a specific network scenario e.g., urban scenario) and are only compatible with proposed Cloud architecture. On the other hand, most of multipurpose Clouds (i.e., Amazon Web Services (AWS), Google Cloud etc.) provide various type of services at generic level (i.e., infrastructure, storage, software service, and platform) and are difficult to be customized for network simulation services. Although, these Clouds are quite powerful in terms of available resources; however, usefulness of the existing Cloud platforms is questionable because of difficult integration with the existing state-of-the-art simulation tools (i.e., NS-2, NS-3, OMNeT++ etc.). Second, the Parallel and Distributed Simulation (PADS) approaches are used for large-scale network simulations. PADS approaches partition the large-scale wireless network simulation into a number of components called Logical Processes (LPs) and each LP is assigned part of the simulation model (i.e., equal number of Simulation Entities (SEs). The SEs in the wireless networks are mobile in nature and thus involve high remote communication cost due to high mobility of the SEs that ultimately increase simulation execution time. A number of migration and load-balancing techniques have been proposed to reduce the high-end remote communication for large-scale wireless network simulations [23], [27], [26], [23]. Although, the existing approaches lead to a good level of Local to Remote Communication Ratio (LCR); however, huge number of migrations (in wireless networks) may lead to additional overhead and producing a negative impact on the overall performance gain. The higher number of migrations make these approaches unsuitable for large-scale wireless network simulations involving high mobility.

1.3 Research Objective

The following research objectives which are intended to be achieved in this research thesis as follows.

- analysis of state-of-the-art approaches to perform large-scale network simulation by harnessing the contemporary approaches;
- design and implementation of an Academic Cloud that solely employed for small to medium scale network simulations and more specifically for large-scale wireless network simulations;
- implementation of dynamic partitioning (migration-based approach) algorithm for large-scale network simulations using PADS approach to minimize the number of migrations and to obtain time-efficient simulation execution;
- facilitate the students/faculty/researchers by providing them a Cloud-based network simulation service that is efficient in terms of memory, time, cost and accuracy.

In addition, the proposed Cloud also helps in maintaining sustainable green IT.

1.4 Research Contribution

The contributions of this thesis are discussed as follows.

- This thesis provides a comprehensive and detailed survey of the existing techniques used for large-scale network simulation. In this work, the detailed literature survey served as a baseline for designing and deploying the proposed academic Cloud framework.
- An academic Cloud framework named SIM-Cumulus is designed and implemented to support large-scale wireless network simulations. The proposed

Cloud framework provides the solution to facilitate the student/faculty researchers in validating their research work with an ease without facing the complexity of configuring the simulation environment setup. Moreover, the employment of SIM-Cumulus reduces the time required for simulating large-scale network simulation.

- Contributed Migration-based Adaptive Heuristic Algorithm (MAHA) that reduces the high remote communication of SEs across PADS for both multi-core and distributed architectures.

- A set of simulation experiments are executed to show the effectiveness of the proposed Cloud and migration algorithm. Results reveal that SIM-Cumulus assist the researchers in various ways such as provisioning high performance computing, lowering the costs of IT infrastructure for academic institutions and reducing power consumption that will pave the way towards sustainable green IT.

1.5 Research Evaluation

To evaluate the performance of large-scale network simulations, we have considered three different ad hoc network simulation scenarios (i.e., Vehicular Ad hoc Networks (VANETs), Underwater Wireless Sensor Networks (UWSNs), and Wireless Local Area Networks (WLANs). The VANETs scenario is considered because of its computation complexity due to the involvement of a number of factors i.e., realistic mobility traces, nodes mobility, propagation model, heterogeneous access technologies, packet rate in routing protocols etc. These factors with microscopic details are associated with each node in the simulation scenario. Therefore, the memory consumption and computation time increase exponentially with addition of each vehicular node in the simulation scenario. Various platforms (details are available in Table 5.2 in revised version of Submitted thesis) with diverse hardware specifications have been used to evaluate the performance of large-scale network

simulations in terms of execution time, simulation speed, and energy consumption. Large-scale UWSNs scenario is considered to evaluate the effectiveness of proposed academic Cloud SIM-Cumulus in terms of simulation execution time utilizing monolithic and parallel execution modes. Like VANETs, the UWSNs scenario considers three aspects i.e., mobility, propagation model and routing that ultimately increase the computational complexity of the network simulations especially in the case of large-scale deployment. Large-scale WLAN scenarios have been taken into account to evaluate and compare the performance of MAHA and EHA approaches. The ARTIS/GAIA framework has been selected (for the implementation of large-scale WLAN scenarios), which supports the dynamic migration of simulation entities (SEs) and is specifically designed for the execution of large-scale Parallel and Distributed Simulations (PADS).

1.6 Thesis Organization

The rest of the thesis is structured as follows (also shown in Figure 1.2): Chapter 2 provides details regarding large-scale network simulation issues and existing approaches to handle large-scale network simulation. The discussion related to the existing popular Academic Clouds is also part of this section. Moreover, the review of partitioning (i.e., static and dynamic) approaches used to cope with high remote communication is presented at the end of Chapter 2. The Chapter 4 delineates the contributions of this thesis (i.e., SIM-Cumulus, MAHA and A-SIM-Cumulus). The proposed work is divided into three parts that are: First part presents SIM-Cumulus architecture overview, working details, mathematical model and cost analysis pertaining to the large-scale network simulations. A-SIM-Cumulus architecture and working details is presented in second part of Chapter 4. The discussion related to the proposed migration algorithm (i.e., MAHA) is presented in last part of Chapter 4. Chapter 5 presents the experimental setup details for large-scale wireless network simulations. The results and discussion related to the large-scale network simulation and migration implementation are also part of

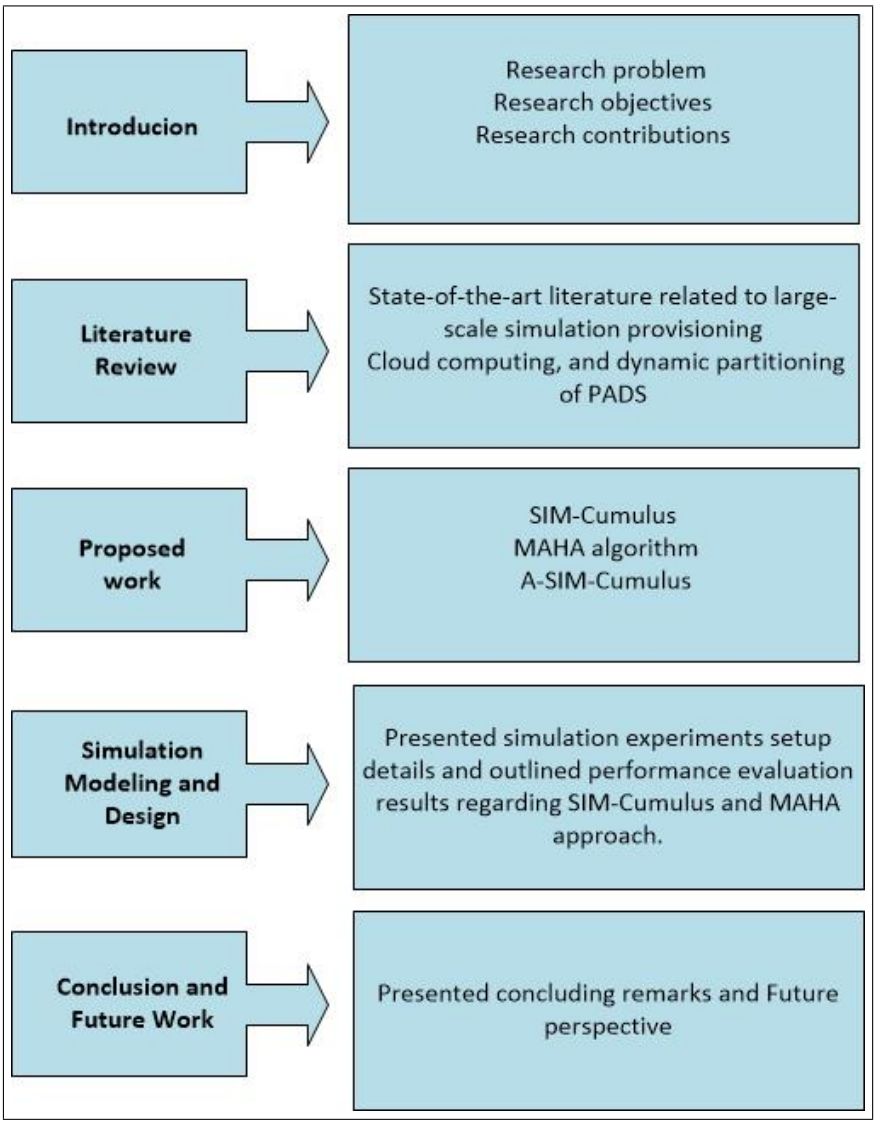

FIGURE 1.2: Thesis structure.

Chapter 5. The conclusion of the study is presented in Chapter 6 along with the future perspectives of this thesis.

Chapter 2

Literature Review

The wireless systems are often very complex and may comprise of large number of nodes. The discovery of new wireless applications is compelling researchers to meet new and very stringent requirements for its simulations that demand the involvement and integration of traffic generators, map generators, mobility models and communication models. Moreover, the computation complexity of scalable wireless network simulation is increased due to the involvement of mobility models and heterogeneous access technologies e.g., accurate channel modeling, encoding techniques and routing protocols [46, 47] etc. The complex nature, expensive experimentations, and limitations of conventional analytical evaluation methods adhere the espousal of network simulation for pre-deployment performance evaluation of scalable communication and wireless network systems [21]. However, the simulations of such large-scale complex wireless network systems are resource and time intensive tasks due to a number of factors i.e., setup configuration, computation time, hardware, and energy cost. Many solutions are proposed to cope with large-scale simulations. One of the solutions can be the use of simulators that particularly target the large-scale wireless simulations e.g., `DIVERT` [12], `iTETRIS` [14], and `NCTUns` [13], `VNS` [48], `DRMSim` [8], `Artery` [9]. However, the problem related to the high computing time for multiple simulation runs on a single workstation still persists. As an alternative, PADS [24] approaches are used that effectively executes the large-scale simulations. In PADS, the simulation is segregated into different

parts and each part is assigned to specific logical unit (i.e., called LP). These LPs may reside on same or different PEU. The PADS has a capability to employ a huge amount of memory and computation resources from different PEUs. For precise simulation execution using PADS, all LPs are required to be synchronized during the whole span of a simulation. PADS simulations could be implemented on multiple platforms, such as grid computing [32], HPC [29], Cloud computing [27] etc. The nodes in the wireless networks are mobile in nature and frequently move due to high mobility involved. This leads to high-end remote communication and thus every form of static partitioning may not be an optimal choice. The partitioning problem becomes more complex due to a number of factors that must be taken into account. The LPs in the simulation may not be homogeneous [23] in nature and the communication pattern may vary during the course of the simulation as the message sending by the SEs is random in wireless networks. Moreover, the execution architecture may be comprised of several components that can be heterogeneous (i.e., both in terms of hardware used and performance). In contrast to the static partitioning techniques, the dynamic partitioning was proposed to keep the load balanced on LPs and reduce high end remote communication [26]. The dynamic partitioning techniques use the concept of SE migrations from one LP to another with the aim of reducing the high remote communication. The discussion related to the provisioning of large-scale wireless network simulation is presented in section 2.1. Section 2.3 delineates the work related to the discussion of academic clouds for the provisioning of large-scale network simulations. The discussion pertaining to simulation partitioning of large-scale wireless networks is presented in section 2.4. In addition, section 2.4 also discuss the state-of-the-art static and dynamic partitioning techniques.

2.1 Large-scale Wireless Network Simulations Provisioning

Recent innovations in internet-based systems have attracted network researchers towards the adoption of web-based simulation environments [49]. The notion of web-based simulation design and modeling is initiated by Lorenz et al. in [50] where authors have presented prototypes of web-based simulations. The proposed approaches provides the user web-based interactive interface to perform their simulation experiments and visualize the obtained results. The author discussed the basic components and standards used for the adoption of web interfaces for managing simulation. The users of the simulation can either run the existing model or change the simulation model according to their requirements. Taylor et al. [51] have discussed research efforts related to the exploitation of web infrastructure for online simulation modeling, design, and result visualization. The author highlighted the issues involved in using virtualized environment for modeling and simulation. Moreover, discussed the challenges associated with the integration of Internet of things (IoT) with existing modeling and simulation approaches. All of these schemes have emphasized the classical way of offering web interfaces for controlling simulations. Nevertheless, emerging trends of large-scale simulations demand the distribution of simulation jobs over physical and virtual computing resources for load balancing [29]. A number of schemes i.e., Parallel processing [52], [53], Grid computing [32], Agent-based solution [33], [34], [35], [36], [37], Distributed computing, HPC [29], and Cloud computing are being employed by the researchers to attain the demands of large-scale simulation.

Parallel processing [52], [53] is used to perform large-scale simulations via running multiple copies of the simulator (each representing a different portion of the simulation) in parallel. The author in [52] proposed parallel computing framework for the simulation of large-scale mesoscopic transportation system. A synchronous space parallel simulation approach is presented where different parts of the simulation model are assigned to different LPs. Figure 2.1 shows how different LPs are assigned to different CPUs according to required partitioning strategy.

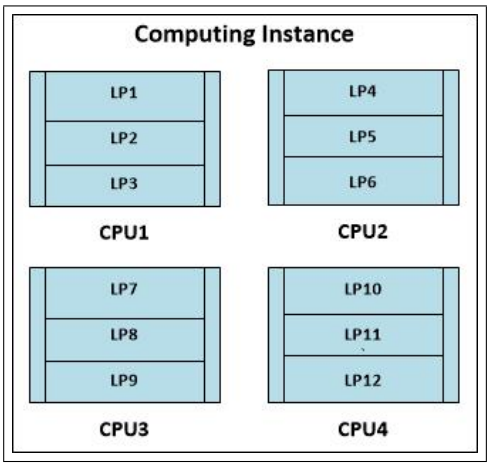

FIGURE 2.1: LP distribution on parallel computing instance.

Grid computing is designed for large-scale distributed data processing and storage, and works with assigning subtasks across different clusters [32], [42]. In [32], the author presented the applications of smart grid used for electric, gas, and water usage across the Austin city. The paper also discussed different monitoring methods used for collecting residential energy consumption. The grid computing environment is most suitable choice for specialized type of large-scale computing tasks and often in-efficient for network simulations. Agent-based simulation approaches [33–35, 43–45] are suitable to effectively deal with situations where each agent has been independently assigned different set of tasks. The author presented the tools required for the implementation of agent based modeling and simulation and discussed different applications of agent-based simulation [33]. The author in [45] evaluate how the agent-based simulation tools can offer value able services in the modeling and simulation of large-scale complex networks such as wireless mesh networks, VANETs, UWSNs, large-scale peer-to-peer systems,and networks involving considerable human interaction. Moreover, an agent-based approach requires HPC to support the simulations. However, HPC environment is restricted and requires pre-training to be configured to meet the simulation requirements [29]. Figure 2.2 shows the generic structure of agent-based approach where each agent has either synchronized locally or globally with agents that resides on remote agent platform.

In recent times, Cloud computing is used to solve large-scale problems including

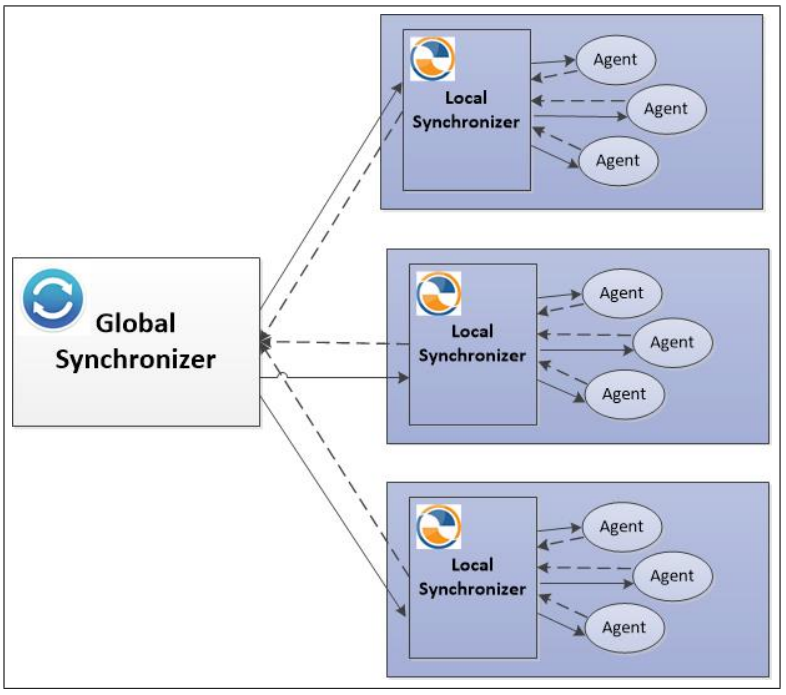

FIGURE 2.2: Generic structure of Agent-based approach.

storage, computational intensive, real-time response applications, simulation [38–41] etc. The following section explains the effectiveness of Cloud computing for solving large-scale simulation.

2.2 Cloud computing

The evolutionary concept of Cloud computing has long history dating back to the era of 1970's. The emergence of various technologies in chronological order upon which Cloud computing concept is essentially based as shown in Figure 2.3. The paradigm of Cloud computing provides an environment that facilitates sharing of resources in the form of scalable infrastructure, platform, and value-added applications for business and academia on utility basis [54]. Nevertheless, from plethora of available definitions, the Cloud can defined as: *the straightforward thinking about Cloud computing is just an imagination of a colossal web data center where integrated hardware and virtualized software resources offer a variety of on-demand, elastic services in terms of infrastructure, platform, and software to the end users*

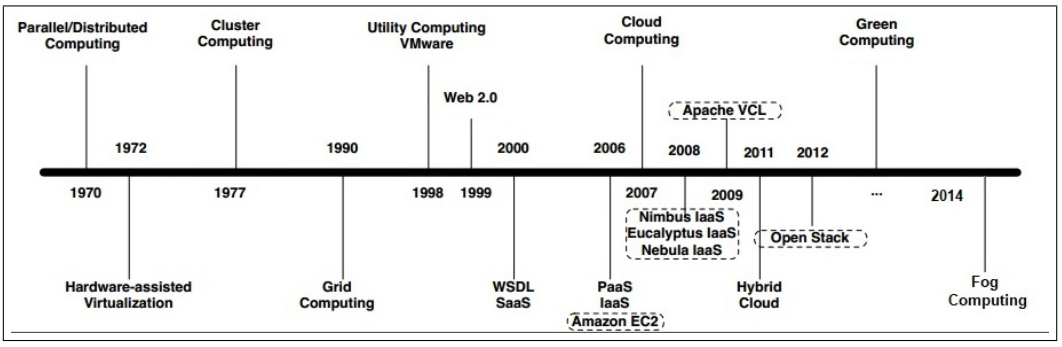

FIGURE 2.3: Evolution of Cloud and related technologies.

on both utility and non-utility basis through abstract web interfaces, to facilitate business organizations as well as academic institutions.

Cloud computing has been in use in various fields of life including health care, business, data analytics, academia etc. The business Clouds [55] are related to the availability of infrastructure and usage of information technology advancements with a clear objective of increasing business productivity. Cloud computing with its benefits (i.e., low IT infrastructure cost, performance, scalable storage, availability, green computing, remote access, better response time etc.) opens new opportunities for small-scale and large-scale businesses. A number of business cloud models have been proposed and currently organizations focal concern is related to the selection and adoption of an appropriate business model to achieve sustainability [56]. Healthcare cloud [57] provide the environment for all the medical and health units to make use of the cloud computing conveniently in way to reduce the usage costs and make some improvements in medical standards. Authors in [27] have favored the idea of Cloud computing paradigm for dynamic sizing of the large-scale network simulations. Guo et al. [58] proposed a five layered framework for systematic support of *Simulation Software as a Service* (SIMSaaS) in Cloud. According to Cayirci [59], the cloud computing has been considered as an important model to support modeling and simulation for most of military and civilian applications. The provisioning of Cloud toolkits i.e., CloudSim [60] and CloudnetSim++ [61] have enabled researchers and professionals to perform modeling of data centers and virtual machines, while managing the energy efficiency in Cloud. In [62], the author proposed a layered hierarchical model GP-CMW that works

to manage meta computing nodes in simulation process. GP-CMW is a kind of simulation model that works well in both parallel as well as distributed simulations with diverse nature of computing nodes having GPUs and CPUs. All these issues are tried to resolve through implementation a three layered GP-MW model for simulation. After analyzing this approach it come to notice that it is efficient in intensive situations with multi core GPUs and CPUs. In terms of scalability it is also more effective than other proposed models. Through exploiting multiple resources for computing this model not only overcome the scalability issues, it also make the communication process 3.6% faster among heterogeneous sources involved in simulation. Rahman et al. [63] have compared various Cloud simulators and highlighted their strengths and limitations. In addition, the authors have proposed Cloud simulator *Nutshell*, which offers realistic Cloud environments and protocols. Ficoo et al. [64] have devised a method to cope with simulation of large-scale critical systems on private Cloud. Gabriele [23] proposed an adaptive solution for large-scale simulations using parallel and distributed simulation (PADS) approach. CloudSME [65] was proposed as a Cloud-based simulation framework for large-scale simulation of manufacturing and engineering industry. In [66] the author has proposed an E-Learning cloud framework that virtualizes the physical machines and then performs allocation to the clients on demand for E-Learning systems. Recently, Academic Clouds [67] [68] [69] grab attention from the researchers due to its versatile services. The discussion about Academic Cloud is presented in next section.

2.3 Academic Clouds

The Academic Clouds are intended to facilitate the provisioning of infrastructure and usage of technology to solve problems related to research and IT establishment. The emergence of various technologies in chronological order upon which Cloud computing concept is essentially based are shown in Figure 2.3. However, Cloud computing is a merger of several of its precursor technologies (i.e., Grid, Cluster etc.). Many people from different backgrounds have been involved in the

TABLE 2.1: Approaches to support network simulation.

Paper	Approach	Large-scale simulation support	Web-interface	Elastic
[52]	Parallel processing	Yes	No	No
[32]	Grid computing	Yes	No	No
[33]	Agent-based	Yes	Yes	No
[27]	Cloud computing	Yes	Yes	Yes
[64]	Private cloud	Yes	Yes	Yes
[67]	Academic Clouds	Yes	Yes	Yes

development of Cloud; therefore, research community has not been able to develop a consensus on a common definition for Cloud [70].

The usage of Cloud technology for academia is threefold as shown in Figure 2.4:

- Adoption of Cloud computing technology by educational establishments;
- Provisioning of high computing power to facilitate research work within academic institutions;

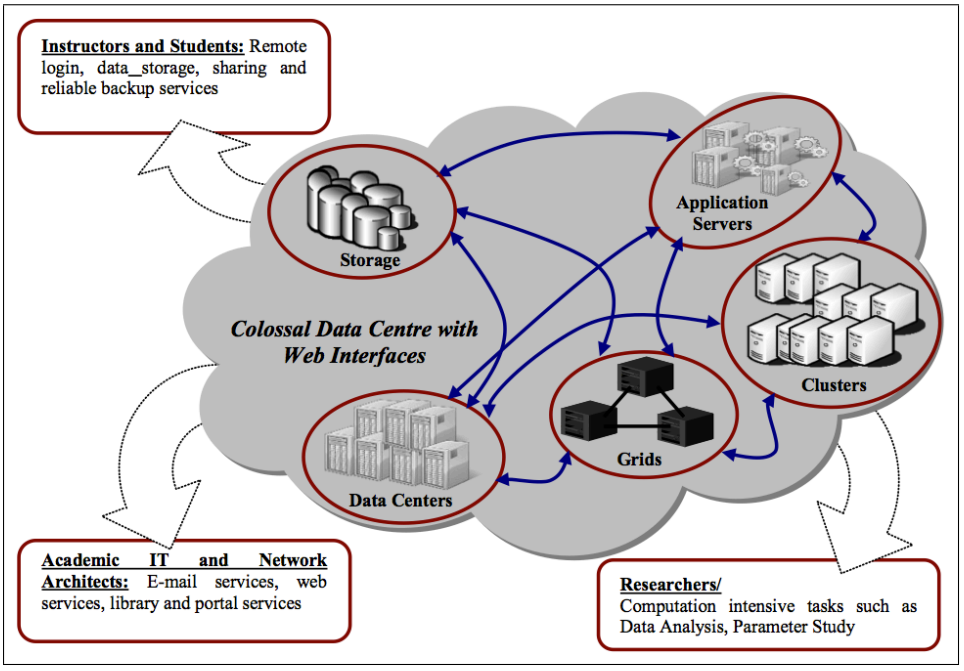

FIGURE 2.4: Users' academic cloud perspective.

