IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 22, 2018, accepted April 25, 2018, date of publication May 4, 2018, date of current version June 5, 2018.

*Digital Object Identifier 10.1109/ACCESS.2018.2833212*

# SIM-Cumulus: An Academic Cloud for the Provisioning of Network-Simulation-as-a-Service (NSaaS)

**MUHAMMAD IBRAHIM, MUHAMMAD AZHAR IQBAL, MUHAMMAD ALEEM, AND MUHAMMAD ARSHAD ISLAM, (Member, IEEE)**
Department of Computer Science, Capital University of Science and Technology, Islamabad 44000, Pakistan
Corresponding author: Muhammad Ibrahim (ibrahimislamian@gmail.com)

**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 involved or in abridging expected micro-level details. The Cloud technology facilitates researchers to address mentioned factors by the provisioning of pre-configured instances on shared infrastructure. In this paper, an academic Cloud architecture SIM-Cumulus targeting the research institutions is proposed. SIM-Cumulus provides the framework of virtual machine 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. Simulation results show that SIM-Cumulus is beneficial in three aspects i.e., 1) promotion of research within the domain of computer networks; 2) consumption of considerably fewer resources in terms of simulation elapsed time and usage cost; and 3) reduction of carbon emission leading toward sustainable IT development.

**INDEX TERMS** SIM-Cumulus, academic Cloud, network simulation, OMNeT++, wireless networks.

## I. INTRODUCTION

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) and analytical methods are unable to fully grasp the characteristics associated with communications in wireless networks. A number of simulation tools (i.e., 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] etc.) are being employed by computer network researchers to verify the working of designed protocols for both 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 respective programming language, type of communication system, 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* [15]). 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 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.

Configuration complexity at early stage hinders the selection and usage of network simulators. However, scalability problem arises at later stages when the researchers have gained substantial expertise of a simulator and are compelled to reduce the scope of simulations due to the scarcity of available hardware resources. Moreover, simulation scenarios involving a large number of nodes (with high inter-node communication) poses a scalability challenge for the execution over multi-cores and Cloud. In this work, we refer large-scale simulation scenario demanding large computational and memory resources that are usually not available on a single machine. To address these challenges, this work proposes

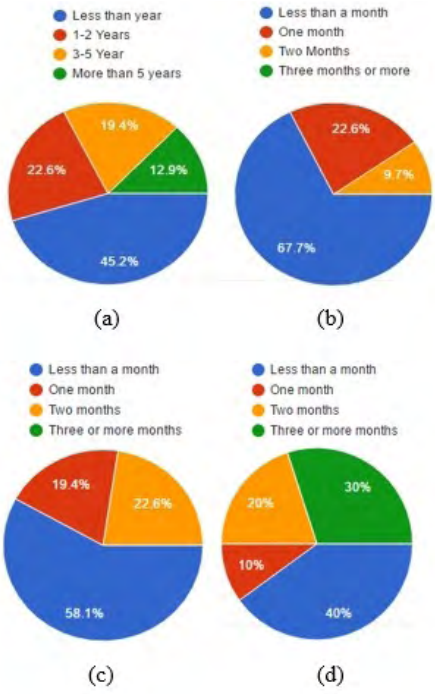


**FIGURE 1.** (a) Researchers' experience with network simulators, (b) Time taken to configure network simulator, (c) Time taken to execute basic example, (d) Time taken to execute simulation with available frameworks/patches.

an Academic Cloud Framework SIM-Cumulus that provides scalable network simulation service. SIM-Cumulus makes use of configured Cloud instances using web interfaces. The fundamental goals of the SIM-Cumulus framework are three folds: 1) addressing the issues of simulator configurations faced by the naive users, 2) the provisioning of Network-Simulation-as-a-Service (NSaaS), and 3) automation of appropriate VM instance creation for large-scale network simulation and parallel simulation execution (to manage scalable network simulations). SIM-Cumulus harnesses the power of Cloud environment while taking into account the aspects related to IT sustainability and green computing. We have performed multiple simulations to analyze the potential of SIM-Cumulus that assists network researchers in terms of high performance computing, infrastructure cost, and power consumption, which ultimately pave the way towards sustainable green computing.

The rest of the paper is structured as follows: Section II provides related work discussing the generic structure of popular academic Clouds. Section III presents SIM-Cumulus architecture overview, working details, and mathematical model that target the scalability problems of network simulations. Section IV presents the experimental setup details for large-scale wireless network simulations. In Section V, the performance of SIM-Cumulus is investigated through large wireless network simulations using two network simulators (i.e., OMNeT++ (GUI-based) and ARTIS/GAIA (non-GUI-based)). The conclusion of the study is presented in Section VI along with the future perspectives of our work.

## II. RELATED WORK

Recent innovations in internet-based systems have attracted network researchers towards the adoption of web-based simulation environments [16]. The notion of web-based simulation design and modeling is initiated by Lorenz *et al.* in [17] where authors have presented prototypes of web-based simulations. Taylor *et al.* [18] have discussed research efforts related to the exploitation of web infrastructure for online simulation modeling, design, and result visualization. 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 [19]. A number of schemes i.e., Parallel processing [20], [21], Grid computing [22], Agent-based solution [23], [24], Distributed computing, High Performance Computing (HPC) [19], and Cloud computing are being employed by the researchers to attain the demands of large-scale simulation. Parallel processing [20], [21] is used to perform large-scale simulations via running multiple copies of the simulator (each representing different portion of the simulation) in parallel. Grid computing is designed for large-scale distributed data processing and storage, and works with assigning subtasks across different clusters [22], [25].