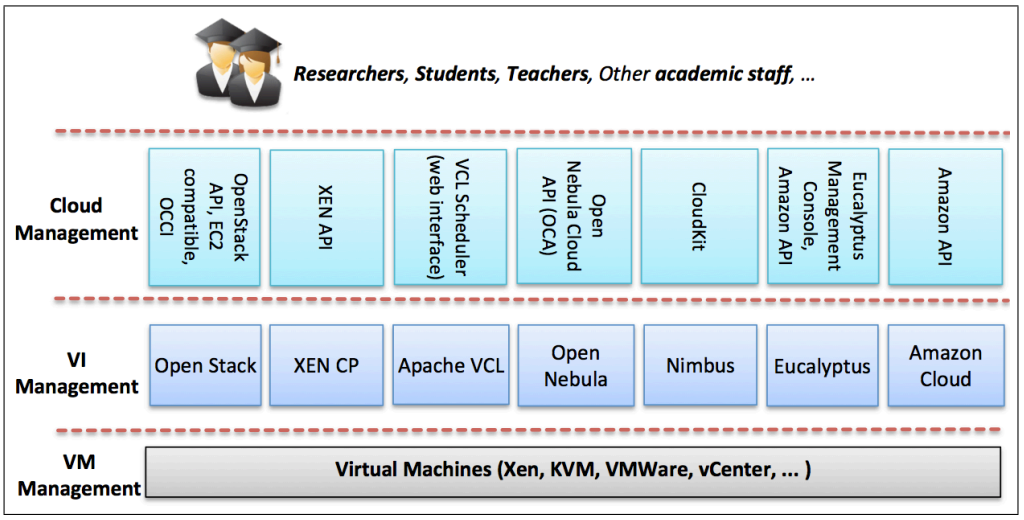

FIGURE 2.5: Structure of popular academic clouds.

- Provisioning of IT services to students and academic staff;

Although several existing academic Clouds fit in various informal Cloud definitions, they differ in several aspects such as how they are configured, managed, and programmed [70]. The nature of academic applications (both in research and teaching domains) requires full control over the system. Therefore, private or community Cloud computing platforms are considered a first choice by the academia. This requirement also favors open-source solutions as they have additional benefits such as low initial investment and support for customizable scientific libraries. Mostly, Cloud systems tend to provide several compatible APIs to make it easy for Cloud engineers to integrate different functionalities of various Clouds to work together as shown in Figure 2.5.

The discussion about few of the most popular Academic Cloud systems [71, 72] is presented as given.

OpenNebula [73] is an open-source toolkit utilized for the design and implementation of private Clouds. The flexible design and extensive customizability options of OpenNebula lends it one of the best candidates to be integrated with networking and storage solutions and to fit into existing data centers. From administrator's perspective, the significant customizability feature is the shared file system that

is used to store all of the OpenNebula's files; whereas from user's perspective, resources such as memory, processor, network, and storage can be requested in any configuration.

Nimbus [74] advertises itself as a science cloud solution, which is equipped with the `cloudkit` to manage the configurations. Cloudkit consists of a manager service hosting, an image repository and Globus [75] credentials for all users authentication. Nimbus provides several customization options to Cloud administrators. Moreover, Nimbus has the capability to turn clusters into an *Infrastructure as a Service* (IaaS) for Cloud Computing focusing mainly on scientific applications.

OpenStack [76] is among the most complete open-source solution for creating and managing small Cloud infrastructures, however it lacks advance monitoring and reconfiguration mechanisms, useful to implement dynamic service calling features. It consists of mainly four customizable components for both computation and storage resources to enable dynamic allocations of *Virtual Machines* (VMs) i.e., *API Service*, *compute service*, *network services*, and *schedular services*. OpenStack is also compatible with Amazon EC2 and Eucalyptus to interact with shared web services.

Apache Virtual Computing LAB (VCL) [77] is an open-source solution for the remote access over the Internet to dynamically provision and reserve computational resources for diverse applications, acting as *Software as a Service* (SaaS) solution. VCL manager is the vital component that uses system profile and provenance information to optimize scheduling of the request and problem resolution. A typical VCL environment consists of an operating system (with an application or a combination of applications) that is distributed in nature including resource scheduler, resource manager, computational, storage, network resources, and an end-user interface.

Eucalyptus [78] is considered as one of most popular open-source Cloud computing framework. Eucalyptus is flexible, modular, extensible, and has the potential to be integrated with the existing commercial solutions (i.e., Amazon Cloud etc). The highly decentralized design of Eucalyptus (with multiple clusters, distributed

storage, and locally stored virtual disks) lends itself to a large number of machines. The Eucalyptus is considered an implementation platform for various research studies that includes *intrusion detection* and *video streaming*.

In particular, academic Cloud systems provide several compatible APIs to facilitate Cloud engineers in integrating different functionalities of various Clouds to work together as shown in Figure 2.5. Such Clouds could be based on virtual machine (VM) management, Virtual Instance (VI) management, and Cloud management layers.

Keeping in view the architecture of existing academic Clouds, we have envisioned the general academic Cloud architecture for network simulations. The architecture comprises of three components that are *User Interface* (UI), *Work Breakdown Manager* (WBM), and *Simulation Execution Manager* (SEM) as shown in Figure 2.6. Users are provided with web-based interfaces to start Cloud instances through available Cloud APIs. The UI component performs the functionality available in cloud management layer. Concerning VI management, WBM is responsible for the provisioning of required virtual instances and allocation/de-allocation of demanded resources in Cloud. The core module SEM performs management of simulation execution. Simulation executions can be serial or parallel. In case of monotonic simulations, the SEM signals the simulator to start simulation execution in serial. However, if the simulation is large-scale, the SEM decomposes the simulation model into a number of components and allocates them on the available execution units. To achieve parallelism, *Message Passing Interface* (MPI) [21] can be used for interaction between SEs on different cores or on different VMs. Taking into account these design considerations of general academic Cloud architecture, this work implemented SIM-Cumulus, which is presented in Chapter 4. The simulation on Cloud computing can be executed in parallel on one (i.e., multi-core) or more Cloud instances. The simulation model thus need to be partitioned into smaller parts where each part need to be executed independently on a core or physical VM instance. The simulation partitioning is further discussed in next section.

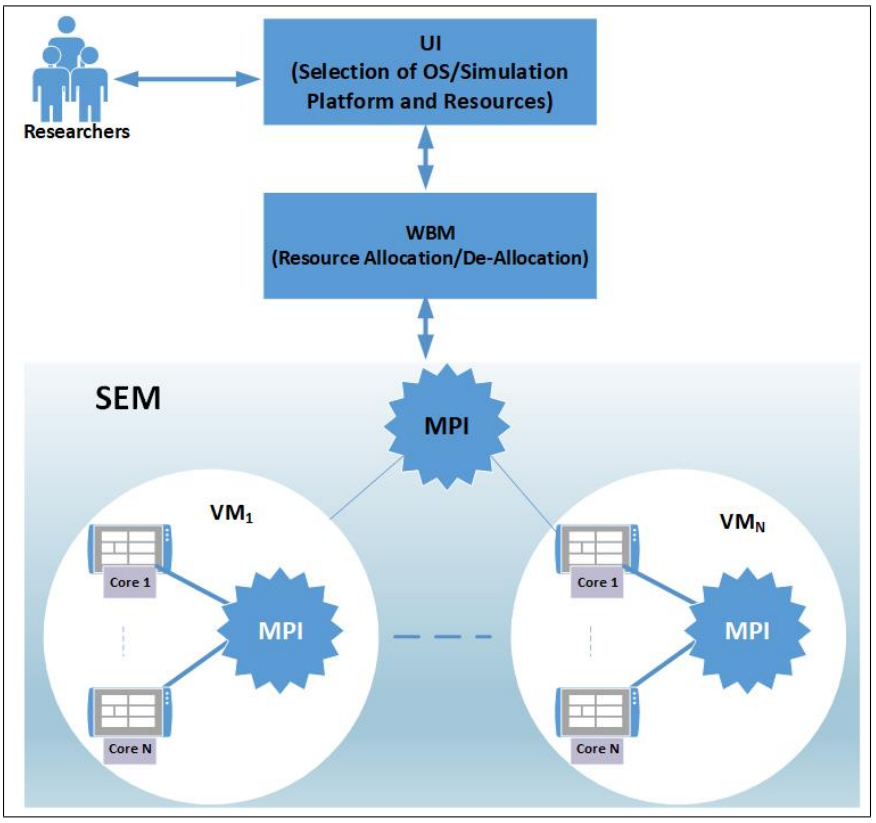

FIGURE 2.6: Generic Cloud architecture for network simulations.

2.4 Simulation model partitioning and SE migration

Executing large-scale wireless network simulation in parallel involve partitioning of the simulation model into a number of smaller components also called LPs [23]. The components encompasses part of the simulation that need to be executed independently on an individual PEU. Concerning the partitioning of simulation models in a distributed environment, a number of approaches have been proposed to maintain a global shared-state in the distributed simulation. Some partition schemes are based on the static approach [25], while other schemes such as simulation domains [79], data distribution management [80], [81], dynamically adaptive partitioning [82], and hierarchical federations [83] use dynamic approach. The author presented a static partitioning approach for parallel conservative simulations [84]. In [85], the author proposed the static as well as dynamic load balancing

strategies. The proposed approach [85] relies on the use of conservative synchronization algorithm and thus is applicable only on shared memory architecture. The author introduced a novel approach [86] for the partitioning of disjoint parts of the large-scale network topology and assigning them to different LPs for performance efficiency. Three set of partitioning strategies (i.e., random, k-cluster, and METIS) are implemented in order to quantify the performance of the proposed partitioning scheme. The author proposed a novel partitioning approach that use the concept of convex hulls [87] for partitioning of crowded simulations. The authors in [88], [89], [90] presented the work related to the provisioning of the Internet of Things (IoT) simulations. The proposed methodology in [88] integrates multiple simulation models (i.e., OMNeT++, MATLAB, and ARTIS/GAIA) for the implementation of IoT-based complex scenarios. Logan et.al floated a novel agent-based approach for PADS simulations [91]. In their approach, each of the agent is implemented as distinct LP. The dynamic partitioning method is used to implement the parallel simulation using shared-state approach. The proposed technique support conservative as well as optimistic synchronization in an adaptive way to meet the requirements of computation and communication demands of the simulated model. The associated cost related to the migration is not optimal and may vary due to the involvement of high mobility nodes. The author in [92], proposed a load balancing and migration scheme for the HLA-based large-scale distributed simulation. The proposed scheme balance the communication as well as computation among different parts of the simulations. The idea is to employ local as well as cluster monitoring mechanism to observe the imbalance of communication pattern and load variation and repartition the simulation model dynamically. A number of migration decision making approaches are employed for prediction management of load balancing across large-scale PADS systems. The obtained results shows better performance in terms of local communication and load balancing. Most of the PADS solutions have been suffered due to the high cost of remote communication and computation required for synchronization management. A number of techniques have been proposed to accomplish

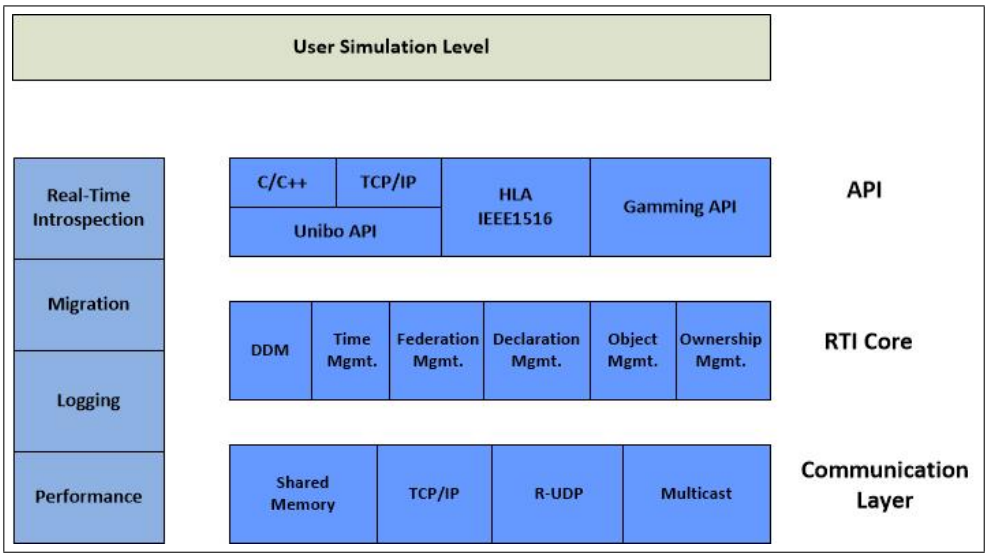

FIGURE 2.7: ARTIS high level architecture [95].

the speedup in simulations using migration and load balancing for PADS simulation [23], [27], [26]. In [93], [94], the authors have proposed dynamic partitioning algorithms for optimistic distributed simulation that is based on the migration of parts of the simulated model. The proposed approaches reduces the computation and communication cost using migration and load balancing mechanisms. Anglo et al. [94] have introduced an adaptive and dynamic approach to obtain simulation speedup through the use of two proposed heuristics for efficient migration and load balancing of SEs. In [81], the authors have proposed ARTIS Middleware, which supports PADS for large-scale wireless networks over heterogeneous execution architecture. The high level architecture of ARTIS is shown in Figure 2.7. The ARTIS is comprised of four different layers. At the top of the model, the user simulation layer is shown that allow the user to run the simulation. A number of APIs are provided by the ARTIS to work with different services provided by ARTIS. The RTI core layer is responsible for the management of distributed simulation across different LPs. The bottom layer represent the communication layer, responsible for message defemination among different components of the simulation model. The ARTIS is also integrated with GAIA framework that manage the migration of SEs among different LPs. GAIA is also responsible for synchronization and state updating of different SEs upon their migration. The ARTIS uses migration mechanism to minimize the remote communication among SEs located on

different LPs. Moreover, ARTIS is in combination with GAIA [96] is implemented for load balancing of SEs across different LPs. Anglo et al. [97] have designed a new simulation tool namely *PaScas* that enables the simulation of scale-free networks. The ARTIS middleware is used as the underlying module for remote communication provisioning and management among SEs (i.e., located on different LPs). In addition, the authors have presented a number of algorithms (named as GOSSIP) as test cases to evaluate the performance of the scale-free networks. The proposed solution leads to a good level of LCR; however, the number of migrations leads to an extra overhead. Thus, producing a negative impact on the overall performance gain. A slightly different technique has been proposed in [98] that uses a fault-tolerant parallel simulation Middleware to support PADS. The SEs are replicated and distributed on different LPs, which consequently provides a mechanism to handle crash-failures of execution units (LPs). In [23], PADS approach is presented that uses the concept of dynamic partitioning (i.e., migration) of the simulation model to handle large-scale wireless network simulation. A set of heuristics are being employed that are evaluated (i.e., locally on each LP) for the SEs migration during the simulation execution. In Table 2.2, the comparison about various partition approaches (i.e., static and dynamic) is presented. The techniques are compared in terms of migration support, the mobility speed level, and the number of migrations.

TABLE 2.2: Partitioning approach comparison.

Paper	**Partitioning approach**	**Migration**	**Mobility**	**number of Migrations**
[25]	Static	No	Yes	N/A
[82]	Dynamic	Yes	Yes	High
[83]	Dynamic	Yes	Yes	Moderate
[84]	Dynamic	Yes	Yes	High
[85]	Static/Dynamic	Yes	Yes	Moderate
[98]	Dynamic	Yes	Yes	Moderate
[97]	Dynamic	Yes	Yes	High

The static partitioning techniques do not support migration. The approach in [82] and [84] support high mobility and thus results in higher number of migrations to localize the communication among SEs. The technique in [85], support static as

well as dynamic partitioning and require less number of migrations as compared to [82] and [84]. The summary of the chapter is presented below.

2.5 Chapter Summary

To meet the requirements of the large-scale network simulations, the researchers proposed a number of approaches including Grid computing [32], Agent-based solution [33], [34], [35], [36], [37], HPC [29], and Cloud computing [38–41].

The grid computing environment is considered as most viable solution for specialized type of large-scale computing tasks. Considering gird computing for the simulation of large-scale network scenarios often lead to synchronization overhead as the components of the grid are usually loosely coupled.

The agent-based approaches are suitable for large-scale jobs where the synchronization among the agents is less frequent. However, agent-based approaches are not appropriate for large-scale wireless network simulations where the nodes in the simulation are mobile in nature and thus frequent synchronization among the agents is required. Moreover, the agent-based approaches require HPC to support the network simulations. However, specialized type of training is required to configure the simulation environment for large-scale network simulation on HPC [29].

Recently, Cloud computing provides solution encompassing large-scale problems including the domains of storage, computationally intensive tasks, real-time applications, and simulation jobs [38–41] etc. Due to the complex nature of large-scale network simulations and limitations of network simulators; to-date only few Cloud solutions (e.g., SEMsim cloud service [29], CloudSME [65] etc.) are available that are specifically designed for large-scale simulation jobs.

The existing Cloud frameworks have considered specifically designed network simulators (encompassing a specific network scenario e.g., urban scenario) for large-scale network simulations. Similarly, most of the publicly available generic Clouds (i.e., Amazon Web Services (AWS), Google Cloud etc.) provide various type of

services (i.e., infrastructure, storage, software service, and platform). Although, these Clouds are quite powerful in terms of available resources; however, usefulness of the existing Cloud platforms is questionable because of difficult integration with the existing state-of-the-art simulation tools (i.e., NS-2, NS-3, OMNeT++ etc.).

Large Scale network simulation can be dealt by using the approach of Parallel and Distributed Simulation (PADS).This approach divides the simulation into a number of components labeled as Logical Processes (LPs) and each LP contribute as an integral part of simulating model which has (LP) having equal number of (SEs). In the wireless system the SEs have high mobility due to which remote communication (having high cost) get involved hence resulted into increased simulation execution time. Various migration and load-balancing Techniques [23], [27], [26], [23] have been suggested for minimizing the costly distant remote communication in context of large-scale wireless network simulations. Although, the prevailing approaches resulted into an increased level of Local to Remote Communication Ratio (LCR); yet, huge number of migrations (in wireless networks) can contribute towards additional overhead and reflects a negative impact on the overall acquired performance. Due to the high volume migration existing approaches become unfitting for large-scale wireless having high mobility. To cop this situation an adaptive migration approach is required that could administer the appropriate level of LCR by keeping the possible minimal number of migrations. The subsequent chapter delineates the proposed methodology, grounded by the discussion provided in this chapter.

Chapter 3

Research Methodology

This chapter provides an overview of the operational and systemic research framework to achieve the objectives of this research work that are:

1) propose an academic Cloud SIM-Cumulus to support the provisioning of large-scale NSaaS. The SIM-Cumulus supports small-to-medium as well as large-scale wireless network simulations.

2) propose adaptive migration algorithm MAHA that provides dynamic migration of SEs in the context of PADS simulations. To support migration of SEs, ARTIS/-GAIA middleware is integrated into SIM-Cumulus (named as A-SIM-Cumulus).

The research operational framework provides the detailed research methodology used to design and implement SIM-Cumulus and MAHA algorithm, respectively. The details of different phases involved in research operational framework are presented in next section.

3.1 Research Operational Framework

Figure 3.1 corresponds to the operational framework of this research that is comprised of three phases i.e., 1) Problem Investigation, 2) Design and Procedure, and 3) Simulation Setup and Performance Evaluation.

The problem investigation provide an insight into the details regarding research gap followed by the design and implementation of the proposed academic Cloud SIM-Cumulus and MAHA algorithm. Furthermore, the details of simulation experiments and performance evaluation are explored in the last phase of the operational framework. The discussion details about each phase is presented in the following sections.

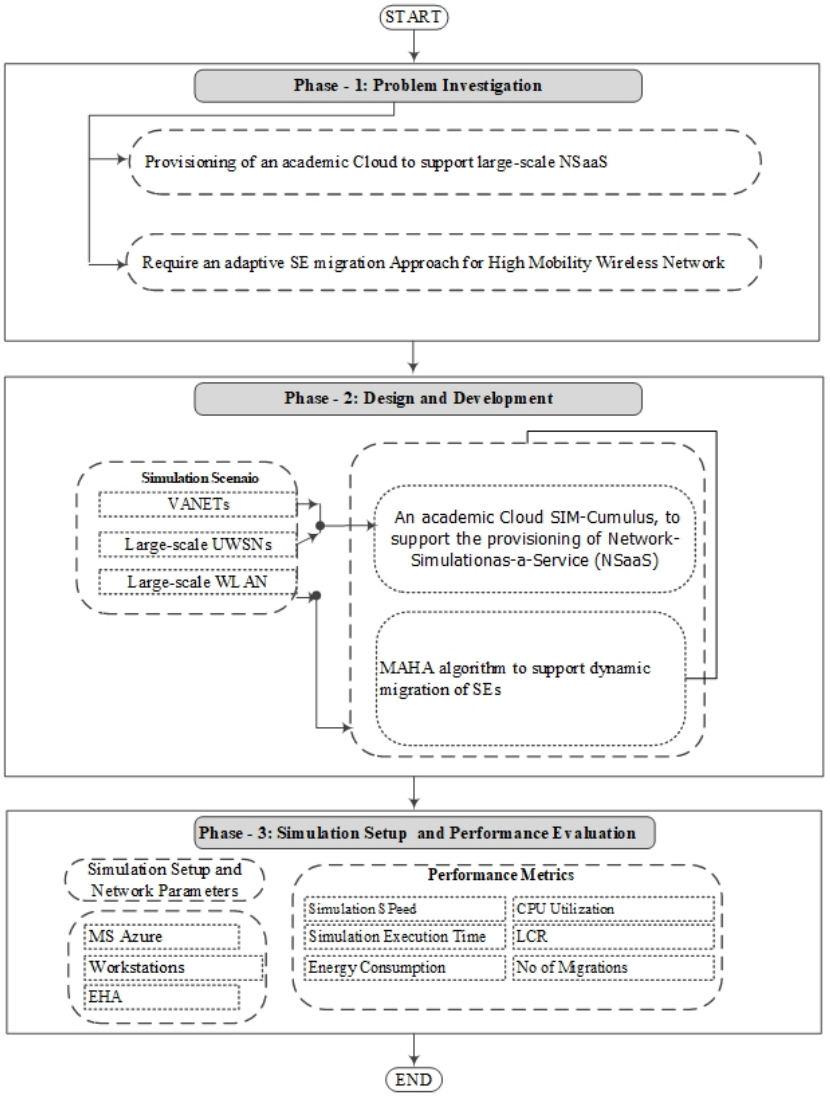

FIGURE 3.1: Flowchart of the Research Operational Framework.

3.2 Problem Investigation

The problem formulation concerning proposed work is based on the literature review of the existing approaches used to support large-scale NSaaS. The existing approaches are studied in detail aiming to identify the current challenges in the design and development of an academic Cloud that provide network simulation services to the researchers and handling large-scale network simulations. Furthermore, the existing literature related to simulation model partitioning techniques (i.e., static as well as dynamic partitioning) for PADS in Cloud are discussed with an aim to propose adaptive migration algorithm.

3.2.1 Provisioning of an Academic Cloud to Support Large-scale NSaaS

Numerous approaches i.e., Grid computing [32], Agent-based solution [33–37], HPC [29], and Cloud computing [38–41] are being employed by the researchers to attain the demands of large-scale network simulations. Grid computing is devised for large-scale distributed data processing/storage that functions with allocated subtasks across different clusters [32], [42]. The grid computing environment is best suited choice for specialized type of large-scale computing tasks. It is often inefficient for network simulation jobs as the constituents of the grid are loosely coupled and thus the synchronization is quite complex among the respective components. Agent-based simulation approaches are well suited for scenarios where each agent has been assigned to do individual segment of task independently. The agent-based approaches are not suitable for large-scale network simulation tasks due to required frequent synchronization among different agents. In addition, agent-based approaches require HPC to support the network simulations. Moreover, the HPC environment is restricted and requires pre-training to be configured to meet the simulation requirements [29]. Recently, Cloud computing offers solution to large-scale computing problems related to the domains of storage, computational intensive tasks, real-time applications, and simulation jobs etc. Keeping in

mind the complex nature of large-scale network simulations and limitations of network simulators; only few Cloud solutions (e.g., SEMsim cloud service, CloudSME etc.) have been proposed that are particularly designed for large-scale simulations. To support large-scale network simulations, the existing Cloud approaches suffer from following issues that are: 1) for large-scale network simulations, the existing Cloud frameworks have considered specifically designed network simulators (encompassing a specific network scenario e.g., urban scenario). These network simulators are only compatible with the proposed Cloud architecture. 2) Rest of the available public Clouds (i.e., Amazon Web Services (AWS), Google Cloud etc.) offer multifaced services including infrastructure, storage, software service, and platform. Even though, these Clouds are quite powerful in terms of available resources; yet, usefulness of the these Cloud platforms is questionable because of difficult integration with the contemporary simulation tools (i.e., OPNET, NS-2, NS-3, OMNeT++ etc.).

3.2.2 Require an adaptive SE migration Approach for High Mobility Wireless Networks

Executing large-scale wireless network simulations in parallel involve partitioning of the simulation model into a number of smaller components also called LPs, where each LP encompasses part of the simulation that need to be executed independently on an individual PEU. The SEs in the wireless networks are mobile in nature and thus involve high remote communication cost due to high mobility of SEs that ultimately increase simulation execution time. The authors in [23], [26], [27] contributed a number of migration and load-balancing techniques with an aim to reduce the high-end remote communication for large-scale wireless network simulations. Although, existing approaches lead to a good level of LCR; however, huge number of migrations (in wireless networks) may lead to additional overhead and producing a negative impact on the overall performance gain. The higher number of migrations make these approaches unsuitable for large-scale wireless network simulations involving high mobility.

3.3 Research Design and Development

In this section, the aforementioned research gaps are addressed by the provisioning of two contributions that are 1) designing and implementing an academic Cloud named SIM-Cumulus and 2) propose MAHA approach to provide dynamic migration of SEs. These two contributions along with the respective methodology details are discussed in Table 3.1.

TABLE 3.1: Design and Development of the Research

Table

Research Questions	**Research Objectives**	**Methodology**
How to reduce the execution time required for large-scale network simulations (that are time consuming and resource intensive).	To design and develop an academic Cloud framework with an aim of providing NSaaS and handling large-scale network simulations.	The proposed academic Cloud executes large-scale (VANET and UWSNs) simulations in parallel on multiple LPs to speedup the simulation that ultimately decrease the execution time.
How to reduce the higher number of migrations in highly mobile and scalable wireless network simulation?	To design and develop a migration-based adaptive heuristic algorithm (MAHA) that reduces the higher number SEs migrations across PADS for both multicore and distributed architectures.	The proposed MAHA approach minimizes the number of migrations by exploiting the transitive dependency of SEs in order to localize the simulation and decrease the execution time.

3.4 Simulation Setup and Performance Evaluation

Simulation setup and performance evaluation is the last phase of the research framework. In this phase, a number of simulations executed in sequential and in parallel to evaluate the performance of the SIM-Cumulus and compare the results against stand-alone workstations and generic Cloud instance(s). In addition, simulation results (in terms of LCR, number of migrations, simulation execution time, and simulation speedup) are obtained to evaluate the performance of MAHA and compared to the EHA. This section presents simulation setup, performance metrics, and performance evaluation steps in the following sub-sections.

3.4.1 Simulation Setup

A network simulator provides a virtual environment to design and analyze the proposed models and protocols without getting into details of real-life environment. In this research, three different sets of simulations are performed to evaluate the performance of the proposed SIM-Cumulus and MAHA approach. The first set of simulations (using VANET simulation over OPNET modeler) is performed (sequentially) on three stand-alone machines as well as on Amazon EC2 instances. The second set of experiments consists of large-scale UWSN simulations on OMNeT++ and WLANs simulations on ARTIS/GAIA simulator. Three workstations and two Cloud instances (i.e., SIM-Cumulus and MS Azure) are utilized for the simulation experiments. The large-scale UWSNs simulation are executed in monolithic as well as parallel fashion on all the machines. The third set of simulations are executed for the proposed MAHA approach on ARTIS/GAIA PADS framework. In this experiment, four simulations are executed on multi-core and distributed setup for MAHA and EHA comparison.

3.4.2 Performance Metrics

The efficiency and performance of SIM-Cumulus for large-scale wireless networks are tested and evaluated in terms of simulation speed (in number of events per second), execution time, energy consumption, CPU utilization, and LCR. Simulation results in terms of LCR, number of migrations, simulation execution time, and simulation speedup are obtained to evaluate the performance of MAHA and compared against EHA approach. These performance metrics are discussed in the following sub-sections.

Simulation speed The simulation speed shows the number of events executed per second.

Execution time The execution time indicates the wall clock time required to complete the simulation execution on each machine and Cloud instances.

Energy consumption Energy consumption shows the total energy consumed by the machine to generate simulation events during the simulation execution. The energy consumption is represented by two sub-metrics that are electricity consumed and CO_2 emission. The electricity consumed represents the total electricity consumed during the simulation execution on all machines and Cloud instance whereas CO_2 emission indicates the total CO_2 emitted during each simulation execution.

CPU utilization The CPU utilization represents the CPU usage trend on each LP during all the simulation runs.

LCR The LCR corresponds to the ratio of the total number of local communications by all SEs in a given LP, with respect to all interactions originating from this LP.

Number of migrations The number of migrations represents the total number of SEs migrated during each simulation run on a specific machine. The migrations corresponds to the movement of an SE from one LP to another remote LP (after an SE has done sufficient remote communication and thus eligible for migration).

3.4.3 Performance evaluation

The experiments (using VANET simulation) are performed (sequentially) on three stand-alone machines as well as on Amazon EC2 instances. The results are plotted for three parameters that are simulation speed in terms of simulation events per second, execution time in number of hours to complete the simulation execution, and energy consumption (i.e., electricity consumed and CO_2 emission). To evaluate the performance of SIM-Cumulus, large-scale UWSN simulations (on OMNeT++ and WLANs simulations on ARTIS/GAIA simulator) are executed on three workstations and two cloud instances (i.e., SIM-Cumulus and MS Azure). The large-scale UWSNs simulations are executed in monolithic as well as parallel fashion on all the machines and Cloud instances. Moreover, four set of simulations are executed on multi-core and distributed setup for the proposed MAHA approach to evaluate its performance against EHA approach.

Chapter 4

Proposed Work

The research work in this thesis contributed an academic Cloud namely SIM-Cumulus, to support the provisioning of NSaaS. SIM-Cumulus handle small-to-medium as well as large-scale wireless network simulations. Moreover, MAHA algorithm is proposed that supports dynamic partitioning of large-scale PADS simulation. In addition, ARTIS/GAIA middleware is integrated into SIM-Cumulus (named as A-SIM-Cumulus) that supports PADS simulation. This chapter provides discussion about SIM-Cumulus and A-SIM-Cumulus in section 4.1 and section 4.2, respectively. The details about Dynamic partitioning and MAHA approach are presented in section 4.3 and section 4.3.3, respectively.

4.1 SIM-Cumulus

This section delineates the high-level architecture and workflow of SIM-Cumulus. In addition, details regarding mathematical modeling, and cost analysis of sequential and parallel simulation execution on SIM-Cumulus are also part of this section.

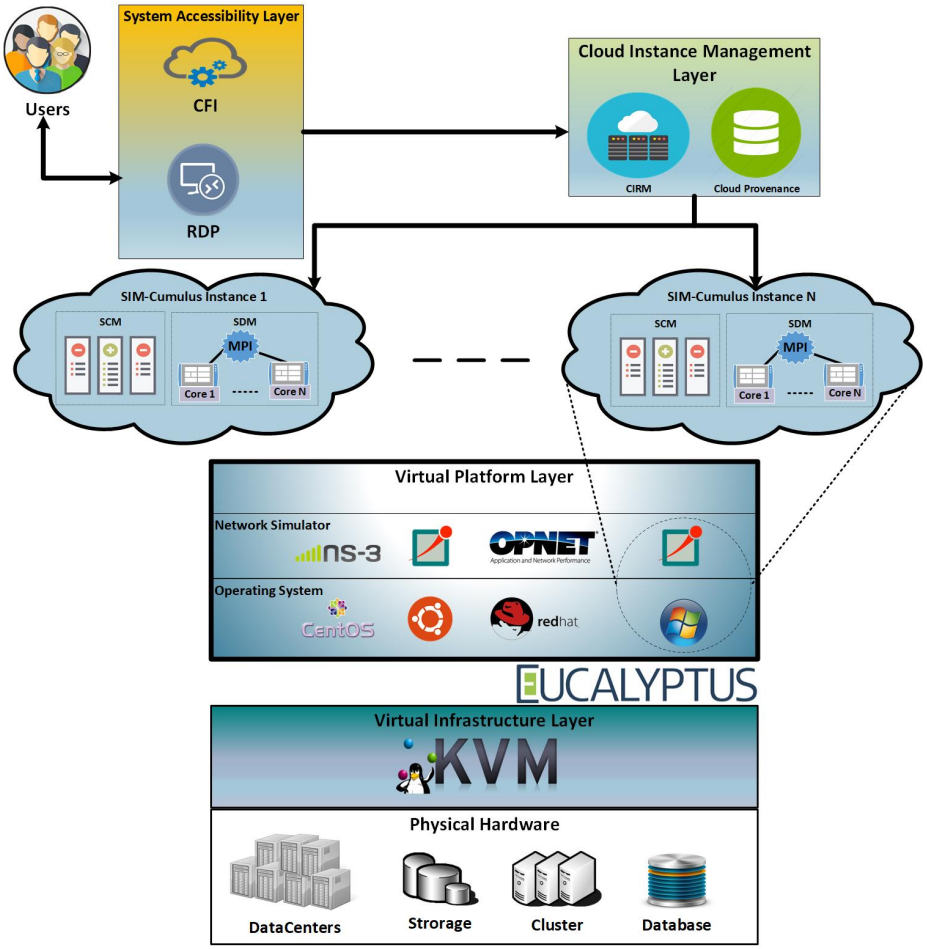

FIGURE 4.1: SIM-Cumulus proposed architecture.

4.1.1 SIM-Cumulus Architecture

Considering the layering approach of our envisioned Cloud for network simulations, the implemented SIM-Cumulus comprises of four layers i.e., System Accessibility Layer, Cloud Instance Management Layer, Virtual Platform layer, and Physical Infrastructure layer as shown in Figure 4.1. The details of all necessary SIM-Cumulus components (at all layers) required to perform network simulations are presented below.

System Accessibility Layer (SAL) The SAL enables the users to interact with *Cloud Instance Management Layer* (CIML) by using *Cloud Front-end Interface* (CFI). At CFI, the *Representational State Transfer* (REST) API is used to provide end-user registration and acquire Cloud instance usage information. RESTful web services allow end-users to configure and launch required instances

of SIM-Cumulus through selection of operating system, network simulator, mobility model, traffic generators etc. The *Remote Desktop Protocol* (RDP) serves as a launcher of SIM-Cumulus instances.

Cloud Instance Management Layer (CIML)

The CIML layer hides the configuration complexities associated with network simulation tools, which reduces the time consumed during installation/configuration/setup phase. CIML layer consists of two major components that are *Cloud Instance Resource Management* (CIRM) and *Cloud Provenance*. CIRM is responsible for configuration, control, and management of on-demand SIM-Cumulus virtual instances. This component exposes simulation services to the SAL. The envisioned components of proposed CIRM include *Service Level Agreement Management* (SLA), *Account Management Module* (AMM), *Snapshot Organization Module* (SOM), *Configuration Module* (CM), and *Load Balancer* (LB). However, the focus of this work is related to the realization of two modules (i.e., AMM and CM) of this layer. The AMM is responsible for management of user-related information such as authentication, instance usage information, accessibility rights etc. CM is responsible for creating VM instances according to the user's configuration. The description of other related components of the proposed SIM-Cumulus can be found in [99]. The Cloud Provenance component keeps track of user and machine-level information. This includes the demands/requests of a user for prescribed network simulation as well as for the resource patterns of Cloud instances. The provenance related to the individual Cloud instances is used to dynamically model the resources according to the behavior of network simulation. The SIM-Cumulus uses the services of Eucalyptus platform [100] to provide instance-level Cloud provenance. In addition, a third-party library *JavaSysmon* [101] is utilized to obtain system-level Cloud provenance.

Virtual Platform Layer (VPL) VPL is fundamentally responsible for configuration of virtual operating system instances configured with various network simulators (i.e., NS-2, NS-3, OPNET, OMNeT++ etc.) based on the DES kernel and

parallel simulation support. Moreover, it provides realistic mobility map trajectory and traffic generator tools. The VPL enables users to access simulation-based Cloud instances using Eucalyptus services. VPL layer comprises of two modules (i.e., Simulation Configuration Module (SCM) and the Simulation Distribution Module (SDM)) to manage small and/or large-scale network simulations. Initial network simulation configuration parameters (simulation type, number of nodes, simulation area, simulation time, mobility model, sequential/parallel mode etc.) are provided by the users through the SCM module. Once the initial parameters are provided, the SDM module decides on either a sequential or parallel run. For small-scale networks, the simulation is executed in sequential fashion. To execute large-scale simulations, the SDM module of the SIM-Cumulus is responsible to partition the simulation into equal number of components also called LPs. SDM also assign each LP to a separate execution unit (core). After LP assignment, the SDM module signals the simulation tool (i.e., OMNeT++ in our case) to execute simulation in parallel.

In general, the SDM considers Cloud provenance information (accumulated in the SIM-Cumulus repository) for decision making in terms of sequential or parallel simulation execution. The Cloud provenance information is used to keep track of: 1) Cloud instance resource usage (i.e., RAM, CPU, Virtual Memory) and 2) simulation details i.e., simulation tool, scenario type, number of nodes, simulation execution time, execution strategy (sequential or parallel) etc.

To elaborate the SDM decision making process, a number of simulations are performed using different problem sizes (number of nodes). Table 4.1 demonstrates the results of simulation execution with sequential and parallel modes. The speedup value smaller than 1 (as compared to sequential execution) shows degradation in performance of parallel execution (based on 2 LPs).

Speedup in execution time indicates that results with sequential execution are better as compared to the parallel execution (2 LPs) for the simulation runs using 400 or less nodes. This degraded performance of the parallel simulation is due to the overhead involved in the parallelization (for smaller problem size). This

TABLE 4.1: Obtained Execution Speedup

Nodes	Sequential (Sec)	Parallel (Sec) with 2 LPs	Speedup
100	600	1250	0.48
150	720	1380	0.52
200	1320	2074	0.64
300	3600	4050	0.89
400	11520	11952	0.96
500	17280	15300	1.13
600	24876	21890	1.14
700	32580	28260	1.15
800	46080	37152	1.24
900	54360	42476	1.28
1000	71820	41840	1.72

overhead leads to an increase in execution time. However, as the simulation size increases (greater than 500 nodes), an improved execution speedup of parallel simulation is observed up to 1.28 for 2 LPs. With an even larger simulation size (1000 nodes), speedup of 1.72 for 2 LPs is observed that is very close to an ideal theoretical speedup of 2.

Keeping in view these results, the SDM module uses provenance information (i.e., number of nodes, simulation execution time, and execution strategy (sequential or parallel)) for simulation execution decision. Results provided in Table 4.1 are specifically related to a realistic simulation scenario of Underwater Wireless Sensor Networks (UWSNs) using OMNeT++ (detailed configuration is provided in simulation setup section). However, the decision of SDM is not confined to this simulation scenario and OMNeT++ simulator only, but can be applied to any other realistic simulation/simulator. In section V, we have also provided the results of ARTIS/GAIA-based wireless network simulations to highlight the usability of SIM-Cumulus with other network simulators.

Virtual Infrastructure Layer (VIL) VIL resides at the lower layer of SIM-Cumulus. It has a role to expose the Cloud resources to the upper layers. We have considered *Eucalyptus* Open Source Cloud platform [100] to implement the SIM-Cumulus Cloud architecture. The SIM-Cumulus exploits the benefits of the *Eucalyptus* to provide NSaaS to the researchers. The *Eucalyptus* provides feasibility

for private and hybrid Cloud implementation [102]. The main reason for adopting Eucalyptus is its inherent flexibility and interoperability with contemporary commercial solutions (i.e., Amazon EC2, IBM SmartCloud etc.). In addition, the highly decentralized design of Eucalyptus (with multiple clusters, distributed storage, and locally stored virtual disks) lends itself to a large number of machines. Eucalyptus uses KVM as a baseline hypervisor for virtualization in the Cloud environment. KVM can achieve hardware acceleration by making use of emulated I/O supported by QEMU [103]. KVM minimizes the virtualization overhead to very low levels by combining hardware acceleration and para-virtual I/O. Moreover, KVM supports live migration of running VMs (without disrupting the guest OS) during Cloud maintenance. However, dynamic migration is beyond the scope of this work.

Physical Infrastructure Layer (PIL) PIL represents the lowest layer in SIM-Cumulus architecture. It consists of physical computing resources such as: *multi/many-core machines, clusters, data-centers, networks, storage devices* etc., (see Figure 4.1). These resources are the actual hardware components used in the simulations and users are assigned to them in a transparent manner.

4.1.2 SIM-Cumulus Workflow

The *workflow* of SIM-Cumulus represents an automation of a network simulation task in the Cloud. In general, at each layer of the SIM-Cumulus architecture, the workflow can be distinguished into several phases as shown in Figure 4.2. The System Accessibility Phase deals with the important aspects of end-user verification and privacy. Registered users make use of CFI (i.e., REST and RDP) module to access configured Cloud instances. The Deployment Phase handles the preservation of the provenance information related to resource usage of the Cloud instances. The Provenance information assists network researchers by keeping track of resource usage for load balancing. The Configuration Phase is concerned with parameters of the simulation experiments. The Simulation Execution Phase

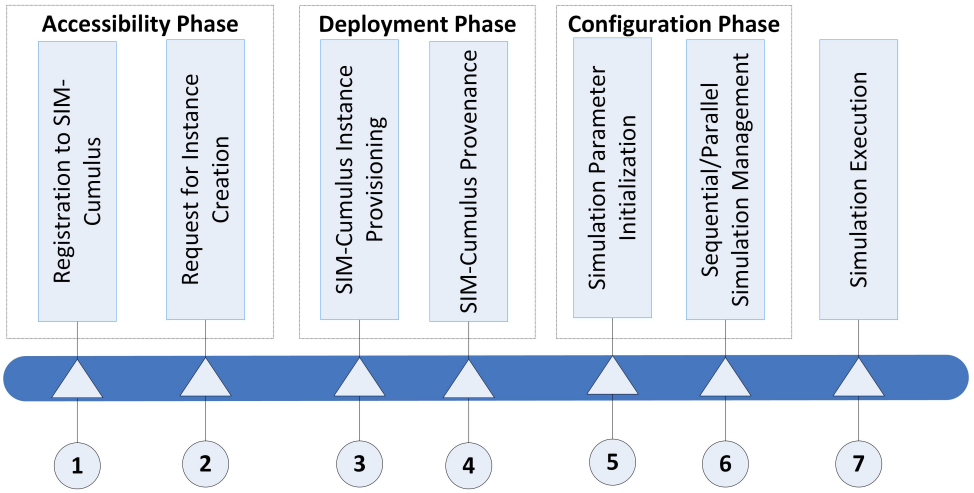

FIGURE 4.2: SIM-Cumulus workflow.

performs execution management based on the input parameters provided in configuration phase. In addition, dynamic preservation of resource (i.e., CPU, RAM) usage eventuates at this layer. Resource usage information is helpful for the provisioning of suitable Cloud instances for particular future simulation scenarios.

4.1.3 Parallel Simulation Execution

To obtain an insight into the performance of parallel simulation on SIM-Cumulus, a large-scale UWSN has been simulated using OMNeT++. OMNeT++ as Parallel Discrete Event Simulation (PDES) [104] offers flexible approach for parallel simulations. PDES has the capability to achieve high speed by distributing the simulation over several LPs. Each LP maintains their simulation clock independently and is responsible to keep track of the concurrent events execution [52]. Various approaches (i.e., Message Passing Interface (MPI), named pipes, file system based communication mechanism etc.) have been proposed to support synchronization among multiple LPs. In this work, we have utilized the MPI standard [21] for the communication of time stamped messages (i.e., event messages). Most of the simulators use the placeholder (PH) modules and proxy gates to achieve parallelism, such as OMNeT++ in our case. Placeholder modules hold sibling sub-modules that are instantiated on other LPs. Proxy gates receive the messages at the placeholder module and transparently forward it to the real module while residing on

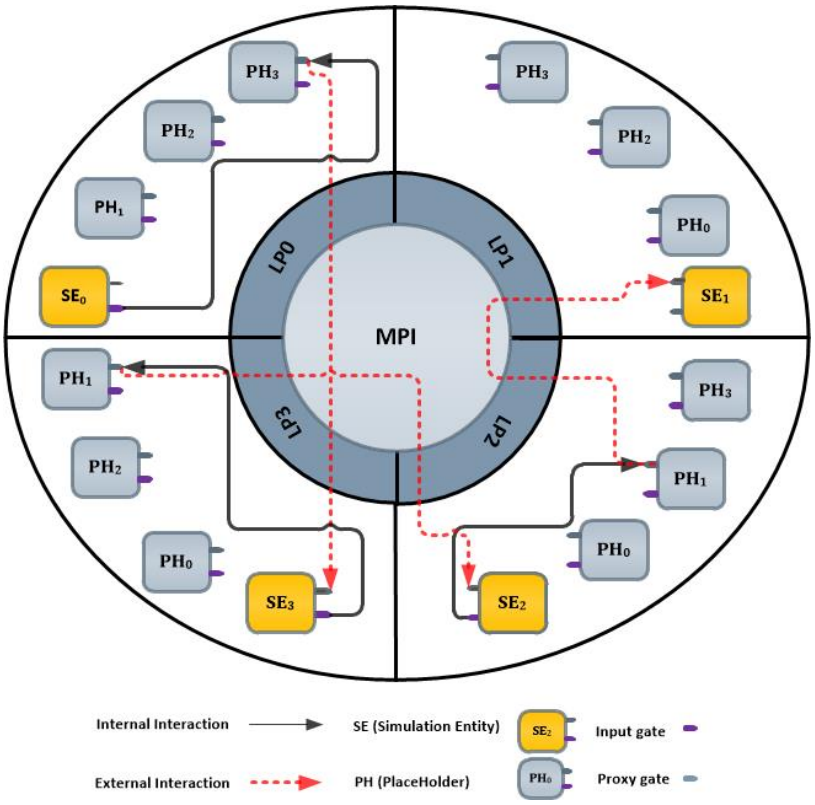

FIGURE 4.3: Remote communication among SEs.

another LP. Two types of message flows are illustrated in Figure 4.3. The first one indicates a flow between real node and placeholder node on the same LP and second shows a flow between placeholder nodes of two different LPs. For instance, SE_0 on LP_0 wants to send a message to SE_3 on LP_3. The SE_0 has no direct link with SE_3, therefore, it sends message to the placeholder node PH_3(on LP_0), which in turn is responsible for forwarding the message to SE_3 using proxy gate.

The detailed mathematical modeling of simulation execution on SIM-Cumulus is given below.

Let S_{LP} represents the set of LPs as shown in Equation 4.1:

$$S_{LP} = \{LP_1, LP_2, LP_3, ..., LP_m\} \tag{4.1}$$

and S_{SE} represents the set of SEs in the simulation model:

$$S_{SE} = \{SE_1, SE_2, SE_3, ..., SE_n\} \tag{4.2}$$

Equation 4.2 corresponds to the number of SEs in the simulation model. To distribute SEs across available LPs, it is required to find out the number of SEs (N) on each LP as shown in Equation 4.3.

$$N = n/m \tag{4.3}$$

where **n** represents the number of SEs in the simulation and **m** represents the number of LPs. In order to execute parallel simulation, SEs are required to be distributed across different LPs. The allocation of specific range of SEs on different LPs can be obtained by the given equation:

$$LP_i = \{(i-1)(N)+1, (i-1)(N)+2..., (i-1)(N)+N\} \tag{4.4}$$

In Eq.4.4, i represents index in S_{LP}. To execute simulation in parallel, SEs are required to be distributed across different LPs. The allocation of specific range of PHs on different LPs is obtained using Eq.4.5.

$$LP_{PH} = S_{SE} \setminus LP_i = \{SE : SE \in S_{SE} \; and \; SE \notin LP_i\} \tag{4.5}$$

The simulation execution is started after the placement of SEs and PHs on their respective LPs. During the simulation execution, SEs communicate with other SEs that reside either on the same LP on which the sending SE is located or on some other remote LP. Sender and receiver SEs (i.e., SE_S, SE_R) located within the same LP, send messages directly by using the Input gate (Ig) during their communication. However, if SE_S, SE_R are located on different LPs, the messages will be sent to the destinations in two steps. In the first step, the SE_S will send the message to the PH node using Proxy gate (P_g). In the second step, the PH will forward message to the corresponding SE_R through the use of P_g. The pseudo-code shown below presents how communication is performed among different SEs.

if $(SE_S \& SE_R) \in LP_i$ **then**

$Send_{msg}(msg, SE_S, SE_R, I_g)$

else

$Send_{msg}(msg, SE_S, PH, P_g)$

$Forward_{msg}(msg, PH, SE_R, P_g)$

end if

4.1.4 Simulation Cost Analysis

This section discusses the aspects related to the cost associated with the simulation execution in sequential/parallel mode. In the sequential mode, the simulation execution takes place on stand-alone machines and Cloud Instances in a monolithic fashion. For sequential execution, the *Overall Execution Cost* (OEC) can be defined as the time required to complete the simulation execution. OEC is composed of two different costs i.e., *State Updating Cost* (SUC) and *Local Interaction Cost* (LIC) as shown in Eq.4.6:

$$OEC = SUC + LIC \tag{4.6}$$

Simulation execution in DES environment illustrates that all the time is spent either on messages delivery among the SEs or updating the state variables after each simulation event. SUC corresponds to the cost involved in updating the state variables after each simulation event. LIC pertains to the cost involved in delivering the messages among entities (on same LP) during the course of simulation. On the other hand, if the simulation is partitioned into a number of LPs and need execution in parallel, then OEC can be obtained as shown in Eq.4.7:

$$OEC = SUC + GCC \tag{4.7}$$

GCC is generic communication cost that is comprised of *Interaction Cost* (IC), *Synchronization Cost* (SynC) and *Middleware Management* (MM) cost:

$$OEC = SUC + (IC + SynC + MM) \tag{4.8}$$

Thus the then modified OEC can be obtained as shown in Eq.4.8. The synchronization among different LPs should be ensured to obtain more accurate results. The IC pertains to the cost that is involved in delivering messages among different SEs. The cost of message delivery depends on the message size and the destination LP of the receiving SE. The location of the receiving SE is quintessential as it can lead to subtle difference in cost that whether the SE is located on the same LP or not. IC is composed of local and remote interaction cost as given in Eq.4.9. *Local Interaction Cost* (LIC) refers to the cost involved in delivering the message to the SE of the same LP and *Remote Interaction Cost* (RIC) refers to the delivery of message to an SE located on a remote LP.

$$IC = LIC + RIC \tag{4.9}$$

Thus overall execution cost can be calculated as given in Eq.4.10.

$$OEC = SUC + (LIC + RIC + SynC + MM) \tag{4.10}$$

The OEC depends on the ratio between local and remote communication. If L_{Comm} represents the size of local communication and R_{Comm} represents size of remote communication then LRR (i.e., Local to Remote Communication Ratio) can be obtained by the given formula (Eq.4.11).

$$LR_R = L_{Comm} \mid R_{Comm} \tag{4.11}$$

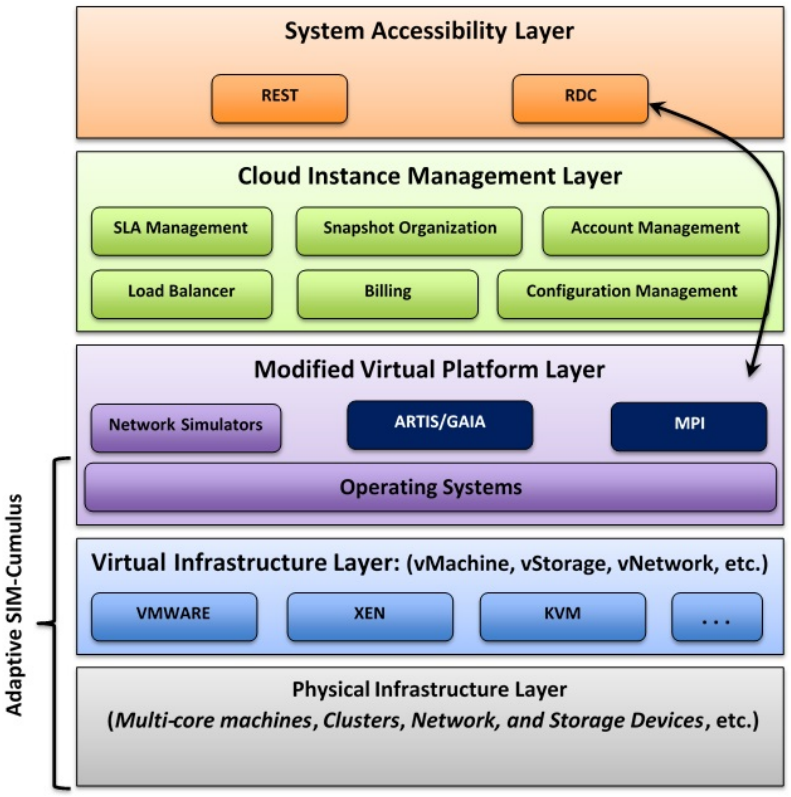

FIGURE 4.4: A-SIM-Cumulus Architecture.

4.2 Adaptive SIM-Cumulus Architecture

The SIM-Cumulus deals with the execution of network simulations on multi-core Cloud instances. To implement the execution of network simulations on multiple (distributed) SIM-Cumulus instances, ARTIS/GAIA middleware is integrated into SIM-Cumulus. The extended SIM-Cumulus is called as Adaptive SIM-Cumulus (A-SIM-Cumulus). The A-SIM-Cumulus comprises of four layers i.e., SAL, CIML, modified virtual platform layer (M-VPL), and PIL as shown in Figure 4.4. The details of necessary A-SIM-Cumulus components required to perform network simulations are presented below.

The description of SAL, CIML, and PIL of the A-SIM-Cumulus is already presented in section 4.1.1.

Modified Virtual Platform Layer (M-VPL): The M-VPL is responsible for the provisioning of Cloud instances configured with various network simulators (i.e., MATLAB, NS2, OMNeT++ etc.). The M-VPL uses the Eucalyptus services

and enable users to access Cloud instances. ARTIS and GAIA are integrated into VPL layer to support parallel simulations. The ARTIS is middleware that implements several modules to optimize parallel simulation (of microscopic, dynamic, and complex nature systems) over homogeneous and heterogeneous execution platforms [96]. The ARTIS offers several services such as simulation bootstrap and termination, LP co-ordination, track runtime statistics, synchronization management, and managing the interaction primitives for the communication among SEs on different LPs [23]. The GAIA framework allows the migration of SEs from one LP to another LP. The migration decision in GAIA is implemented on the basis of heuristics that basically involves the local and remote communication among different SEs. The local and remote communication of the SEs is useful in determining the migration decision of a particular SE. If the ratio of remote to local communication of an SE crosses a certain threshold, then that SE is considered as candidates for migration. The migration has a certain computational cost, which could be very crucial. In other words, the introduction of the complex heuristic algorithm for migration decision may attribute to a high overhead. Moreover, the heuristic may contribute to higher number of migrations while achieving the same LCR value. This work exploits the migration and communication services of ARTIS and GAIA middleware to perform adaptive migration of SEs dynamically. In this paper, the MAHA is proposed to use an intelligent heuristic for migration decision and minimize the number of migrations with an ultimate goal to achieve better LCR. The objective is to pay some extra computation (i.e., reduce number of migrations) with the aim to reduce the computation overhead required for large number of migrations. The MAHA algorithm is better in terms of achieving the same LCR with reduced number of migrations. The proposed algorithm is implemented and evaluated using our proposed Cloud framework known as A-SIM-Cumulus. The details pertaining to migration decision is provided in section. The VIL layer deals with the creation and destruction of cloud instances and PIL deals with the management of resource pool (i.e., hardware resources).

4.3 Dynamic Partitioning of Simulation Model

The PADS provides the opportunity to solve simulation problems that encompasses large number of nodes with micro-level details. The PADS approach requires the partitioning of simulation model into a number of equal partitions and then placing each component on a separate execution component called LP. Each LP is responsible for the management and handling of simulation events among the SEs that are located on it. A major hurdle in getting the required gain in performance is due to the LP synchronization and remote communication cost in the PADS simulation [23]. Moreover, the distributed simulation environment is dynamic and unpredictable in nature with respect to the network load. Partitioning the simulation model in an effective way can attribute to the performance gain. However, the static partitioning of wireless networks simulation does not help in a performance gain due to involvement of the high mobility which could lead to an increase in the cost of LP synchronization and remote communication. Thus, every form of static partitioning is inadequate and therefore, requires a dynamic and adaptive migration approach. In this work, a adaptive migration algorithm MAHA is proposed that automatically reconfigures the network simulation in an adaptive fashion to meet the run-time dynamics of the simulation. This is the starting point to design and implement a mechanism that will result in a reduction in communication cost and is discussed in section4.3.1. The proposed adaptive algorithm details are presented in section 4.3.3.

4.3.1 Remote Communication Cost Reduction

The simulation model comprises a number of SEs which interact with each other during the simulation execution. The intra-LP communication between SEs has low link latency. However, the intr-LP communication between SEs may face inconsistent latency depending on the type of link between the LPs. Figure 4.5 and Figure 4.6 shows the intra-LP and inter-LP communication of different SEs, respectively. The main idea is to observe the communication pattern of every SE

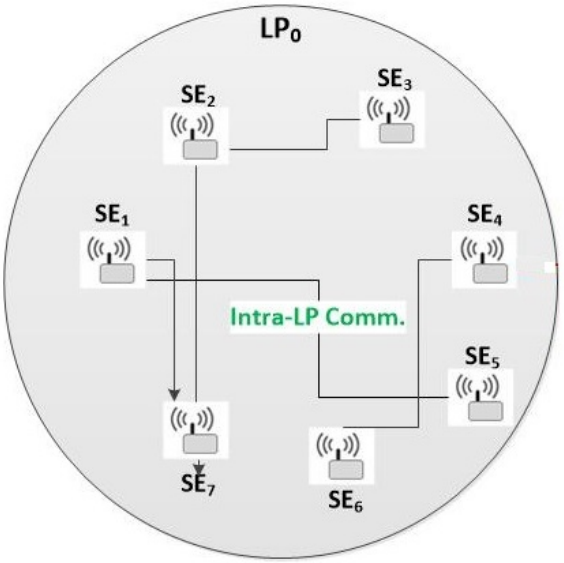

FIGURE 4.5: Intra-LP communication between SEs.

during simulation execution and decides whether the migration of SE to another LP is necessary. SEs that have high interaction with other remote SEs during certain time period are considered as appropriate candidates for migration. In this way, the high cost of inter-LP (remote) communication is reduced. The SE migration leads to a performance gain, however; it may results in performance degradation if the cost of migration is higher than remote communication cost.

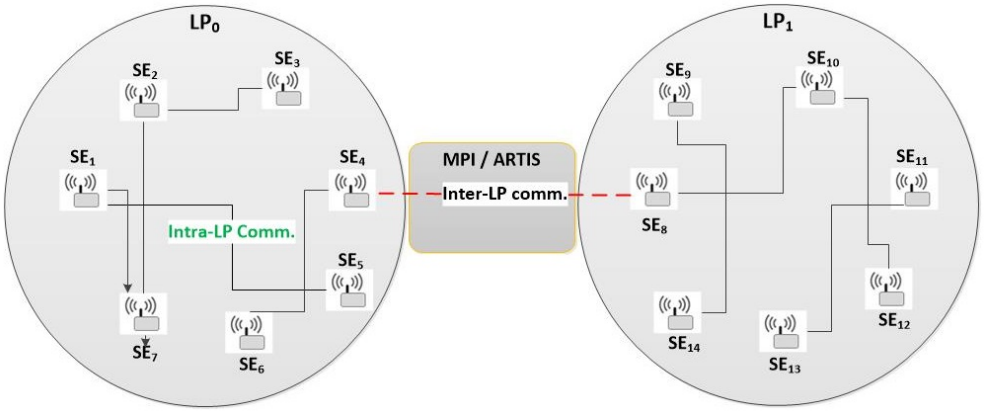

FIGURE 4.6: Inter-LP communication between SEs.

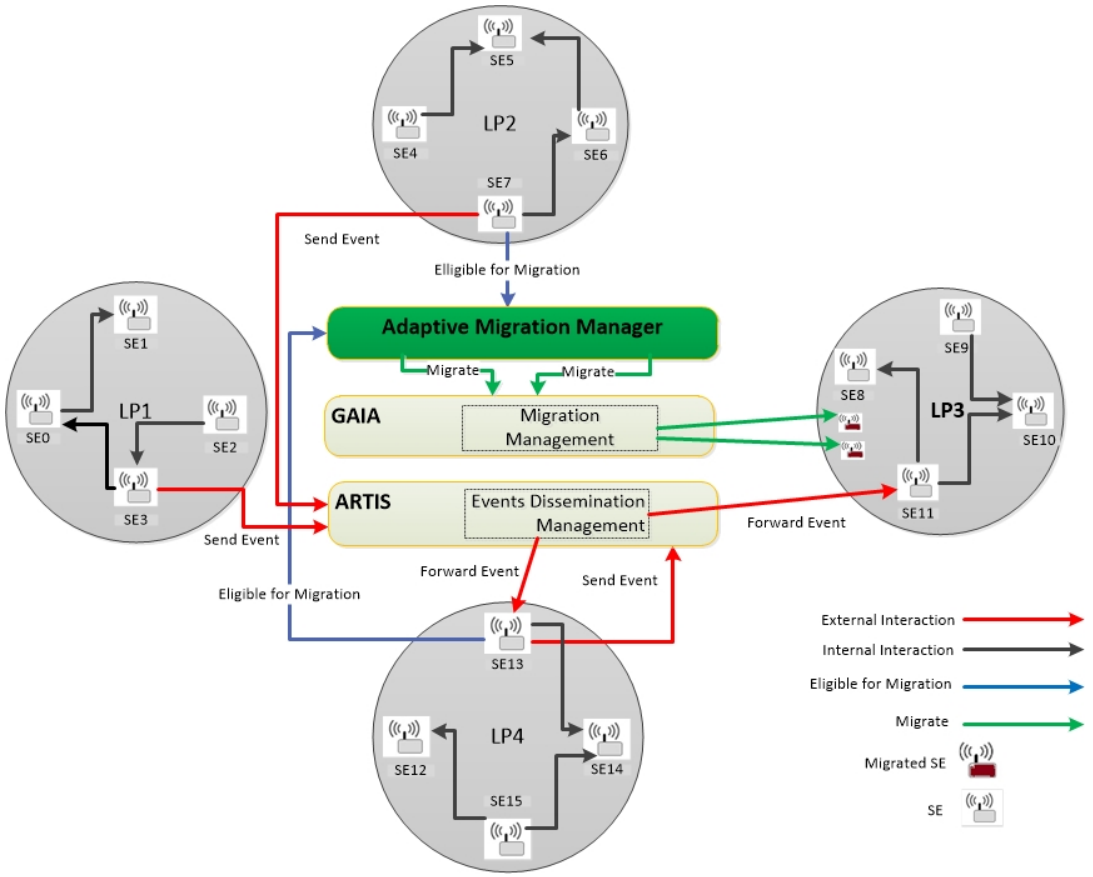

FIGURE 4.7: SE internal/external interaction and migration execution.

4.3.2 Heuristic Basis for Dynamic Partitioning

The MAHA approach considers the transitive dependency among SEs located on different LPs for migration. Transitive dependency has been explained with an example. Suppose, simulation model shown in Figure 4.7 comprises of multiple simulation nodes that are placed at four LPs where each LP contains part of the simulation model.

SE_7 has most of its interaction with SE_{13} and at the same time SE_{13} has high interaction with SE_{11}. Thus, after a certain amount of time, SE_7 needs to be migrated to LP_2 and SE_{13} needs to be migrated to LP_3. In order to cluster SE_7 with SE_{13}, the SE_7 is first migrated to LP_2 and then moves to LP_3. This will lead to extra overhead in terms of computing resources for moving SE_7 to LP_2 first and then migrating to LP_3. This problem is more specifically known as a transitive dependency. To resolve this issue, this work proposed MAHA algorithm to handle

migrations in PADS environment. The heuristic can be evaluated in two ways; 1) firstly the evaluation of the heuristic is done after each timestep [23]. This evaluation is beneficial for systems that involves a lot of communication among the SEs and there are rare chances that a timestep results in zero interaction. However, with a large number of SEs the cost of heuristic evaluation can lead to extensive computation, that will, in turn, result in performance degradation. 2) The alternate approach for heuristic evaluation uses the interaction size as a decision parameter for evaluation. The evaluation is triggered only if SE has sent at least M messages since the last evaluation. The value of M represents the maximum size of interaction window. In many large-scale systems, this evaluation approach greatly reduces the number of evaluation at each timestep and thus leads to more scalable performance. The base condition used for the evaluation of this approach employs the technique introduced in [23]. The algorithm checks whether there exists SEs that meet the condition of the transitive dependency and thus migration is triggered. WLAN simulation is used to evaluate the performance of the proposed adaptive heuristic algorithm for large-scale parallel and distributed simulation. The wireless communication networks are suitable as the interaction among the wireless nodes is generally location dependent. In wireless networks, all the nodes move due to the mobility changes which lead to a change of the neighbor nodes and network topology. Therefore, the communication pattern among the SEs changes very frequently during the course of the simulation.

4.3.3 MAHA: Migration-based Adaptive Heuristic Algorithm

Algorithm 1 lines 1-2, the values of CommWinSize and MaxWinThreshold are initialized. The CommWinSize and MaxWinThreshold parameters are used in the decision-making pertaining to the short-listing of SEs for migration evaluation. Next, all SEs eligible for migration during each timestep is determined by the algorithm (lines 312). During migration eligibility process, the local and remote communications of a particular SE interactions are added to calculate the total

interactions of that SE at each timestep (lines 5-6). The CommWinSize is determined on the basis of total *SEInteractions* (line 7). At lines 8-9, all SEs that have done sufficient communication are evaluated for the eligibility of possible migration. The information about SE local interaction, remote interaction, and LP are added to hash table SEMigEvalHashtbl (line 10). The hash table is then passed to EvalSEforMig module (in algorithm 2), which is used to find out all SEs that need migration.

Algorithm 1 FindSEMigEval - Finds all SEs Eligible for Migration Evaluation

Input: $S_{SE} = ListofSEs(i.e., SE_1, SE_2, SE_3, , SE_n)$
Output: $SEMigEval_{Hashtbl}$

1: $CommWinSize \leftarrow NULL$
2: $MaxWinThreshold \leftarrow N$
3: **for all** t *timesteps* **do**
4: **for all** LPs **do**
5: **for all** SEs *in the communication* **do**
6: $SE_{Interaction} += SE_{LocInt}$
7: $SE_{Interaction} += SE_{RemInt}$
8: $CommWinSize += SEInteraction$
9: **if** $(CommWinSize >= MaxWinThreshold)$ **then**
10: **Add** SE_j, LP_i, SE_{LocInt}, SE_{RemInt} *to* $SEMigEval_{Hashtbl}$
11: **end if**
12: **end for**
13: **end for**
14: **end for**
15: **EvalSEforMig**$(SEMigEval_{Hashtbl})$
16: **Return** $SEMigEval_{Hashtbl}$

The algorithm 2 determines all the SEs that need to be migrated upon their eligibility. The information regarding the local interaction, remote interaction and LP (i.e., where the SE is located) is provided as an input to Algorithm 2 in the form of SEMigEvalHashtbl. The flaglist, Migration Threshold (migTS), Standard Migration Factor (SMF), and Migration Factor (Migf) are initialized (lines 1-4). The SEs that are eligible for the migration are flagged and necessary details required for migration are added to hash table MigHashTbl (lines 5-11). The Migration Factor (MF) is calculated using SElocInt and SEremInt of the SE (i.e., MigF = SERemInt / SELocInt) as given in line 6. After the determining of MigF, the SE is considered for migration only (line 7) if:

(MigF ¿ SMF) and At least ΔTSSE timesteps having passed since the last evaluation of the corresponding SE.

All the SEs that meets the criteria mentioned at line 7 are added to flagList (line 8). The information regarding source SE, Destination SE, Source LP, and Destination LP is added to hash table MigHashTbl (line 9). This information is given as an input to FlagMig module given in algorithm 3. All the SEs that meets the criteria mentioned at line 7 are added to flagList (line 8). The information regarding source SE, Destination SE, Source LP, and Destination LP is added to hash table MigHashTbl (line 9). This information is given as an input to FlagMig module given in algorithm 3.

Algorithm 2 EvalSEforMig - Finds all SEs to be migrated

Input: $MigEvalHashtbl, SMF, MigF, TS_{SEi}, migTS$
Output: $SEsMigrHashTbl$

1: $flagList[]$
2: $migTS \leftarrow 10$
3: $SMF \leftarrow 1 - 20$
4: $MigF \leftarrow 0$
5: **for all** i *in* $SEMigEval_{Hashtbl}$ **do**
6: $MigF = SE_{RemInt} \,/\, SE_{LocInt}$
7: **if** ($MigF >= SMF$ AND $\Delta TS_{SEi} > migValue$) **then**
8: $flagList[i] = SEi$
9: **Add** $src_{SE}, dest_{SE}, src_{LP}, dest_{LP}$ $toMigHashTbl$
10: **end if**
11: **end for**
12: **FlagMig**(MigHashTbl)
13: **Return** $MigHashTbl$

The first two algorithm represents the EHA [23] approach which is extended by proposed MAHA in Algorithm 3 and 4. Algorithm 3 represents the MAHA core algorithm to determine all those SEs that meet the criterion of transitive dependency required for migration decision. The input of Algorithm 3 is the MigHashTbl containing information regarding Source SE, Destination SE, Source LP, and Destination LP. All the SEs meeting the criterion of transitive dependency are flagged for migration and added to MigHashTbl table (lines 1-11). Each destination SE is compared with all the source SEs one by one and if the Destination SE is Source

SE in another record, then both Destination SEi and Source SEj are migrated to LP where destination SEj is located (line 5-8).

Algorithm 3 FlagMig - Filter SEs based on transitive dependency to be flagged for migration using MAHA approach.

Input: $MigHashTbl$
Output: $SEsMigList$

1: $i \leftarrow 0$
2: $j \leftarrow 0$
3: **for all** i *in* $MigHashTbl$ **do**
4: **for all** j *in* $MigHashTbl$ **do**
5: **if** $(destSEi == srcSEj\ )$ **then**
6: $flagList[i] = SEi$
7: $execMig(destSE[i], destSE[j], MigHashTbl)$
8: $execMig(srcSE[j], destSE[j], MigHashTbl)$
9: **else**
10: $ExecMig(srcSE[i], destSE[i], MigHashTbl)$
11: **end if**
12: **end for**
13: **end for**
14: **Return** $SEsMigList$

Algorithm 4 ExecMig - Filter SEs are migrated to the respective LP.

Input: $SEsMigList$
Output: $FlagedSEsMigrated$

1: $k \leftarrow FlagedSEs$
2: **for all** k *in* $SEsMigList$ **do**
3: $MigrateSEs(srcSE[k], destSE[k], MigHashTbl)$
4: **end for**

The initial placement of SEs on the LPs is static, thus all the SEs are placed at random. Figure 4.8 shows the initial distribution of SEs. All the SEs are random and thus their is a very rare chances of clustered SEs that have most of its interaction as local. The EHA algorithm is applied that find out SEs that have most of its with SEs that are located on the remote LPs. The EHA migrate SEs that have high-end remote communication. Figure 4.9 shows the distribution of SEs across the LPs. A more balanced and better clustering of SEs is observed in Figure 4.9. However, their are still some SEs that need further migration to form a more stable cluster. Moreover the situation tends to be more worst if the

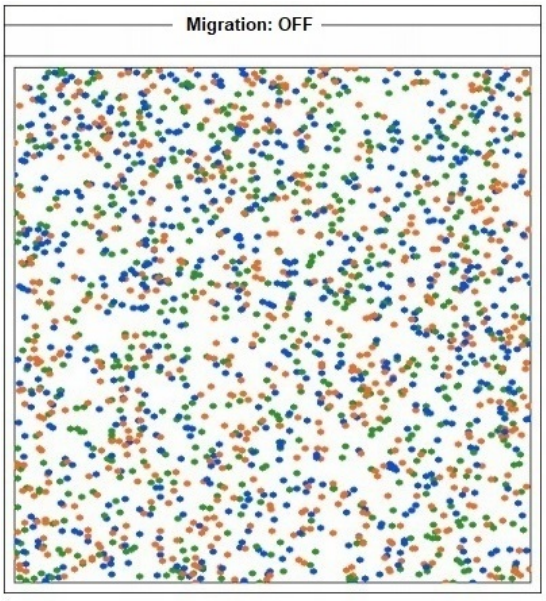

FIGURE 4.8: Migration OFF [95].

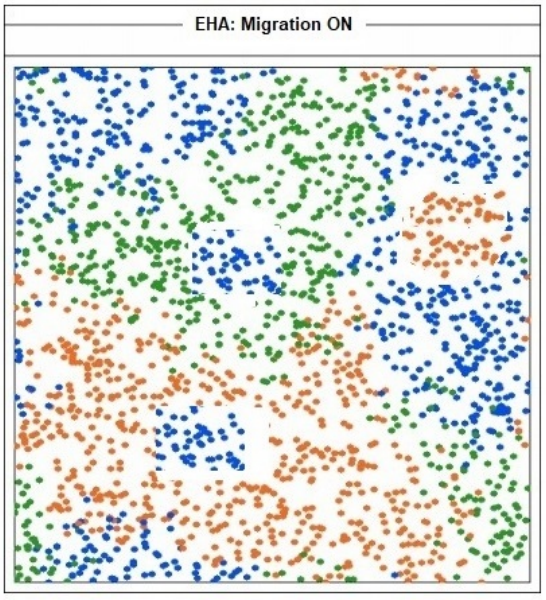

FIGURE 4.9: Migration ON EHA [95].

nodes in the simulation having high mobility thus need very careful decision of migration. One major issue with EHA approach is that if the SEs are migrated to some other LP, the already clustered SEs also will need migration. This will increase the number of migrations to achieve better localization of interaction. The problem is the migration cost involved in executing the required migration.

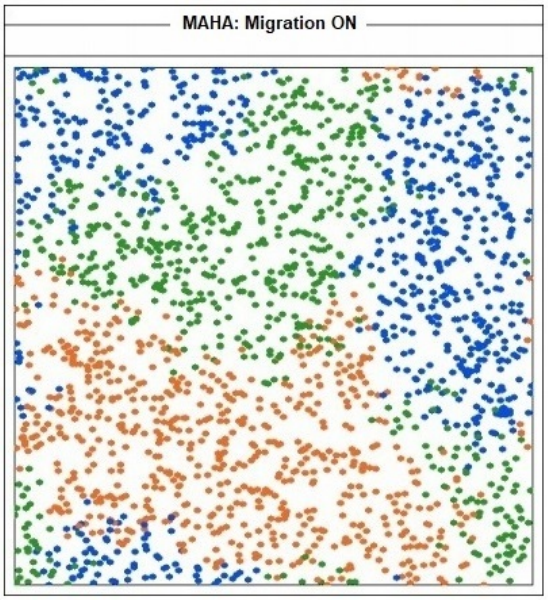

FIGURE 4.10: Migration ON MAHA.

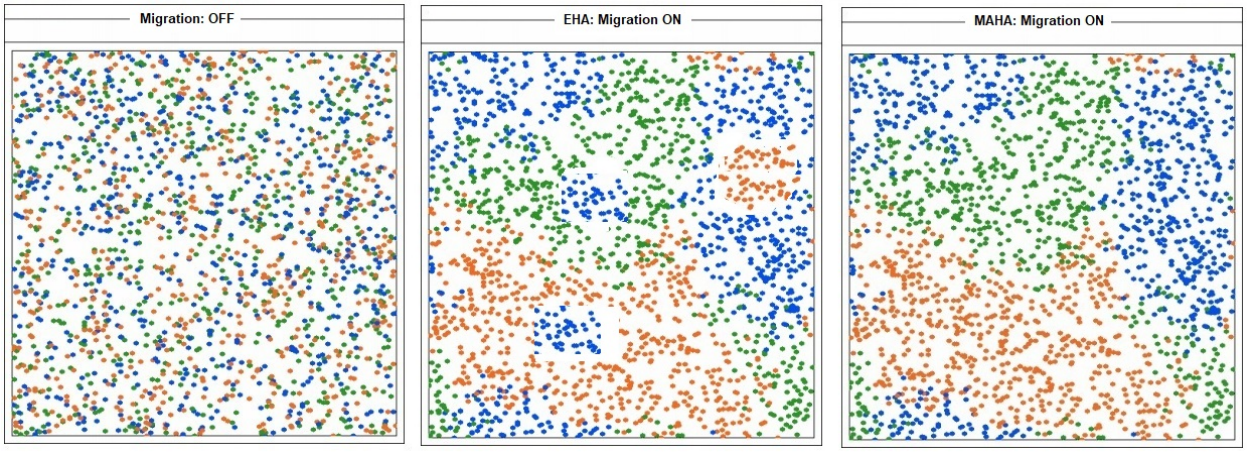

FIGURE 4.11: Migration OFF, EHA and MAHA comparison.

The MAHA approach take advantage of SEs behavior and migration is done keeping in the view the transitive dependency property of SEs interaction. This will lead to achieve better clustering thus leading to a decrease in the total required migrations. Figure 4.10 shows the placement of SEs when the MAHA approach is employed. Figure 4.11 shows the placement of SEs with migration OFF and ON (i.e., EHA and MAHA).

4.3.4 Mathematical Model for Simulation Model Partitioning

The ARTIS/GAIA as Parallel Discrete Event Simulation (PDES) [104] provides flexible platform for the execution of large-scale simulation. The simulation speedup using PDES can be achieved by distributing the simulation over a number of LPs that run in parallel. Each LP maintains their simulation clock independently and is responsible to keep track of the concurrent events execution [52]. ARTIS framework is utilized to perform communication and synchronization management among multiple LPs. To support dynamic partitioning of the simulation model based on the run-time dynamics of the simulation, the GAIA framework is used. The SEs that have high remote communication are analyzed during communication and migration is executed if required to minimize the remote communication. The detailed mathematical modeling of static and dynamic (i.e., SEs migration) partitioning is discussed as below.

Let α represents the set of available LPs that can be obtained as shown in Eq. 4.12

$$\alpha = \{LP_1, LP_2, LP_3, ..., LP_N\} \tag{4.12}$$

and β be the set of SEs in the simulation model as given in Eq. 4.13:

$$\beta = \{SE_1, SE_2, SE_3, ..., SE_n\} \tag{4.13}$$

Initially, the Simulation model is divided into equal number of static partitions i.e., LPs and each is assigned equal number of SEs. The number of SEs on each LP can be obtained as shown in Eq. 4.14

$$\gamma = n/N \tag{4.14}$$

where **n** represents the number of SEs in the simulation and **N** represents the number of LPs. In order to execute parallel simulation, SEs are required to be

distributed across different LPs. The allocation of specific range of SEs on different LPs can be obtained by Eq 4.15

$$LP_i = \{(i-1)(N)+1, (i-1)(N)+2..., (i-1)(N)+N\} \tag{4.15}$$

To execute simulation in parallel, SEs are required to be distributed across different LPs. The simulation execution is started after the initial placement of SEs to their respective LPs. During the simulation execution, SEs communicate locally within the same LP or remotely with SEs located on other LP. The information about local and remote communication is maintained by each of the LP locally and if the remote communication crosses a certain threshold the migration will be required to minimize the high-end remote communication. Let Γ represents the ratio of the remote communication to the local communication as shown in Eq. 4.16

$$\Gamma = SE_{RemInt} \,/\, SE_{LocInt} \tag{4.16}$$

The migration will be required only if

$$\Gamma > \omega \tag{4.17}$$

and at least ρ timestep has passed since last migration as can be seen in Eq. 4.17. ω is standard migration factor used to control the number of migrations. This represents the EHA approach employed for migration execution. The obtained local communication is increase however, the number of migration tends to increase for wireless network with high mobility nodes. The MAHA approach is employed to control the number of migrations achieving an approximate same level of LCR. All SEs that were tagged for migrations are given as an input to the MAHA. The SEs will be finalized for migration only if

```
for all i in MigHashTbl do
  for all j in MigHashTbl do
    if (destSEi == srcSEj ) then
      flagList[i] = SEi
      execMig(destSE[i], destSE[j], MigHashTbl)
      execMig(srcSE[j], destSE[j], MigHashTbl)
    else
      ExecMig(srcSE[i], destSE[i], MigHashTbl)
    end if
  end for
end for
```

4.3.5 Cost Analysis of MAHA Approach

This section discusses the cost associated with the simulation execution when the migration algorithm is implemented for PADS execution.

Given the cost difference between these two communication types, the best performance can be obtained maximizing the local interactions. For this reason, we propose a migration mechanism that re-allocates SEs among the LP. In other words, a mechanism that changes the partitioning configuration. This aims to cluster the SEs that interacts frequently in the same LP, reducing the use of costly inter-LP communications. In Eq. 4.18 all these aspects are considered.

$$OEC = SUC + (LIC + RIC + SynC + MM) + Mig_C \tag{4.18}$$

The new term Mig_C introduced in the previous equation is the Migration Cost. It means that the total execution cost of the PADS now includes the computation and communication costs paid for the reallocation of SEs. More in detail, Mig_C can be seen as composed of few addends as shown in Eq. 4.19.

$$Mig_C = Mig_{CPU} + Mig_{Comm} + Heu_C \tag{4.19}$$

The computation cost of CPU is represented by MigCPU whereas Mig_{Comm} is the cost associated with the cost associated with the SE migration. The Heu_C

correspond to the computation cost required for the implementation of heuristic implementation.

4.4 Chapter Summary

This chapter provides details related to the design and implementation of SIM-Cumulus, an academic Cloud that supports the provisioning of NSaaS to the end-users. SIM-Cumulus provides web-based interfaces that can be used by the users to access the simulation services and execute their simulations in an on-demand fashion. SIM-Cumulus provides several modules, including MPI, to support execution of large-scale wireless network simulations. These strengths make SIM-Cumulus suitable candidate for the simulation of large-scale wireless network simulations. Moreover, MAHA algorithm is proposed and presented that supports migration for PADS simulations on Cloud (i.e., multi-core and distributed architectures).

Chapter 5

Simulation Modeling, Design and Analysis

To evaluate the performance of proposed SIM-Cumulus and MAHA algorithm, we have considered three different large-scale wireless networks (i.e., VANETs, UWSNs and WLANs) simulations. The simulation experiments are divided in to three different sets. The first set of the simulations are executed in sequential and parallel fashion on Cloud instances and stand-alone workstations. The experiments (using VANET simulation) are performed (sequentially) on three stand-alone machines as well as Amazon EC2 instances. The results are plotted for three parameters that are simulation speed in terms of simulation events per second, execution time in number of hours to complete the simulation execution, and energy consumption (i.e., electricity consumed and CO2 emission). The second set of experiments consists of large-scale UWSN simulations on OMNeT++ and WLANs simulations on ARTIS/GAIA simulator. Three workstations and two cloud instances (i.e., SIM-Cumulus and MS Azure) are utilized for the simulation experiments. The large-scale UWSNs simulation are executed in monolithic as well as parallel fashion on all the machines. The simulation results are obtained for simulation speed in number of events per second, execution time, energy consumption, CPU utilization, and LCR. The results of SIM-Cumulus are compared with the workstations as well as with the MS Azure Cloud instances, for sequential and

parallel executions. The third set of simulations are executed for the proposed MAHA approach. Overall, four set of simulation experiments are executed on multi-core and distributed setup respectively. Simulation results in terms of LCR, number of migrations, simulation execution time, and simulation speedup) are obtained to evaluate the performance of MAHA and compared to the EHA. The discussion related to the simulation experiments and obtained results is presented in the following sections.

5.1 VANETs simulation on SIM-Cumulus

The VANETs operates in hybrid deployment mode i.e., *Vehicle to Vehicle* (V2V) and *Vehicle to Infrastructure* (V2I). The computational complexity of network simulation is magnified in hybrid VANET due to the involvement of mobility models and heterogeneous access technologies [105]. Therefore, we consider VANET as a suitable test case for examining the performance of Cloud based environment with respect to network simulations.

We implemented and executed large-scale realistic scenario related to Urban VANET that considers three aspects i.e., *realistic map*, *propagation model*, and *vehicular mobility*. The realistic maps are utilized to represent realistic road trajectories for the vehicular movements. We have considered the stop sign mobility model definitions reported in [106] for further movement related aspects such as acceleration and deceleration of vehicles. Moreover, we have employed free-space line-of-sight propagation model that represents the simplified situation related to signal transmission in an urban environment. Table 5.1 summarizes all the parameters used in defined simulations.

VANET simulations have been performed by utilizing the available *Four* standalone workstations and compared the resource utilization performance with *Amazon Elastic Computing Cloud* (EC2) service i.e., on Windows operating system instances (configured with OPNET simulator representing VPL layer shown in

TABLE 5.1: Simulation parameters.

Parameter	**Value**
Network Simulator	OPNET Modeler
Simulation Area	$2350 \times 2000\ m^2$
Mobility Model	Stop Sign Mobility Wait time at Intersection = 5sec
Nodes Count	250
Base Station Count	4
Access Point Count	13
Mobility (Speed)	15 KM/h
MAC Protocol	IEEE 802.11a
Data Rate	54Mbps
Total Simulation Time	36000 sec
Routing Protocol	AODV
AODV Configurable	Active Route Timeout=3 Allowed Hello Loss = 2
Transmission Power	Base Station = 0.5W Access Point = 0.005W
Propagation Model	FreeSpace Line of Sight
Total Simulation Time	36000 seconds
Topology	Huazhong University of Science Technology's Road Layout

SIM-Cumulus architecture. The amount of CPU that is allocated to a particular instance is expressed in terms of Amazon EC2 Compute Units (ECU).

An ECU provides the relative measure of the integer processing power of an Amazon EC2 instance. A single ECU contains a processor having $1.0 - 1.2$ GHz Xeon processor and 17.1GB memory. For Cloud computation environment, 6.5 Amazon ECUs are employed. ECUs are used to compare CPU capacity between different instance types. Four different workstation (Wks) configurations have been used for comparison with EC2 cloud. Description of these workstations is given in Table 5.2.

TABLE 5.2: Workstations specification.

Machine	***CPU***	***Memory***
`Wks-A`	Intel Core 2 duo 2.53GHz	2GB
`Wks-B`	Intel Core 2 duo 2.20GHz	2GB
`Wks-C`	Intel 2.93GHz	1.28GB
`Wks-D`	Intel 2GHz	1GB

Figure 5.1 shows experimental results of *three* workstations along with Amazon EC2 cloud instance. Figure 5.1(d) represents `Wks-C` execution results for VANET simulation based on parameters referred in Table 5.1. *Y-axis* shows the number of discrete events handled at any particular instance and *X-axis* represents the elapsed simulation time in seconds. *Blue line* shows the current simulation events (per second) whereas *red line* shows the average number of completed simulation events (per seconds). The simulation speed has been observed at an average 37632 events within initial 785 seconds. The current simulation speed (depicted using blue curve) begins from a single event at the start of simulation and gradually rises to 39225 events at its peak during first 785 seconds. For the rest of the simulation time in Figure 5.1 (d), we observe an average simulation speed of 38387 events (per second). For current simulation speed, minimum of 33640 and maximum 37852 events per seconds are observed.

`Wks-B` shown in Figure 5.1(c) shows on average better simulation speed. The simulation speed reaches up to an average of 50000 events (per second) at 23527^{th} second. `Wks-B` is run under the influence of external load to test its impact on simulation speed. Therefore, the current simulation speed shows irregular behavior due to the external load employed on this machine. After this period, the average speed remains above 50000 events (per second) mark and reaches at the peak point of 52277 events (per second).

`Wks-A` is the fastest among the available machines in the experimental setup and it was run without any external load. Simulation results of `Wks-A` are shown in Figure 5.1(b) which are on average 83248 events (per second). The lower point of current simulation speed is observed at the final stages of the simulation as 77198 events (per second) and the maximum point is observed at 83438 events (per second).

The simulation run on `Wks-D` terminated abnormally due to the in-sufficient resources (i.e., memory and low-clocked CPU). The simulation ended immaturely after 2100 minutes (35 hours) and only partial data could be collected from that machine which is discussed in Figures 5.2 and 5.3.

Figure 5.1(a) shows VANET simulation results on Amazon EC2 Cloud instance. The results highlight average speed of 111268 simulation events (per second). The lowest current simulation speed 99736 events (per second) is observed at 22105^{th} second. Two unexpected spikes with respect to the current simulation speed are observed. This behavior possibly occurs due to the instant availability of more computing resources: allocation of more number of CPUs, more amount of memory, etc.

Results observed for Wks-C Figure (5.1(d)) highlight erratic computing pattern compared to the Wks-A Figure (5.1(b)). This computational behavior of Wks-C is due to the higher memory latency employed during DRAM accesses. The Wks-A has low latency memory with higher memory-bandwidth, which resulted in fast memory-access during simulation run. In realistic scenario, computing machines are often not available exclusively for single application execution. Therefore, on Wks-B Figure (5.1(c)) some external load was employed to observe the effect on simulation performance. From start to 18378^{th} second, the simulation performance is effected negatively because of CPU time sharing between the executing simulation and the external load. In summary, these simulation results show that average simulation speed (in terms of events/sec for VANET scenario) on `Wks-A`, `Wks-B`, and `Wks-C` is 20%, 50%, and 62% lower than that of execution speed on Amazon EC2 proposed model, respectively.

Figure 5.2 shows the EC2 performance in terms of total elapsed time regarding simulation execution. Elapsed time is lower than that of stand-alone machines e.g., elapsed time is approximately 9×, 4×, 3×, and 2× lower than that of `Wks-D`, `Wks-C`, `Wks-B`, and `Wks-A`, respectively. Usage cost area in Figure 5.2 symbolizes affiliation of various cost factors associated with the usage of stand-alone computing machines. Zero involvement of this usage cost area with EC2 shows that without up-front investments, maintenance, and electricity costs etc., EC2 would be an appropriate choice for institutions.

For the past few years, sustainable and green computing is receiving much attention across the world [107]. Considering important role of green computing in

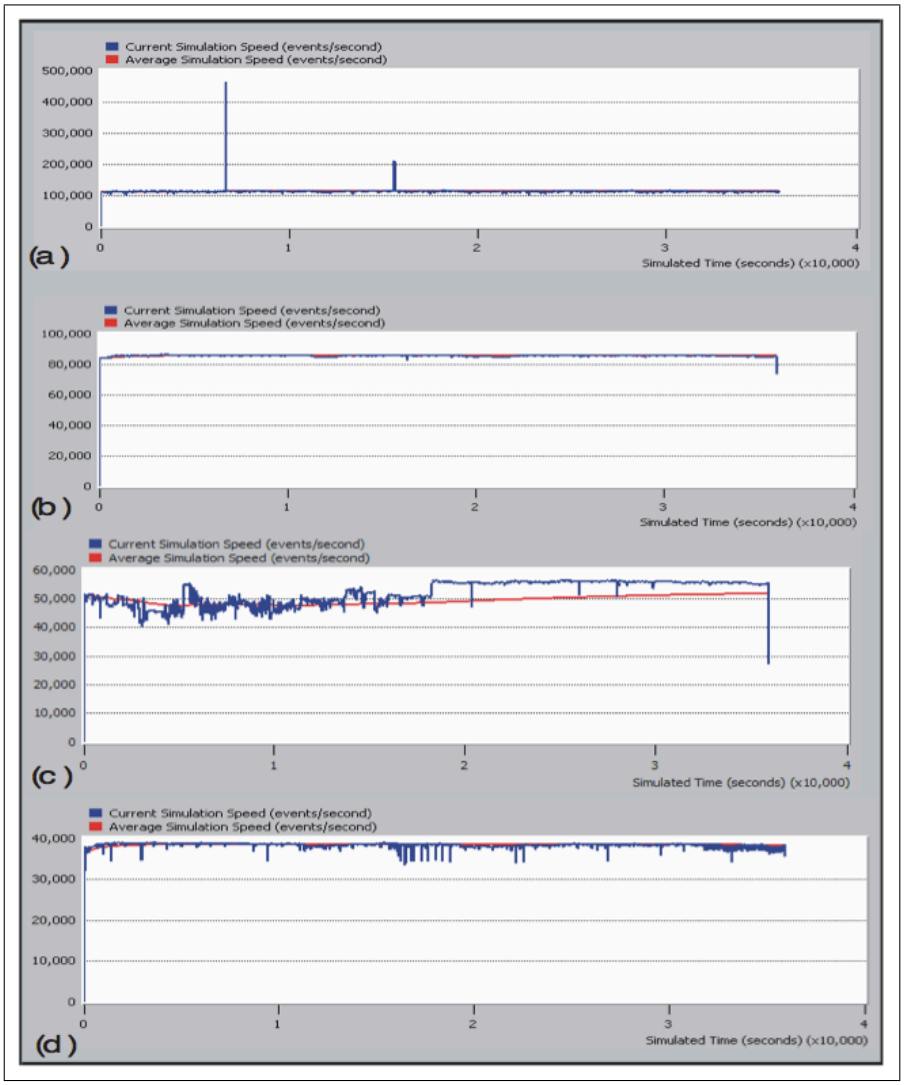

FIGURE 5.1: Average simulation speed of VANET scenario using (a) EC2 Model (b) Wks-A (c) Wks-B (d) Wks-C.

academia, there exists awareness for IT sustainability and green movement for simulating wireless networks. [107, 108] have explained that the cloud computing is a possible game-changer in green IT. High computing and reduced time utilization (for computer application executions) are two significant effects of the proposed cloud usage that decrease electronic waste and electricity consumption within academic institutes. Using `Joulemeter` (a software to estimate the power consumption of desktops, VMs, and individual software running on a computer) [109], this study estimates the consumption of electricity and the emission of *Carbon Dioxide* (CO_2) on stand-alone workstations (with above mentioned configurations) against

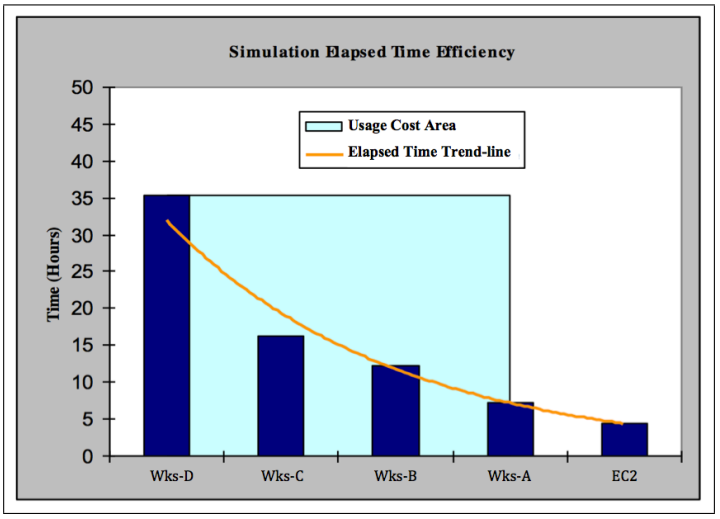

FIGURE 5.2: Simulation elapsed time for large-scale VANET.

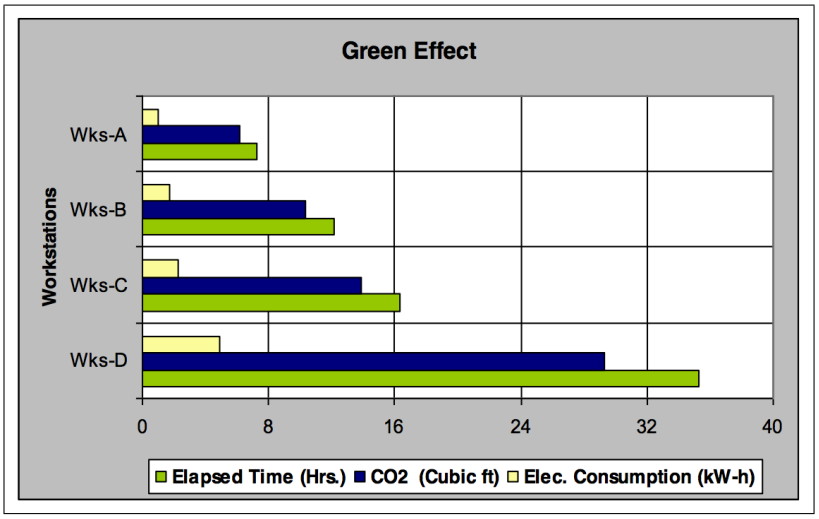

FIGURE 5.3: Electricity consumption and emission of CO_2.

simulation elapsed time (for simulation detail specified in Table 5.1). From Figure 5.3, it becomes clear that both electricity consumption and equivalent estimated emission of CO_2 are directly proportional to the usage time of simulation scenario on stand-alone machine.

Therefore, for execution of simulation scenario that takes approximately 4.35 hours (Figure 5.2) within cloud instance, electricity consumption would be 0.622 kWh with 3.69 ft^3 (calculated from Figure 5.3) of equivalent CO_2 emission (for any local workstation on which cloud has been launched). Given results and comparisons advocate for the adoption of cloud computing for large-scale realistic simulations.

The benefits associated with Cloud in terms of electricity, usage cost, and green IT are shown in Figure 5.3. In order to get better understanding, let's assume that 100 students within an institution (having workstations similar to `Wks-A`) are involved in simulation analysis similar to VANET scenario. `Wks-A` takes 7.26 hours and therefore total electricity consumption for 100 workstations would be 3030 kW-h/month as well as emission of 18480 ft^3 CO_2. On the other hand, same execution takes 4.35 hours within Cloud and therefore electricity consumption (for local workstation on which cloud has been launched) would be 1800 kW-h/month along with the emission of 11070 ft^3 of CO_2 (40% less than execution on `Wks-A`). The results show the advantages associated with the usage of cloud for network simulations in terms of electricity, green IT, and IT sustainability.

5.2 OMNeT++ Simulation on SIM-Cumulus

As discussed earlier, the computation complexity of a network simulation not only depends on the scale of the network but also the granularity of the microscopic details of the network characteristics that a researcher wishes to study. In this study, we have considered two wireless network simulation scenarios on two different network simulators (i.e., GUI-based OMNeT++ and command-line-based ARTIS/GAIA [23]). OMNeT++ being a resource hungry simulator fails to complete the discussed simulations on a stand-alone machine. However, ARTIS/GAIA has the ability to provide results of simulations of even for higher number of nodes with equivalent simulation parameters. The aim of the experiments presented in this work is to show the effectiveness of SIM-Cumulus considering sequential and parallel execution of the network simulations that are either not feasible on stand-alone machines or take extra-ordinary long duration to complete. The effectiveness is evaluated based on reduced simulation execution time and increased rate of completed simulation events by productive distribution of SEs over various number of LPs.

Concerning simulations in OMNeT++, large-scale UWSNs scenario is considered. The UWSNs scenario considers three aspects; mobility, propagation model and routing. We have considered the *Random Way Point* (RWP) mobility model definitions reported in [110] for sensor movement in underwater environment. Moreover, path loss propagation model [111] is employed to simulate the physical layer characteristics of the UWSNs. The opted routing protocol for UWSNs is *Vector-Based Forwarding (VBF)* [112]. The computational complexity of the network simulation is magnified in UWSNs due to the involvement of all considered parameters especially in the case of large-scale deployment [113]. Table 5.3 summarizes all the parameters used in the OMNeT++ simulations.

TABLE 5.3: Simulation parameters

Parameter	**Value**
Simulation Platform	OMNet++
Simulation Area	1500 * 1500 m^2
Mobility Model	Random way point
Node Count	1000
Transmission Range	100m
Mobility (Speed)	2mps
Queue Size	14
MAC Protocol	IEEE 802.15.4
Data Rate	2Mbps
Routing Protocol	VBF [112]
Frame Capacity	10
rtsThresholdBytes	3000B
Simulation Time	100 seconds

In addition to these simulation parameters, the GUI environment has high memory and computational requirements to maintain the state of each node [114]. Therefore, we have considered a large-scale UWSNs as a suitable test-case to evaluate the performance of Cloud-based simulation environment. Simulations are executed multiple times with different number of threads during a single run. The obtained results on SIM-Cumulus are compared with Cloud instances of MS Azure [115] and two workstations. Description of available instances and available workstations is depicted in Table 5.6.

TABLE 5.4: Platform specification

Machine	CPU	Memory	No of Cores
Wks-1	Intel 1.7GHz	4GB	4
Wks-2	Intel 2.50GHz	4GB	4
MS-Azure	Intel 2.60GHz	8GB	8
SIM-Cumulus	Intel 2.66GHz	8GB	8

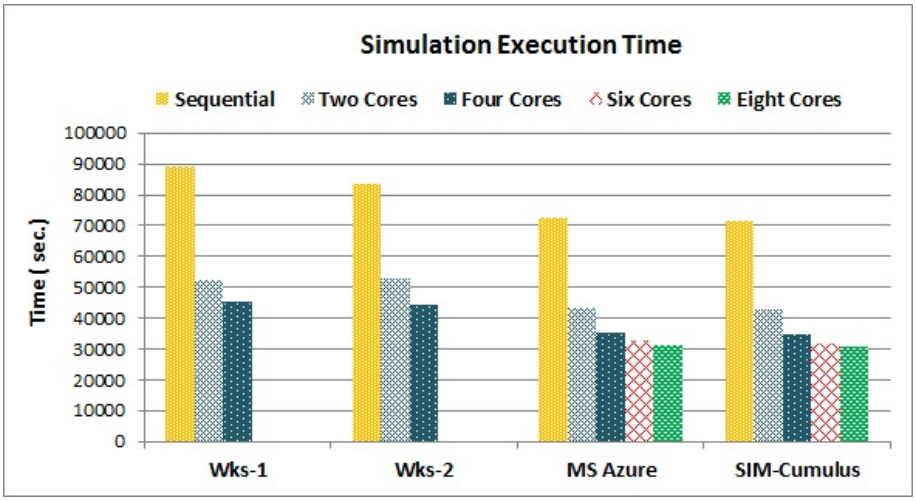

FIGURE 5.4: Simulation execution time.

We have utilized five metrics i.e., *execution time*, *simulation speed*, *CPU utilization*, *Green IT effect* and *Local to Remote Communication Ratio* (LRR) to quantify the performance of SIM-Cumulus. The configuration parameters of the simulation are shown in Table 5.3. The experimental results regarding simulation execution time are presented in Figure 5.4.

Results in Figure 5.4 indicate that the Cloud-based execution of the simulation takes up to 22% less execution time compared to simulations performed on stand-alone workstations. This trend is due to the difference in resources (i.e., high CPU and RAM) available on Cloud-based instances as compared to the stand-alone workstations. The parallel execution of the simulation, using 2 processor cores, results in a decrease of 37-41% in execution time over the sequential execution on all the machines. These results yield good scalability of the parallel execution for 2 CPU cores. Performing a simulation using different number of cores (i.e., 4, 6 and 8), the simulation execution time is further reduced to 71%. Overall, the results are promising and assert that employing the Cloud-based computing

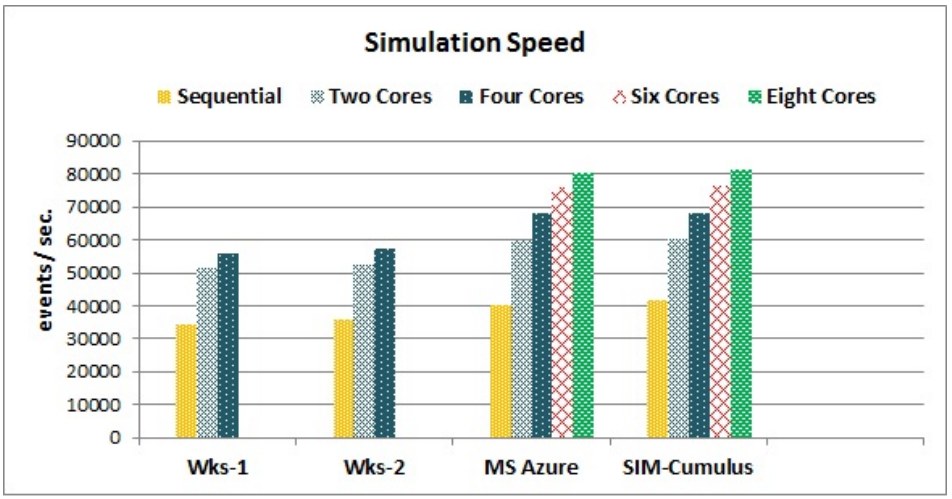

FIGURE 5.5: SSimulation speed (events/second.

infrastructure reduces execution time considerably. Moreover, it attains better speedup as compared to the stand-alone machines.

The large-scale UWSNs simulation comprises of a large number of nodes and is ultimately able to generate a huge number of simulation events. The performance of simulation executions in terms of simulation speed is shown in Figure 5.5. Employing multiple cores for Wks-1 and Wks-2 resulted in improved performance in terms of number of simulated events per second. Simulation executions on the MS Azure [115] and SIM-Cumulus have achieved better results in terms of events per seconds. The overall simulation results ultimately supports the usage of SIM-Cumulus for large-scale wireless networks. For the past few years, sustainable and green computing has grabbed the attention across the world [116, 117]. Cloud computing is considered the appropriate choice for achieving green computing [107]. In this study, we have utilized Joulemeter [109] to approximate the power-consumption of workstations, VMs, and an individual application execution on Cloud (i.e., MS Azure and SIM-Cumulus) instances. The Joulemeter uses separate models to calculate the power usage for CPU, Memory, and Disk. In order to measure the energy consumption of CPU, the processor utilization during active and idle time are used. The power used by the memory represents the last level cache (LLC) misses during a certain time. The disk reads and writes are used to calculate the energy consumed by the disk usage. Figure 5.6 presents CO_2 emission and energy-consumption of different employed multi-core

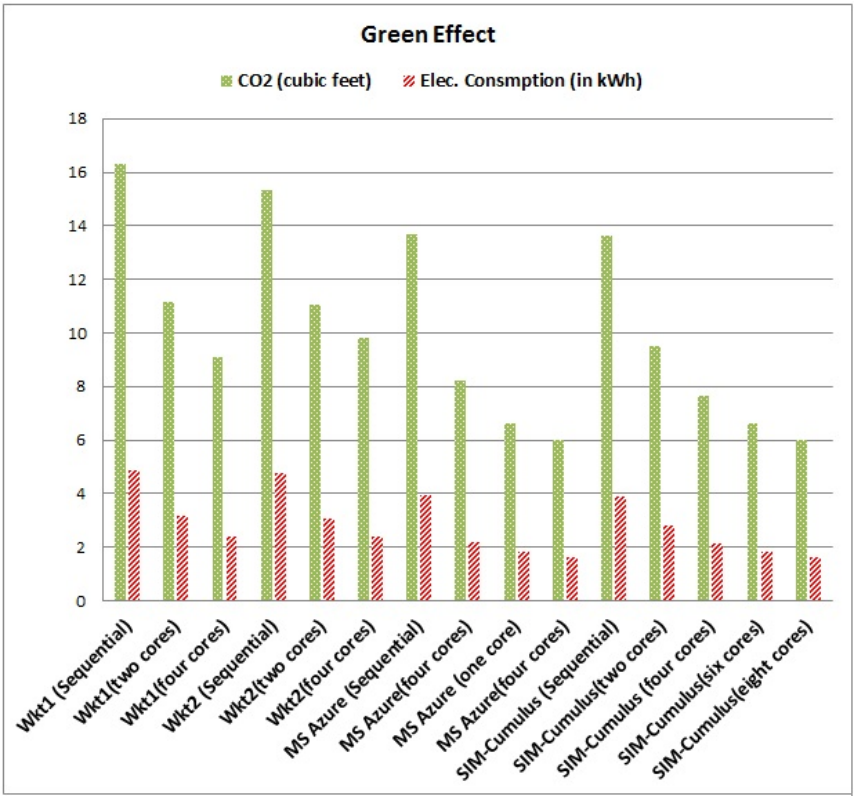

FIGURE 5.6: CO_2 emission and electricity consumption.

simulation machines. Simulation results show that sequential execution on Cloud instances results in 19% decrease of energy consumption, compared to the stand-alone workstations. Moreover, energy consumption is significantly decreased for parallel simulation in the case of 2 and 4 cores. With the increase in number of LPs, look-ahead delay [118] tends to disrupt the total execution time. Thus further increase in the number of LPs does not lead to significant improvement in terms of energy reduction, but still lead to some energy reduction.

In most of the parallel and distributed simulations, the major hurdle in achieving the required speedup is a share of high remote communication involved in the simulation. To provide an insight into the performance gain, we obtained results regarding the LRR. LRR is the ratio of local communication of SEs to remote communication for the LPs. Figure 5.7 to Figure 5.10 represent the obtained LRR on the available machines (having different number of LPs). The simulation execution was initially run with a single LP and afterwards it was repeated several times with a different number of LPs (i.e., 2, 4, 6, and 8). Results shown in

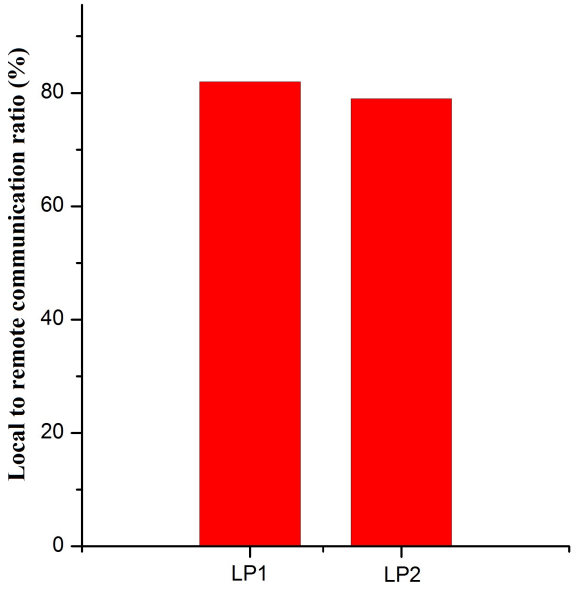

FIGURE 5.7: LRR with 2 LPs.

Figure 5.7 and Figure 5.8 exhibit an average increase of 15% and 19% in remote communication on the machine with 2 and 4 LPs respectively as compared to monolithic execution. This increase directs to a good load distribution of SEs. However, this adds look-ahead cost [118], required for the transmission of packets among SEs located on different LPs.

The results shown in Figure 5.10 demonstrate imbalanced communication pattern between available LPs. The results on the LP_1, LP_4, and LP_6 show average LRR

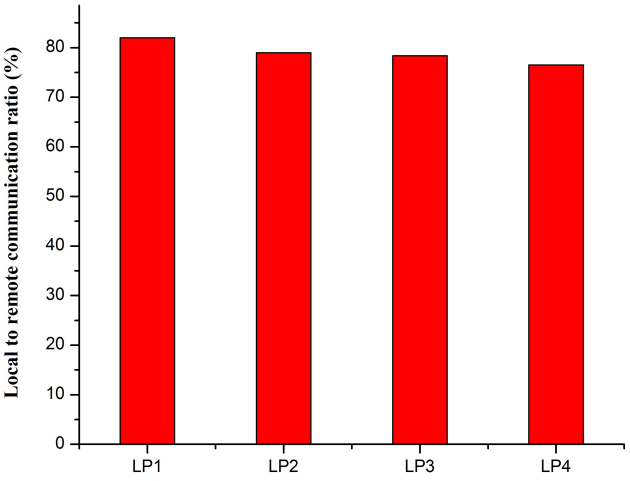

FIGURE 5.8: LRR with 4 LPs.

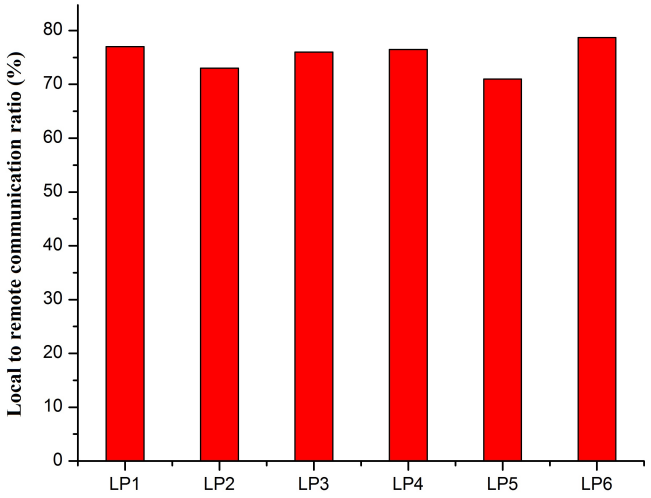

FIGURE 5.9: LRR with 6 LPs.

value of 78% whereas the attained LRR ratio on LP_5 and LP_7 is reduced to 70%. The obtained results of different LPs yield that the speedup is obtained using parallel simulation. However, the look-ahead cost may disrupt the attained speediness.

Figure 5.11 and Figure 5.12 show the results pertaining to the CPU utilization on different cores of a workstation (*Wks-1*) and a Cloud instance (*SIM-Cumulus*). A high CPU utilization up to 95% is observed for the simulation execution. For Wks-1, the average CPU utilization observed for the processor cores is 70.03-79.3% as shown in Figure 5.11.

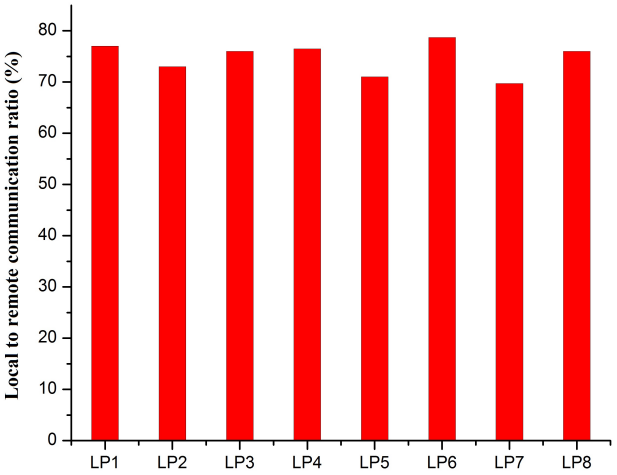

FIGURE 5.10: LRR with 8 LPs.

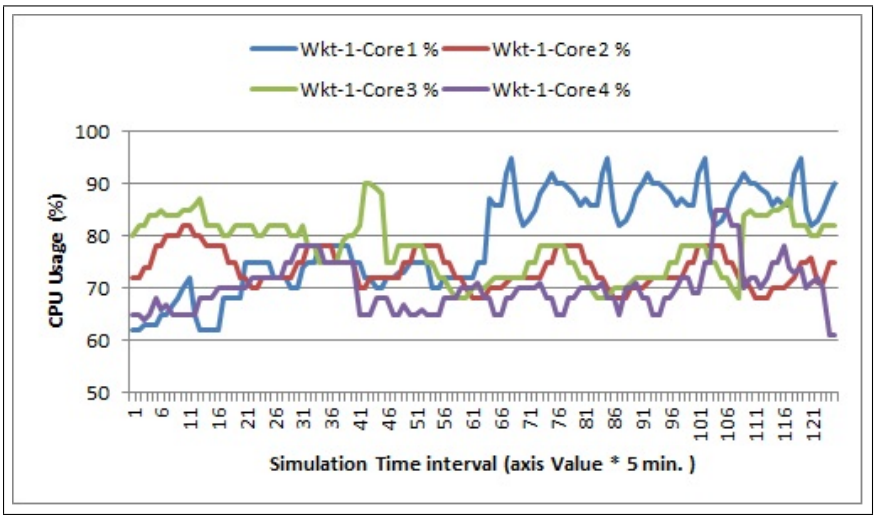

FIGURE 5.11: CPU usage on Wks-1.

For the proposed academic cloud SIM-Cumulus, the CPU utilization observed is on average 72.4-85.5% as shown in Figure 5.12. Results show imbalanced CPU utilization on different cores (of stand-alone workstations and Cloud instances) during the course of the simulation. The resultant imbalance is due to the difference in communication patterns among the SEs located on different LPs. In most of the cases, the real world systems have physical characteristics that have a clear effect on the interaction dynamics. It is possible to exploit such characteristics to re-arrange partitioning at runtime (dynamic partitions) of the simulation components that may lead to significant benefits. Dynamic partitioning will result in two benefits. Firstly, reduce the high cost of inter-communication among the SEs on

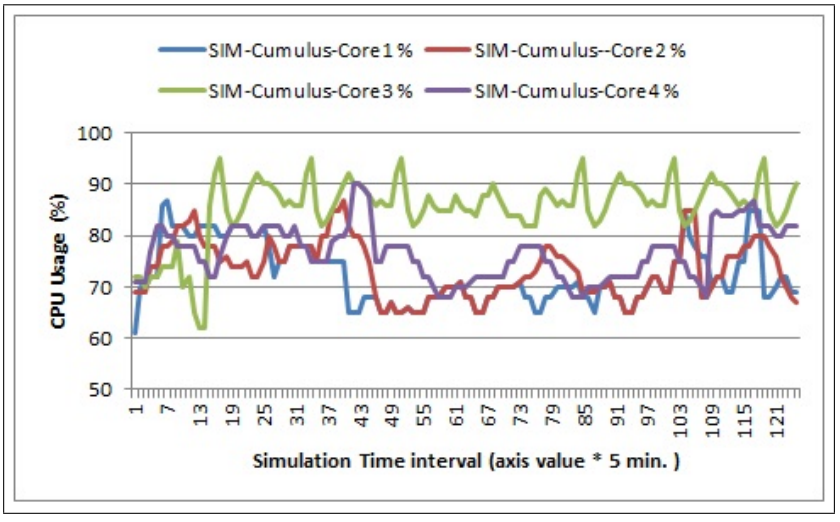

FIGURE 5.12: CPU usage on SIM-Cumulus.

different LPs and secondly, perform load balance to achieve simulation execution speedup.

5.3 ARTIS/GAIA simulation setup and result discussion

In order to show the effectiveness of SIM-Cumulus with other network simulators, we have performed simulations using ARTIS/GAIA. Considering large-scale wireless network scenario in ARTIS/GAIA simulations, a total of 10,000 SEs (wireless nodes) were distributed on 8 LPs. The SEs interact with other SEs that are within the range of 250 spaceunits. Each SE is placed at random position using 2-Dimensional plane and is assigned to a certain LP. An equal number of SEs are placed at each LP. However, the assignment of a specific SE on certain LP is random. Each simulation run is executed for 100 seconds, keeping the simulation area 10000 x 10000 spaceunits. The probability of interaction (where each SE can communicate at a given timestep during the simulation) is set to 0.5. This means that during a certain timestep, half of the SEs are allowed to send messages. The mobility model is RWP (Random way point) and mobility speed is in the range of 1-25 spaceunits per second. We have used two metrics i.e., *simulation execution time* and *simulation speed (events/seconds)* to quantify the performance of SIM-Cumulus. The configuration parameters of the simulation are described in section IV. The simulation is executed in sequential and parallel (i.e., 2, 4, 6, and 8 LPs) modes on the SIM-Cumulus instance. The results are plotted for each independent simulation execution. The experimental results regarding simulation execution time are presented in Figure 5.13. Results reveal that parallel (i.e., 2 LPs) execution of the simulation takes up to 26% less time compared to simulations performed in sequential fashion. The parallel execution of the simulation, using 4-6 processor cores, resulted in a decrease of 47-61% in execution time over

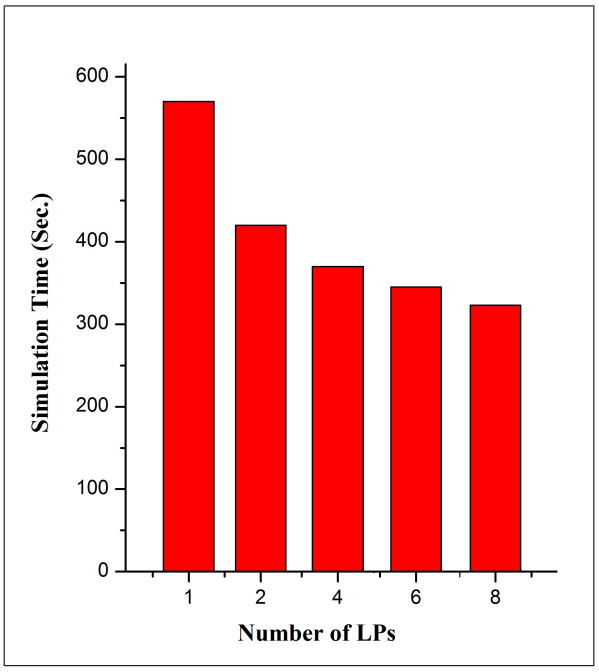

FIGURE 5.13: Simulation execution time.

the sequential execution. These results yield good scalability of the parallel execution for 8 CPU cores. The simulation execution time is further reduced to 72% for simulation that involves 8 cores.

The large-scale wireless simulation in ARTIS/GAIA comprises of a large number of nodes and is ultimately able to generate a huge number of simulation events. The performance of simulation executions in terms of simulation speed is shown in Figure 5.14.

An increase of 35% in simulation events per second is observed for parallel simulation execution (i.e., 2 LPs) as compared to the sequential execution. The obtained results elaborate that employing multiple cores for SIM-Cumulus instance resulted in improved performance in terms of number of simulated events per second. The overall simulation results ultimately supports the usage of SIM-Cumulus for large-scale wireless networks.

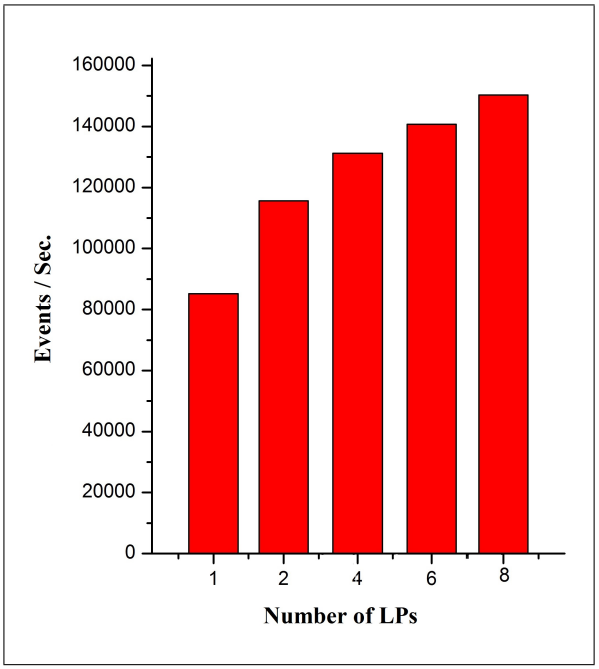

FIGURE 5.14: Simulation speed.

5.4 Performance evaluation of MAHA

For simulation experiments concerning MAHA and EHA, we consider large-scale WLANs network with microscopic details and high-density nodes. The computational complexity of the network simulation is magnified in wireless networks due to the involvement of continuous node (i.e., SE) mobility; especially in the case of large-scale deployment [113]. The choice of the model is Agent-based approach [23], where the agents correspond to the SEs. The simulation area comprises of two dimensions wherein each agent is allowed to move freely. All the agents communicate with their neighbor nodes that are located in their proximity. In other words, if any SE has a new interaction, it will be forwarded to all SEs that are situated within a threshold distance. To comply with the interaction requirements, the Random Waypoint (RWP) mobility model is selected [119]. It is one of the most prevalent mobility models that allows unrestricted movement of all of the agents in the entire simulated area and these agents are not correlated with each other. To obtain the simulation results, the experiments are performed using the proposed algorithm and EHA [23]. Three sets of simulation experiments are executed to scrutinize the performance of the adaptive algorithm.

5.4.1 Simulation 1

In the first round of simulation, 10,000 SEs are distributed on eight LPs. The SEs can send an interaction to other SEs that are located within a range of 250 space units. Each SE is placed in random positions according to two dimensional plane and is assigned to a certain LP. Each LP contain equal number of SEs and the assignment of SE to each LP is performed randomly. The simulations are executed for 3600 seconds, occupying the 10000 x 10000 space units of simulation area. The probability of interaction (where each of the SE can communicate at a given timestep, during the simulation) is set to 0.5. This indicates that during a certain timestep, half of the SEs are allowed to send messages. The mobility model is *RWP* and the mobility speed is in the range of 1-25. The range of MF is set to 1-19 and the migration threshold (MT) is equal to 10.

For every MT value and mobility speed value in the specified range, an independent simulation is executed. Figure 5.15 plots the relationship between the total number of migrations, the obtained LCR and the mobility speed. The results reveal that for low-speed mobility values, a limited number of migrations can lead to a very high LCR values. The important observation is that for a static allocation of SEs on 8 LPs, the reported LCR value is 24%. For moderate speed values, the

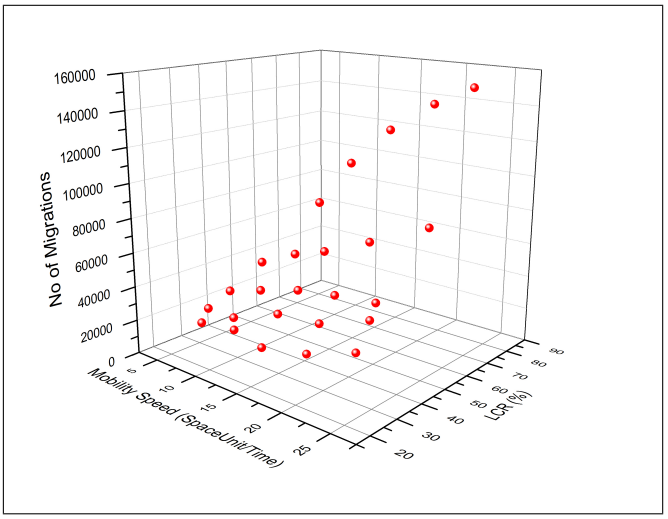

FIGURE 5.15: LCR obtained with a given speed and with an increasing number of migrations.

obtained LCR value rises up to 81%. For higher mobility speed, a good clustering is obtained, however, leading to the escalation in the number of migrations.

5.4.2 Simulation 2

To obtain an insight into the performance of the proposed adaptive algorithm, with respect to the LCR obtained, the simulation is executed multiple times, by employing different number of LPs (i.e., 2, 4, 8, 16 and 32) during each simulation run. The partitioning of the simulation model into more and more LPS results in smaller clusters, which ultimately lead to the decrease in overall local communication. In this simulation, performance is investigated with respect to the ΔLCR gain and MR value when the number of LPs is in 2-32 range. Here, the ΔLCR is defined as the difference between the average value of local communication ratio when the migration status is on and off written as given in Eq. 5.1

$$\Delta LCR = LCR_{MigrationON} - LCR_{MigrationOFF} \tag{5.1}$$

The positive value of ΔLCR means that the employed clustering technique is able to cluster the interacting SEs in a better way, whereas a negative or zero means that the approach is unable to cluster the SEs merely added the processing overhead. All the parameters of this simulation experiment are same as that of simulation except MR and the mobility speed that is set to 11 spaceunits per timestep. Figure 5.16 presents the results of gain in local communication (ΔLCR) when the simulation is executed multiple times for different number of LPs (2, 4, 8, 16 and 32) in each execution. The results in Figure 5.16 demonstrate that for a moderate number of LPs, the proposed algorithm is able to achieve large LCR (61 to 65%) gain. When the simulation is divided into smaller partitions, it will be more difficult to produce effective clustering among the interacting SEs, even though there is still some LCR gain.

The results in figure 5.17 presents the details regarding number of migrations when the simulation model is partitioned into a different number of LPs during each

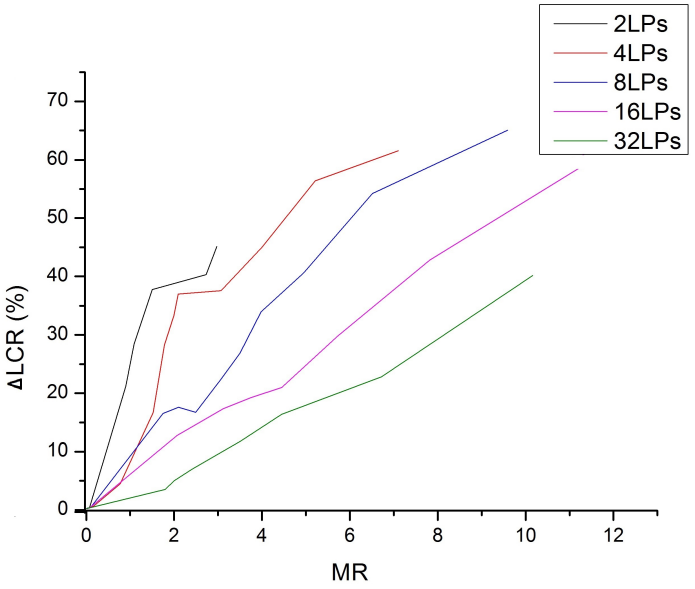

FIGURE 5.16: Effect of the number of LPs on ΔLCR (LCR gain).

simulation run. The simulation is executed multiple times with parameters stated in the initial discussion of simulation 5.4.1. The obtained results demonstrate 15 to 22 % decrease in the number of migrations, while the attained LCR is same for both the approaches (MAHA and existing).

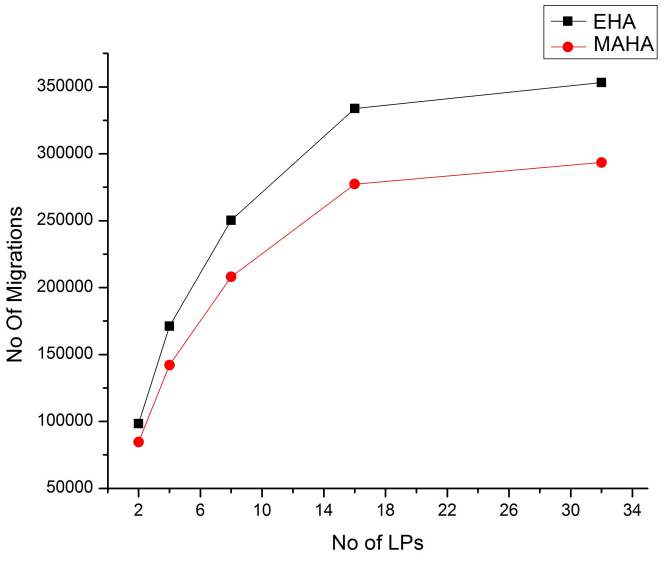

FIGURE 5.17: Migration comparison proposed MAHA approach with existing approach.

5.4.3 Simulation 3

The third simulation experiment is executed to measure the performance of the adaptive algorithm in terms of the attained speedup on the A-SIM-Cumulus Cloud for both parallel and distributed setup. The obtained results are then compared to that of the EHA. The simulation runs over parallel and distributed setup and the obtained execution time (Wall Clock Time (WCT)) with and without the migration is compared. For the parallel execution of the simulation, the A-SIM-Cumulus multi-core VM instance is configured with ARTIS/GAIA middleware. The number of cores allocated to VM is eight and the installed physical memory reserved for VM is 8 GB. The simulation executed multiple times and with different interaction probability (π) that is $\pi = 0.2$, 0.5, and 0.8, while keeping the value of MF to 1.1. This means that if the value of π is set to 0.5, then 50% of the SEs sends interaction (messages) during each timestep. The migration state size is set to 100, 45000, and 80000 bytes. The value 100 is default migration size for the simulations whereas 45000 and 80000 are obtained by padding extra bytes. In addition, the size of the communication message during each interaction is kept 1, 100, and 1024 bytes. The obtained results are average of several independent runs.

In Table 5.5, the obtained results pertaining to the execution time over parallel setup are reported. The simulation runs multiple times with different configurations (interaction size and migration size) for three different dissemination probability values (Low, Moderate, and High) represented by $\pi = 0.2$, 0.5, and 0.8 respectively. In all cases, the simulation runs with migration status set to OFF and recorded the time (SimT) taken to complete the simulation execution. Afterwards, the simulation is executed, using the proposed MAHA algorithm and EHA. To evaluate the performance of the proposed approach, ΔSimT is used to show the obtained speedup. The ΔSimT is defined as the difference between the time taken to complete the simulation with and without migration. It is noted that ΔSimT is used to show the attained speedup for both the approaches (i.e., MAHA and EHA). For all the reported results presented in Table 1, the migration approach

TABLE 5.5: Parallel setup with migration OFF/ON. Different migration and interaction sizes. Different dissemination probability (π).

Input Parameters			$\pi = 0.2$				$\pi = 0.5$				$\pi = 0.8$			
			EHA		MAHA		EHA		MAHA		EHA		MAHA	
Migration Status	Migration Size	Interaction Size	SimT	ΔSimT	SimT	ΔSimT	SimT	ΔSimT	SimT	ΔSimT	SimT	ΔSimT	SimT	ΔSimT
OFF	-	1	110	-	110	-	245	-	245	-	324	-	324	-
ON	100	1	102.5	6.82	102	7.27	223	8.98	217.5	11.22	281.5	13.12	263	18.83
ON	45000	1	105	4.55	104.5	5.00	234	4.49	221.8	9.47	287	11.42	271	16.36
ON	80000	1	107	2.73	106	3.64	236	3.67	226	7.76	291	10.19	280	13.68
OFF	-	200	118	-	118	-	279	-	279	-	395	-	395	-
ON	100	200	108	8.47	104.5	11.44	256	8.24	242	13.26	339.5	14.05	326	17.47
ON	45000	200	109.5	7.20	108	8.45	262	6.09	249.1	10.72	341	13.67	332	15.95
ON	80000	200	110.4	6.44	109.5	7.20	263.2	5.66	251	10.04	342.5	13.29	340	13.92
OFF	-	1024	147	-	147	-	365	-	365	-	461	-	461	-
ON	100	1024	122	17.01	117.5	20.07	295	19.18	273.5	25.07	367	20.39	332.5	27.87
ON	80000	1024	126.2	14.29	121	17.69	302	17.26	280.2	23.23	369.4	19.87	344	25.38

is able to obtain a performance gain (2.73% for EHA and 3.64% for MAHA) as highlighted in Table 1. The worst case results are obtained when the migration size is increased up to 80000 bytes and the interaction size is 1 byte. Even though the migration mechanism is able to save costly remote communication for cheaper local communication cost (LCC). However, this results in an increase of cost paid for the migration and thus the gain in performance is narrow. On the other hand, the best results obtained are 20.39% for EHA and 28% for MAHA when the interaction size is large (i.e., 1024 bytes) and the migration size is small (i.e., 100 bytes). It is worth noting that the best results are obtained when the MF is decreased to 1, thus leading to an increase in the number of migrations. The increase in the interaction dissemination probability also has a greater impact on the increase in the total interaction among SEs in the simulation. To obtain a detailed impact of MF on the performance of the proposed algorithm, the worst case and best case results are further investigated. For all the values of MF, the gain and loss in execution time are reported in Figure 5.18. The simulation is executed with two different configurations (i.e., in first experiment the Migration size is kept 100 bytes and interaction size 1024 bytes, and in the second experiment the Migration size is kept 80000 bytes and Interaction size 1 byte). The Y-axis represents ΔSimT (i.e., performance gain in terms of the execution time) and MF is represented on X-axis. The proposed MAHA algorithm outperforms EHA in terms of execution time for both the configurations. The best results obtained are 20.39% for EHA and 28% for MAHA when the interaction size is large (i.e., 1024 bytes) and the migration size is small (i.e., 100 bytes). The results in Figure 5.18 reveal that for MF value in the range of [1-10], EHA and MAHA obtain a performance gain for both the approaches on both configurations. However, the increase in MF beyond 11 leads to a decrease in performance gain.

The simulation is also executed on the distributed setup (A-SIM-Cumulus Cloud). The detailed specification for Cloud instances used in the simulation is given in Table 5.6.

The parameters chosen for parallel simulation are also used for distributed simulation setup. The obtained simulation results are reported in Table 5.7. All the

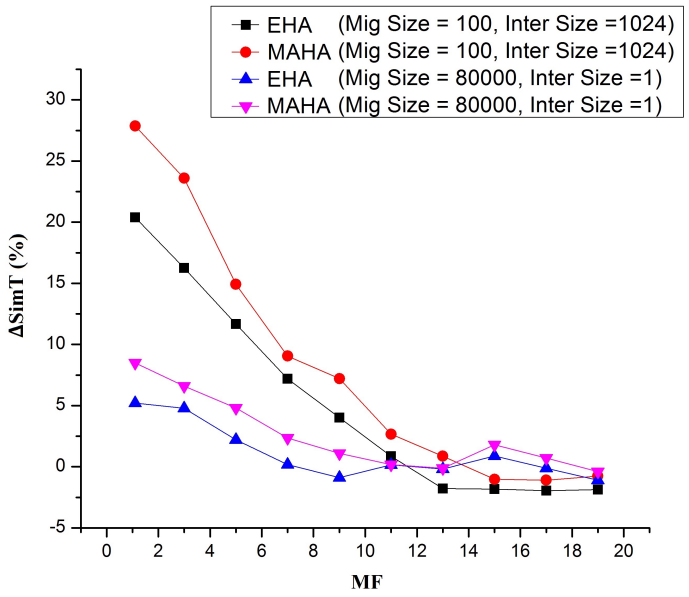

FIGURE 5.18: ΔSimT for parallel setup when the value is in the range 1-19.

TABLE 5.6: Platform specification

Machine	CPU	Memory	No of Cores	Operating System
VM1	Intel Core i3 1.70 GHz	4GB	4	Ubuntu 12.04.5 LTS
VM2	Intel Core i7 3.40 GHz	4GB	4	Ubuntu 12.04.5 LTS
VM3	Intel Core i7 3.40 GHz	8GB	8	Ubuntu 12.04.5 LTS
VM4	Intel Core i5 3.20 GHz	8GB	8	Ubuntu 12.04.5 LTS

configuration in terms of migration size and interaction size is tested for different dissemination probability (i.e., low, moderate, and high). Initially, the simulation runs without migration option and the simulation time (SimT) is measured for different runs. The obtained results are then stated as ΔSimT as shown in Table 3. ΔSimT is the difference between the time required to execute the simulation, with and without migration. In order to find the best configuration, the simulation is executed with different MF value in the range from 1 to 19. The best and worst case results for both MAHA and EHA are represented as italic face in Table 5.7. The best results obtained on EHA is 64% and on MAHA bumped up to 80% when the interaction size is 1024 and the migration size is 100 bytes. The gain is positive for the simulation where the value of MF is less than or equal to

TABLE 5.7: Distributed setup with migration OFF/ON. Different migration and interaction sizes. Different dissemination probability (π).

Input Parameters			$\pi = 0.2$				$\pi = 0.5$				$\pi = 0.8$			
			EHA		MAHA		EHA		MAHA		EHA		MAHA	
Migration Status	Migration Size	Interaction Size	SimT	ΔSimT	SimT	ΔSimT	SimT	ΔSimT	SimT	ΔSimT	SimT	ΔSimT	SimT	ΔSimT
OFF	-	1	801	-	- 801	2178	-	2178	-	3178	-	3178	-	
ON	1000	1	745	6.99	730	8.86	1965	9.78	1899	12.81	2789	12.24	2610	17.87
ON	45000	1	819	-2.25	779	2.75	1998	8.26	1926	11.57	2876	9.5	2705	14.88
ON	80000	1	825	-3.00	804	-0.37	2023	7.12	1982	9.00	2901	8.72	2789	12.24
OFF	0	200	1159	-	1159	-	2350	-	2350	-	3590	-	3590	-
ON	1000	200	1046	9.75	989	14.67	2110	10.21	2056	12.51	3019	15.91	2676	25.46
ON	45000	200	1098	5.26	1026	11.48	2170	11.76	2110	10.21	3112	13.31	2870	20.06
ON	80000	200	1121	3.28	1067	7.94	2208	6.04	2156	8.26	3270	8.91	2915	18.80
OFF	0	1024	2890	-	2890	-	7550	-	7550	-	9866	-	9866	-
ON	1000	1024	1199	58.51	989	65.94	2879	61.84	1767	76.60	3566	63.86	1984	79.89
ON	45000	1024	1265	56.23	1140	60.94	3150	58.28	1925	74.50	3965	59.81	2450	75.17
ON	80000	1024	1304	54.88	1189	58.86	3357	55.30	2377	68.52	4488	54.51	2689	72.74

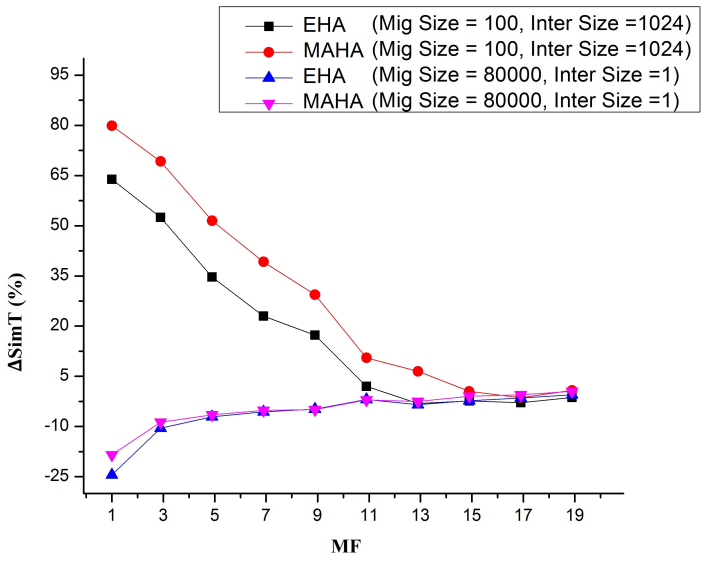

FIGURE 5.19: ΔSimT for distributed setup when the value is in the range 1-19.

10 which tends to decrease when MF is greater than 10.

The best and worst speedup (ΔSimT) results for the distributed simulation configuration are plotted in Figure 5.19. The simulation was executed with two different configurations (i.e., in first experiment the Migration size is kept to 100 bytes and interaction size to 1024 bytes, and in the second experiment the Migration size is kept to 80000 bytes and interaction size 1 byte). The Y-axis represents SimT (i.e., performance gain in terms of the execution time) and MF is represented on X-axis. The results in Figure 5.19 demonstrate that for small value of MF (i.e., 1), the proposed algorithm MAHA is able to achieve 16% improvement in execution time as compared to EHA when the interaction size is large (i.e., 1024 bytes) and the migration size is small (i.e., 100 bytes). From the given results it can be said that the look-ahead delay has a clear impact on the time required to complete the simulation execution.

5.4.4 Simulation 4

The last simulation experiment is executed to measure the performance of MAHA approach in terms of simulation events per second and execution time on the A-SIM-Cumulus Cloud for parallel setup. The obtained results are then compared to that of the EHA. The simulation runs over parallel setup and the obtained execution time (simulation time (WCT)) with and without the migration. For the parallel execution of the simulation, the A-SIM-Cumulus multi-core VM instance is configured with ARTIS/GAIA middleware. The number of LPs used in the simulation is set to 2, 4, 8 ,16 and 32 and the installed physical memory reserved for VM is eight GB. The simulation executed multiple times and with interaction probability (π) $\pi = 0.5$, while keeping the value of MF (Migration factor) to 1.1. This means that if the value of π is set to 0.5, then 50% of the SEs sends interaction (messages) during each timestep. The migration state size is set to 100 bytes. The mobility speed is kept 10 meters per second and the number of SEs in the simulation run is 20,000. The obtained results are average of several independent runs.

The best case and worst case execution time WCT for the parallel simulation configuration are plotted in Figure 5.20. The simulation took 650 seconds when the number of LPs was two and there was no migration involved. A decrease of 11% and 17% in execution time is observed for EHA and MAHA, respectively. A good speed up of 15% and 21% in simulation execution is observed when the number of LPs are increased up to 8. The best results obtained on EHA is 17% and on MAHA increased to 23% for 16 or more LPs. Overall the results show better execution time when the migration is employed for MAHA on multi-core systems as compared to EHA and without migration approach. The large-scale wireless simulation in ARTIS/GAIA comprises of a large number of nodes and is ultimately able to generate a huge number of simulation events. The number of events per second shows the simulation efficiency in terms of simulation speedup. The performance of simulation executions in terms of simulation speed for three different simulation setups (i.e., migration OFF, EHA and MAHA) is shown in Figure 5.21.

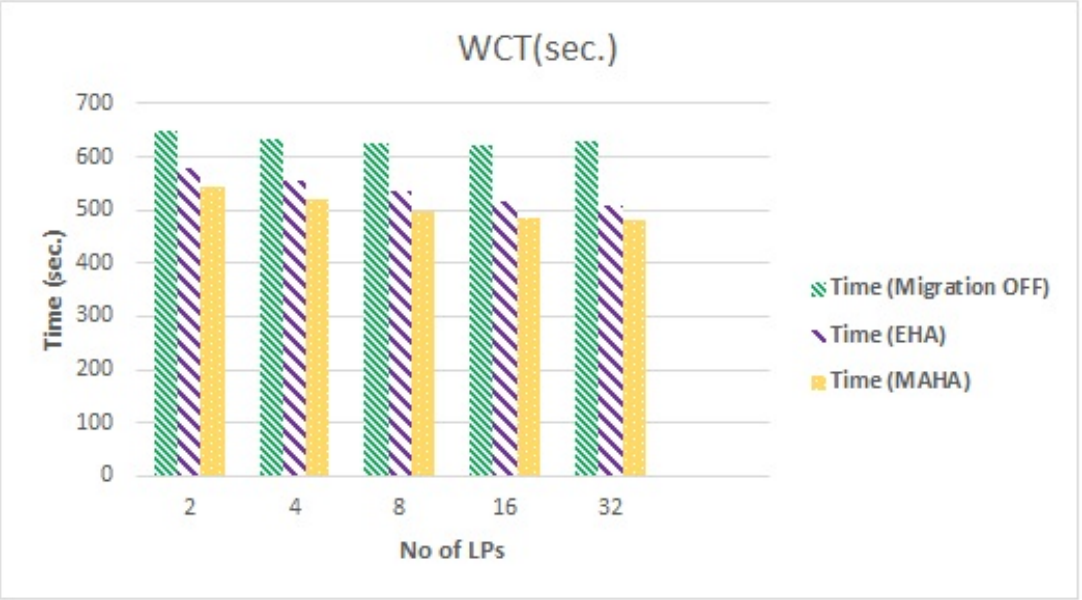

FIGURE 5.20: WCT with and without Migration for EHA and MAHA.

An increase of 12% and 19% in simulation events per second is observed for parallel simulation execution (i.e., 2 LPs) on EHA and MAHA respectively, as compared simulation execution without migration. A further increase of 16-24% in number of events per second for EHA and 26-31% MAHA is observed for 4 or more LPs. The obtained results reveal that employing MAHA for A-SIM-Cumulus instance resulted in improved performance in terms of number of simulated events per second. The overall simulation results ultimately supports the usage of MAHA for large-scale wireless networks.

In summary, the proposed migration algorithm is able to achieve speed up for all the tested configuration on the parallel setup. Even though the magnitude of gain is limited in some configurations but relevant. For distributed simulation in most of the tested configurations, the results are much better in terms of performance gain. However, in some cases, the results are dropped down to negative.

5.5 Chapter Summary

In this chapter, discussion related to the simulation design setup and obtained results is presented. Three set of simulation experiments are performed to evaluate

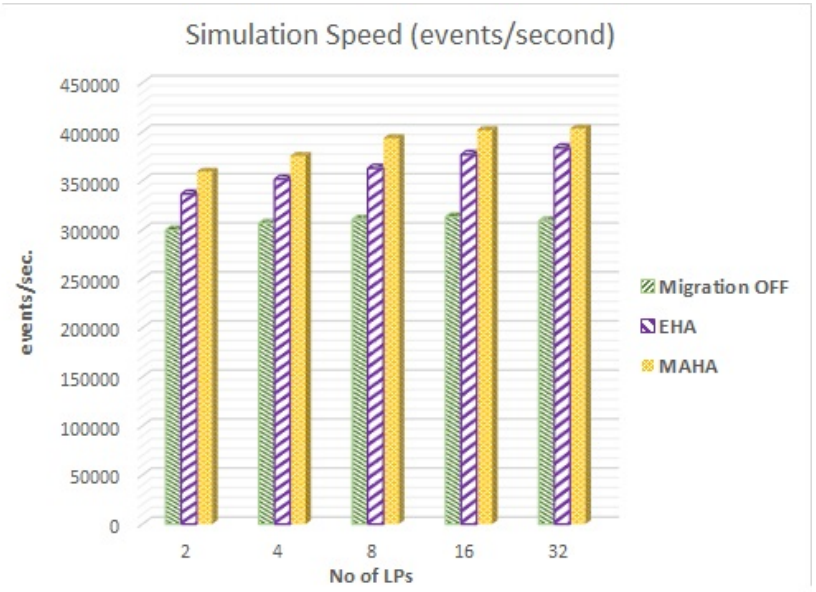

FIGURE 5.21: Simulation speed comparison of EHA and MAHA.

the proposed work. The simulations are performed multiple times sequentially as well as in parallel mode on all the machines (i.e., stand-alone workstations, MS Azure instance, and SIM-Cumulus instance). The results (i.e., simulation execution time, simulation speed, energy consumption, CPU utilization, and LRR) are examined for the proposed SIM-Cumulus and are compared with the workstations and MS Azure Cloud instance. Moreover, the results pertaining to the proposed MAHA algorithm are also obtained and presented in detail.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Network simulation approaches serve as a platform to evaluate the performance of network systems before its actual deployment. The simulation of complex and large-scale systems is a time-consuming process due to numerous factors, i.e., time taken to execute the simulation, the processing, memory requirements and implementation. Over the past few years, Cloud computing is deemed as a viable solution to solve large-scale computing problems related to both industry and academia. In academic institutions, the trend to adopt Cloud computing is gradually increasing to empower IT infrastructure and to assist the researchers. Moreover, the service-oriented paradigm of Cloud computing has lent itself as a suitable platform for customized solutions. In this thesis, an academic cloud SIM-Cumulus is proposed and implemented to provide NSaaS. The required simulation can be small to medium as well as large-scale complex systems. The proposed Cloud framework will assist the network researchers by providing customized simulation services with high computing power and large memory instances to perform large-scale network simulations. SIM-Cumulus provides the users with web-based API to access the Cloud resources and to perform the simulation as per their requirements. The users create their instances configured with their choice of simulator

and use secure RDP connection to work with their simulations. SIM-Cumulus provides two modules (i.e., SCM and SDM) to handle large-scale simulation using parallel execution on the respective instance. The SDM is responsible for the partitioning of simulation model (i.e., divide into several LPs) and distribution of SEs on specific LP. The simulation results of SIM-Cumulus have obtained remarkable effectiveness and efficiency in terms of simulation elapsed time, cost, and green IT effect.

The second contribution of this thesis pertains to the dynamic partitioning of large-scale PADS simulation. The large-scale simulation is partitioned into a number of components called LPs and each component is assigned to a separate execution unit wherein each LP contains a number of SEs. The distribution of SEs on different LPs may follow a different communication pattern. The SEs on the LP can communicate locally as well as to the remote SEs. The remote communication has a certain cost which leads to extra time for remote interaction dissemination. Thus, the parallel simulation may not achieve the required gain in the time cost reduction. The dynamic partitioning (i.e., SE migration) techniques are employed with the aim of remote communication reduction. In this work, we have presented a dynamic partitioning algorithm MAHA to cope with the high-end remote communication end. To support large-scale simulation in the Cloud, ARTIS/GAIA framework is integrated with SIM-Cumulus Cloud. The obtained results, for both the multi-core and distributed simulations, demonstrate that the proposed migration algorithm reduces the total migrations and achieves an optimum level of the LCR. In addition, it is able to achieve the speedup in terms of execution time on both the multi-core and distributed architectures using SIM-Cumulus instances.

Conclusively, the SIM-Cumulus and A-SIM-Cumulus help academia in the following ways:

- to reduce the time and effort required to configure simulation environment;
- to improve simulation performance in terms of simulation elapsed time;

- to exploit the under-utilized computing resources for research work in a more effective way;
- to lower the cost of IT infrastructure for educational establishments;
- to shift software licensing and maintenance responsibilities to Cloud providers;
- to reduce electricity consumption and emission of carbon footprints.

6.2 Future Work

This research work opens new directions in the area of migrations for large-scale PADS simulations. In the case of distributed PADS, all the LPs follow heterogeneous architecture in terms of hardware resources and simulation communication load. An LP may overload due to the computation required for handling huge communication (i.e., local as well as remote), which may result in LP crash, thus the failure of overall simulation execution. To resolve this issue, LP migration approach is required that could detect the LP load at run time and migrate it accordingly. In future work, VM migration approach is going to be proposed to handle the aforementioned issue.

Bibliography

[1] K. Wehrle, M. Günes, and J. Gross, *Modeling and tools for network simulation.* Springer Science & Business Media, 2010.

[2] T. Issariyakul and E. Hossain, *Introduction to Network Simulator NS-2*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[3] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," 2010, pp. 15–34.

[4] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, 2008.

[5] M. A. Iqbal, , M. Aleem, M. A. Islam, and M. Hassan, *Computer Network Simulation using OMNeT++*, 1st ed. Higher Education Commission, Pakistan, 2017.

[6] A. S. Sethi and V. Y. Hnatyshin, *The practical OPNET user guide for computer network simulation.* CRC Press, 2012.

[7] P. Latkoski, V. Rakovic, O. Ognenoski, V. Atanasovski, and L. Gavrilovska, "Sdl+qualnet: A novel simulation environment for wireless heterogeneous networks," in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, ser. SIMUTools '10, 2010, pp. 1–10.

[8] A. Lancin and D. Papadimitriou, "DRMSim: A routing-model simulator for large-scale networks," *ERCIM News*, vol. 94, pp. 31–32, 2013.

[9] R. Riebl, H.-J. Günther, C. Facchi, and L. Wolf, "Artery: Extending veins for vanet applications," in *Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, 2015, pp. 450–456.

[10] M. Won, T. Park, and S. H. Son, "Toward mitigating phantom jam using vehicle-to-vehicle communication," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1313–1324, 2017.

[11] JIST/SWANs, "Jist/swans simulator," http://jist.ece.cornell.edu/ [online] Accessed on, March 2018.

[12] R. Fernandes, P. M. d'Orey, and M. Ferreira, "Divert for realistic simulation of heterogeneous vehicular networks," in *IEEE 7th International Conference on Mobile Adhoc and Sensor Systems (MASS),*, 2010, pp. 721–726.

[13] R. Hussain, S. A. Malik, S. A. Khan, and S. Abrar, "Design, implementation and experimental evaluation of an end-to-end vertical handover scheme on nctuns simulator," *Simulation Modelling Practice and Theory*, vol. 26, pp. 151–167, 2012.

[14] V. Kumar, L. Lin, D. Krajzewicz, F. Hrizi, O. Martinez, J. Gozalvez, and R. Bauza, "itetris: Adaptation of its technologies for large scale integrated simulation," in *Vehicular Technology Conference (VTC 2010-Spring), 2010 IEEE 71st*, 2010, pp. 1–5.

[15] A. Boulis, "Castalia: revealing pitfalls in designing distributed algorithms in wsn," in *Proceedings of the 5th international conference on Embedded networked sensor systems.* ACM, 2007, pp. 407–408.

[16] X. KoutsouKos, G. Karsai, A. Laszka, H. Neema, B. Potteiger, P. Volgyesi, Y. Vorobeychik, and J. Sztipanovits, "Sure: A modeling and simulation integration platform for evaluation of secure and resilient cyber–physical systems," *Proceedings of the IEEE*, vol. 106, no. 1, pp. 93–112, 2018.

[17] "Working with network simulators," in *https://goo.gl/pnzYNW*, Accessed March 2018.

[18] J. Harri, F. Filali, and C. Bonnet, "Mobility models for vehicular ad hoc networks: a survey and taxonomy," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 4, 2009.

[19] D. Cavin, Y. Sasson, and A. Schiper, "On the accuracy of manet simulators," in *Proceedings of the second ACM international workshop on Principles of mobile computing*, 2002, pp. 38–43.

[20] L. F. Perrone, Y. Yuan, and D. M. Nicol, "Simulation of large scale networks ii: modeling and simulation best practices for wireless ad hoc networks," in *Proceedings of the 35th conference on Winter simulation: driving innovation*. Winter Simulation Conference, 2003, pp. 685–693.

[21] M. Mubarak, C. D. Carothers, R. B. Ross, and P. Carns, "Enabling parallel simulation of large-scale hpc network systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, pp. 87–100, 2017.

[22] R. Fujimoto, "Parallel and distributed simulation," in *Proceedings of the 2015 Winter Simulation Conference*. IEEE Press, 2015, pp. 45–59.

[23] G. D.Angelo, "The simulation model partitioning problem: An adaptive solution based on self-clustering," *Simulation Modelling Practice and Theory*, vol. 70, pp. 1–20, 2017.

[24] R. M. Fujimoto, *Parallel and distributed simulation systems*. Wiley New York, 2000, vol. 300.

[25] A. Boukerche and A. Fabbri, "Partitioning parallel simulation of wireless networks," in *Proceedings of the 32nd conference on Winter simulation*. Society for Computer Simulation International, 2000, pp. 1449–1457.

[26] G. D'Angelo, "Parallel and distributed simulation from many cores to the public cloud," in *High Performance Computing and Simulation (HPCS), 2011 International Conference on*. IEEE, 2011, pp. 14–23.

[27] G. D.Angelo and M. Marzolla, "New trends in parallel and distributed simulation: From many-cores to cloud computing," *Simulation Modelling Practice and Theory*, vol. 49, pp. 320–335, 2014.

[28] J. D. Rhodes, C. R. Upshaw, C. B. Harris, C. M. Meehan, D. A. Walling, P. A. Navrátil, A. L. Beck, K. Nagasawa, R. L. Fares, W. J. Cole *et al.*, "Experimental and data collection methods for a large-scale smart grid deployment: Methods and first results," *Energy*, vol. 65, pp. 462–471, 2014.

[29] D. Zehe, A. Knoll, W. Cai, and H. Aydt, "Semsim cloud service: Large-scale urban systems simulation in the cloud," *Simulation Modelling Practice and Theory*, vol. 58, pp. 157–171, 2015.

[30] S. Limet, A. Merlo, and L. Spalazzi, "Hpc & co strike back: Where are distributed paradigms heading toward?" *Concurrency and Computation: Practice and Experience*, p. e4431, 2018.

[31] J. W. Rittinghouse and J. F. Ransome, *Cloud computing: implementation, management, and security*. CRC press, 2016.

[32] J. D. Rhodes, C. R. Upshaw, C. B. Harris, C. M. Meehan, D. A. Walling, P. A. Navrátil, A. L. Beck, K. Nagasawa, R. L. Fares, W. J. Cole *et al.*, "Experimental and data collection methods for a large-scale smart grid deployment: Methods and first results," *Energy*, vol. 65, pp. 462–471, 2014.

[33] P. Janovsky and S. A. DeLoach, "Multi-agent simulation framework for large-scale coalition formation," in *IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, 2016, pp. 343–350.

[34] S. Coakley, P. Richmond, M. Gheorghe, S. Chin, D. Worth, M. Holcombe, and C. Greenough, "Large-scale simulations with flame," in *Intelligent Agents in Data-intensive Computing*, 2016, pp. 123–142.

[35] R. Paranjape, Z. G. Wang, and S. Gill, "Agent-based modeling and simulation," in *The Diabetic Patient Agent*. Springer, 2018, pp. 9–14.

[36] A. L. Bazzan, "Multiagent systems and agent-based modeling and simulation," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 2017, pp. 959–1004.

[37] V. L. Minden, C. C. Youn, and U. A. Khan, "A distributed self-clustering algorithm for autonomous multi-agent systems," in *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*. IEEE, 2012, pp. 1445–1448.

[38] X. Liu, X. Qiu, B. Chen, and K. Huang, "Cloud-based simulation: The state-of-the-art computer simulation paradigm," in *Proceedings of the 2012 ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation*. IEEE Computer Society, 2012, pp. 71–74.

[39] A. Malik, A. Park, and R. Fujimoto, "Optimistic synchronization of parallel simulations in cloud computing environments," in *Cloud Computing, 2009. CLOUD'09. IEEE International Conference on*. IEEE, 2009, pp. 49–56.

[40] X. Liu, Q. He, X. Qiu, B. Chen, and K. Huang, "Cloud-based computer simulation: Towards planting existing simulation software into the cloud," *Simulation Modelling Practice and Theory*, vol. 26, pp. 135–150, 2012.

[41] S. B. Yoginath and K. S. Perumalla, "Efficient parallel discrete event simulation on cloud/virtual machine platforms," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 26, no. 1, p. 5, 2015.

[42] W. W. Smari, M. Bakhouya, S. Fiore, and G. Aloisio, "New advances in high performance computing and simulation: parallel and distributed systems, algorithms, and applications," *Concurrency and Computation: Practice and Experience*, vol. 28, pp. 2024–2030, 2016.

[43] C. M. Macal and M. J. North, "Agent-based modeling and simulation: Desktop abms," in *Simulation Conference, 2007 Winter*. IEEE, 2007, pp. 95–106.

[44] M. A. Niazi, "Towards a novel unified framework for developing formal, network and validated agent-based simulation models of complex adaptive systems," *arXiv preprint arXiv:1708.02357*, 2017.

[45] M. Niazi and A. Hussain, "Agent-based tools for modeling and simulation of self-organization in peer-to-peer, ad hoc, and other complex networks," *IEEE Communications Magazine*, vol. 47, no. 3, 2009.

[46] R. Stanica, E. Chaput, and A.-L. Beylot, "Simulation of vehicular ad-hoc networks: Challenges, review of tools and recommendations," *Computer Networks*, vol. 55, no. 14, pp. 3179–3188, 2011.

[47] J. Eze, S. Zhang, E. Liu, and E. Eze, "Cognitive radio technology assisted vehicular ad-hoc networks (vanets): Current status, challenges, and research trends," in *Automation and Computing (ICAC), 2017 23rd International Conference on*. IEEE, 2017, pp. 1–6.

[48] R. Fernandes and M. Ferreira, "Scalable vanet simulations with ns-3," in *IEEE 75th Vehicular Technology Conference, (VTC Spring)*, May 2012, pp. 1–5.

[49] J. D. Walker and S. C. Chapra, "A client-side web application for interactive environmental simulation modeling," *Environmental Modelling & Software*, vol. 55, pp. 49–60, 2014.

[50] P. Lorenz, T. J. Schriber, H. Dorwarth, and K.-C. Ritter, "Towards a web based simulation environment," in *Proceedings of the 29th conference on Winter simulation*, 1997, pp. 1338–1344.

[51] S. J. Taylor, A. Khan, K. L. Morse, A. Tolk, L. Yilmaz, and J. Zander, "Grand challenges on the theory of modeling and simulation," in *Proceedings of the Symposium on Theory of Modeling & Simulation-DEVS Integrative M&S Symposium*, 2013, p. 34.

[52] Y. Qu and X. Zhou, "Large-scale dynamic transportation network simulation: A space-time-event parallel computing approach," *Transportation Research Part C: Emerging Technologies*, vol. 75, pp. 1–16, 2017.

[53] H. Liu, K. Wang, Z. Chen, B. Yang, and R. He, "Large-scale reservoir simulations on distributed-memory parallel computers," in *Proceedings of the 24th High Performance Computing Symposium*, 2016.

[54] D. Puthal, B. Sahoo, S. Mishra, and S. Swain, "Cloud computing features, issues, and challenges: a big picture," in *Computational Intelligence and Networks (CINE), 2015 International Conference on.* IEEE, 2015, pp. 116–123.

[55] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, "Cloud computingthe business perspective," *Decision support systems*, vol. 51, no. 1, pp. 176–189, 2011.

[56] H. Erdogmus, "Cloud computing: does nirvana hide behind the nebula?" *Software, IEEE*, vol. 26, no. 2, pp. 4–6, 2009.

[57] M. Nkosi and F. Mekuria, "Cloud computing for enhanced mobile health applications," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on.* IEEE, 2010, pp. 629–633.

[58] S. Guo, F. Bai, and X. Hu, "Simulation software as a service and service-oriented simulation experiment," in *Information Reuse and Integration (IRI), 2011 IEEE International Conference on*, 2011, pp. 113–116.

[59] E. Cayirci, "Modeling and simulation as a cloud service: a survey," in *Simulation Conference (WSC)*, 2013, pp. 389–400.

[60] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, pp. 23–50, 2011.

[61] A. W. Malik, K. Bilal, S. Malik, Z. Anwar, K. Aziz, D. Kliazovich, N. Ghani, S. U. Khan, and R. Buyya, "Cloudnetsim++: A gui based framework for modeling and simulation of data centers in omnet++," *IEEE Transactions on Services Computing*, vol. 10, no. 4, pp. 506–519, 2017.

[62] B. B. Romdhanne and N. Nikaein, "General-purpose coordinator–master–worker model for efficient large-scale simulation over heterogeneous infrastructure," *Journal of Simulation*, vol. 11, no. 3, pp. 228–241, 2017.

[63] U. ur Rahman, O. Hakeem, M. Raheem, K. Bilal, S. U. Khan, and L. T. Yang, "Nutshell: Cloud simulation and current trends," in *IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, 2015, pp. 77–86.

[64] M. Ficco, B. Di Martino, R. Pietrantuono, and S. Russo, "Optimized task allocation on private cloud for hybrid simulation of large-scale critical systems," *Future Generation Computer Systems*, vol. 74, pp. 104–118, 2017.

[65] S. J. Taylor, T. Kiss, G. Terstyanszky, P. Kacsuk, and N. Fantini, "Cloud computing for simulation in manufacturing and engineering: introducing the cloudsme simulation platform," in *Proceedings of the 2014 Annual Simulation Symposium*, 2014.

[66] B. Dong, Q. Zheng, M. Qiao, J. Shu, and J. Yang, "Bluesky cloud framework: an e-learning framework embracing cloud computing," in *Cloud Computing*. Springer, 2009, pp. 577–582.

[67] K. A. Delic and M. A. Walker, "Emergence of the academic computing clouds," *ACM Ubiquity*, Aug. 2008.

[68] A. Barker, B. Varghese, J. S. Ward, and I. Sommerville, "Academic cloud computing research: Five pitfalls and five opportunities," in *6th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 14)*, 2014.

[69] J. A. González-Martínez, M. L. Bote-Lorenzo, E. Gómez-Sánchez, and R. Cano-Parra, "Cloud computing and education: A state-of-the-art survey," *Computers & Education*, vol. 80, pp. 132–151, 2015.

[70] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: Towards a cloud definition," *SIGCOMM Computuer Communication Review*, vol. 39, no. 1, pp. 50–55, Dec. 2008.

[71] P. T. Endo, G. E. Gonçalves, J. Kelner, and D. Sadok, "A survey on open-source cloud computing solutions," in *Brazilian Symposium on Computer Networks and Distributed Systems*, vol. 71, 2010.

[72] J. A. González-Martínez, M. L. Bote-Lorenzo, E. Gómez-Sánchez, and R. Cano-Parra, "Cloud computing and education: A state-of-the-art survey," *Computers & Education*, vol. 80, pp. 132–151, 2015.

[73] D. Milojicic, I. M. Llorente, and R. S. Montero, "Opennebula: A cloud management tool," *IEEE Internet Computing*, vol. 15, no. 2, pp. 11–14, March 2011.

[74] K. Keahey, R. Figueiredo, J. A. B. Fortes, T. Freeman, and M. Tsugawa, "Cloud computing with nimbus," in *Cloud Computing for Systems and Computational Biology Workshop*, 2009.

[75] I. Foster, "Globus toolkit version 4: Software for service-oriented systems," *Journal of computer Science and Technology*, vol. 21, no. 4, pp. 513–520, July 2006.

[76] K. Jackson, *OpenStack Cloud Computing Cookbook*. Packt Publishing, 2012.

[77] P. T. Endo, G. E. Gonçalves, J. Kelner, and D. Sadok, "A survey on open-source cloud computing solutions," in *Brazilian Symposium on Computer Networks and Distributed Systems*, 2010, pp. 3–16.

[78] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system,"

in *IEEE/ACM 9th International Symposium on Cluster Computing and the Grid, CCGRID '09*, May 2009, pp. 124–131.

[79] B. K. Szymanski, A. Saifee, A. Sastry, Y. Liu, and K. Madnani, "Genesis: a system for large-scale parallel network simulation," in *Proceedings of the sixteenth workshop on Parallel and distributed simulation*. IEEE Computer Society, 2002, pp. 89–96.

[80] H. W. Group *et al.*, "Ieee standard for modeling and simulation (m&s) high level architecture (hla)-framework and rules," *IEEE Standard*, pp. 1516–2000, 2000.

[81] C. Raczy, G. Tan, and J. Yu, "A sort-based ddm matching algorithm for hla," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 15, no. 1, pp. 14–38, 2005.

[82] B. İ. Kumova, "Dynamically adaptive partition-based data distribution management," in *Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*. IEEE Computer Society, 2005, pp. 292–300.

[83] W. Cai, S. J. Turner, and B. P. Gan, "Hierarchical federations: an architecture for information hiding," in *Parallel and Distributed Simulation, 2001. Proceedings. 15th Workship on*. IEEE, 2001, pp. 67–74.

[84] A. Boukerche and C. Tropper, "A static partitioning and mapping algorithm for conservative parallel simulations," in *ACM SIGSIM Simulation Digest*, vol. 24, no. 1. ACM, 1994, pp. 164–172.

[85] A. Boukerche and S. K. Das, "Dynamic load balancing strategies for conservative parallel simulations," in *Parallel and Distributed Simulation, 1997., Proceedings., 11th Workshop on*. IEEE, 1997, pp. 20–28.

[86] K. Yocum, E. Eade, J. Degesys, D. Becker, J. Chase, and A. Vahdat, "Toward scaling network emulation using topology partitioning," in *Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003.*

MASCOTS 2003. 11th IEEE/ACM International Symposium on. IEEE, 2003, pp. 242–245.

[87] G. Vigueras, M. Lozano, J. M. Orduña, and F. Grimaldo, "A comparative study of partitioning methods for crowd simulations," *Applied Soft Computing*, vol. 10, no. 1, pp. 225–235, 2010.

[88] G. D'Angelo, S. Ferretti, and V. Ghini, "Distributed hybrid simulation of the internet of things and smart territories," *Concurrency and Computation: Practice and Experience*, 2017.

[89] ——, "Modeling the internet of things: a simulation perspective," in *High Performance Computing & Simulation (HPCS), 2017 International Conference on.* IEEE, 2017, pp. 18–27.

[90] S. Ferretti, G. DAngelo, V. Ghini, and M. Marzolla, "The quest for scalability and accuracy: Multi-level simulation of the internet of things," *arXiv preprint arXiv:1710.02282*, 2017.

[91] B. Logan and G. Theodoropoulos, "The distributed simulation of multiagent systems," *Proceedings of the IEEE*, vol. 89, no. 2, pp. 174–185, 2001.

[92] R. E. De Grande and A. Boukerche, "Dynamic balancing of communication and computation load for hla-based simulations on large-scale distributed systems," *Journal of Parallel and Distributed Computing*, vol. 71, no. 1, pp. 40–52, 2011.

[93] P. Peschlow, T. Honecker, and P. Martini, "A flexible dynamic partitioning algorithm for optimistic distributed simulation," in *Proceedings of the 21st International Workshop on Principles of Advanced and Distributed Simulation.* IEEE Computer Society, 2007, pp. 219–228.

[94] G. D'Angelo and M. Bracuto, "Distributed simulation of large-scale and detailed models," *International Journal of Simulation and Process Modelling*, vol. 5, no. 2, pp. 120–131, 2009.

[95] G. DAngelo, "Artis: Design and implementation of an adaptive middleware for parallel and distributed simulation," in *Technical Report*, 2005.

[96] L. Bononi, M. Bracuto, G. D'Angelo, and L. Donatiello, "Performance analysis of a parallel and distributed simulation framework for large scale wireless systems," in *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems.* ACM, 2004, pp. 52–61.

[97] G. D'Angelo and S. Ferretti, "Simulation of scale-free networks," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques.* ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, p. 20.

[98] G. DAngelo, S. Ferretti, M. Marzolla, and L. Armaroli, "Fault-tolerant adaptive parallel and distributed simulation," in *Distributed Simulation and Real Time Applications (DS-RT), 2016 IEEE/ACM 20th International Symposium on.* IEEE, 2016, pp. 37–44.

[99] M. A. Iqbal, M. Aleem, M. A. Islam, M. Ibrahim, and S. Anwar, "Amazon cloud computing platform EC2 and VANET Simulations," *Int. Journal of Ad Hoc and Ubiquitous Computing, In Press (2018)*, 2018.

[100] I. Eucalyptus Systems, "Eucalyptus community cloud," http://open.eucalyptus.com/try/community-cloud [online] Accessed on, March 2018.

[101] JavaSysmon, "Javasysmon," [online] Available on https://github.com/jezhumble/javasysmon/wiki, Accessed September 2017.

[102] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on.* IEEE, 2009, pp. 124–131.

[103] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and linux containers," in *Performance Analysis of Systems and Software (ISPASS)*, 2015, pp. 171–172.

[104] G. A. Wainer and P. J. Mosterman, *Discrete-event modeling and simulation: theory and applications*. CRC Press, 2016.

[105] J. Zhao, J. Tao, and K. Fürlinger, "A framework for comparative performance study on virtualised machines," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 17, no. 2/3, pp. 82–99, Nov. 2014.

[106] A. Mahajan, N. Potnis, K. Gopalan, and A. Wang, "Modeling vanet deployment in urban settings," in *Proceedings of the 10th ACM Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, ser. MSWiM '07, 2007, pp. 151–158.

[107] S.-Y. Jing, S. Ali, K. She, and Y. Zhong, "State-of-the-art research study for green cloud computing," *The Journal of Supercomputing*, vol. 65, no. 1, pp. 445–468, 2013.

[108] S. Ruth, "Green it more than a three percent solution?" *IEEE Internet Computing*, vol. 13, no. 4, pp. 74–78, July 2009.

[109] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. Bhattacharya, "Virtual machine power metering and provisioning," in *Proceedings of the 1st ACM symposium on Cloud computing*. ACM, June 2010, pp. 39–50.

[110] S. Cloudin and P. M. Kumar, "Challenges on mobility models suitable to vanet," *Journal of Software*, vol. 12, pp. 91–101, 2017.

[111] S. Kurt and B. Tavli, "Path-loss modeling for wireless sensor networks: A review of models and comparative evaluations." *IEEE Antennas and Propagation Magazine*, vol. 59, pp. 18–37, 2017.

[112] P. Xie, J.-H. Cui, and L. Lao, "Vbf: vector-based forwarding protocol for underwater sensor networks," in *Networking*, vol. 3976, 2006, pp. 1216–1221.

[113] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin, "Wireless sensor networks: a survey on recent developments and potential synergies," *The Journal of supercomputing*, vol. 68, pp. 1–48, 2014.

[114] A. Virdis, C. Vallati, and G. Nardini, "Automating large-scale simulation and data analysis with omnet++: Lession learned and future perspectives," *arXiv preprint arXiv:1609.04603*, 2016.

[115] M. AZURE, "Ms azure," [online] Available on https://azure.microsoft.com/en-us/, Accessed September 2017.

[116] U. Wajid, C. Cappiello, P. Plebani, B. Pernici, N. Mehandjiev, M. Vitali, M. Gienger, K. Kavoussanakis, D. Margery, D. G. Perez *et al.*, "On achieving energy efficiency and reducing co 2 footprint in cloud computing," *IEEE transactions on cloud computing*, vol. 4, pp. 138–151, 2016.

[117] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong, "Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing," *Journal of Network and Computer Applications*, vol. 59, pp. 46–54, 2016.

[118] A. Varga and A. Sekercioglu, "Parallel simulation made easy with omnet++," in *Proc. 15th European Simulation Symposium and Exhibition*, 2003.

[119] S. Chong, D. Wencai, A. Robert, I. James, and P. Dirk, "A mobility framework to improve heterogeneous wireless network services," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 7, no. 1, pp. 60–69, 2011.