Agent-based simulation approaches [23], [24] are suitable to effectively deal with situations where each agent has been independently assigned different set of tasks. 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 [19].

Angelo and Marzolla [26] have favored the idea of Cloud computing paradigm for dynamic sizing of the large-scale network simulations. Guo *et al.* [27] proposed a five layered framework for systematic support of Simulation Software as a Service (SIMSaaS) in Cloud. According to Cayirci [28], 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 [29] and CloudnetSim++ [30] have enabled researchers and professionals to perform modeling of data centers and virtual machines, while managing the energy efficiency in Cloud. Rahman *et al.* [31] 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.* [32] have devised a method to cope with simulation of large-scale critical systems on private Cloud. Angelo [33] proposed an adaptive solution for large-scale simulations using parallel and distributed simulation (PADS) approach. CloudSME [34] was proposed as a Cloud-based simulation framework for large-scale simulation of manufacturing and engineering industry.

In particular, academic Cloud systems provide several compatible APIs to facilitate Cloud engineers in integrating different functionalities of various Clouds to work together

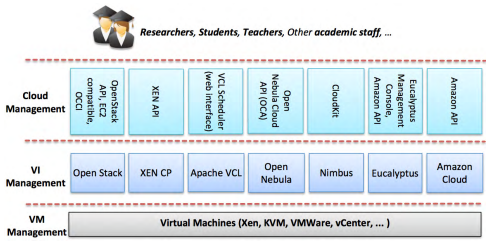


**FIGURE 2.** Structure of popular academic Clouds.

as shown in Figure 2. Such Clouds could be based on virtual machine (VM) management, Virtual Instance (VI) management, and Cloud management layers.

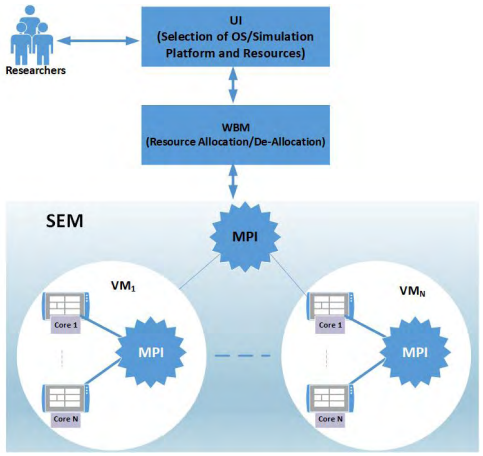


**FIGURE 3.** Generic Cloud architecture for network simulations.

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 3. 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) [35] can be used for interaction between simulation entities on different cores or on different VMs. Taking into account these design considerations of general academic Cloud architecture, we have implemented SIM-Cumulus, which is presented in the following section.

## III. 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.

### A. 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. The details of all necessary SIM-Cumulus components required to perform network simulations are presented below.

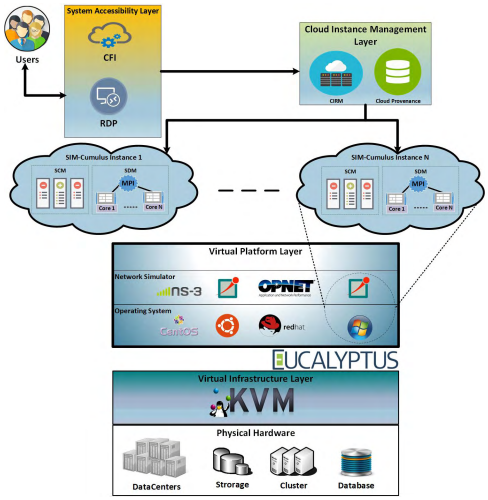


**FIGURE 4.** SIM-Cumulus proposed architecture.

#### 1) 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.

#### 2) 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 [36]. 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 [37] to provide instance-level Cloud provenance. In addition, a third-party library JavaSysmon [38] is utilized to obtain system-level Cloud provenance.

#### 3) 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 Logical Processes (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.

**TABLE 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 |

To elaborate the SDM decision making process, a number of simulations are performed using different problem sizes (number of nodes). Table 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 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 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.

#### 4) 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 [37] to implement the SIM-Cumulus Cloud architecture. The SIM-Cumulus exploits the benefits of the Eucalyptus to provide Network-Simulation-as-a-Service to the researchers. The Eucalyptus provides feasibility for private and hybrid

Cloud implementation [39]. 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 [40]. 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.

#### 5) 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). These resources are the actual hardware components used in the simulations and users are assigned to them in a transparent manner.

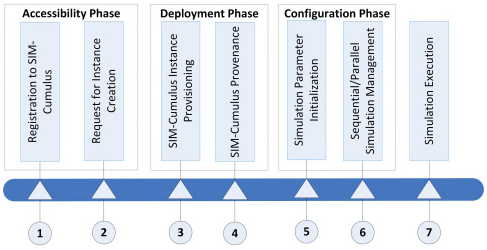


**FIGURE 5.** SIM-Cumulus workflow.

### B. 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 5. 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 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.

### C. 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) [41] 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 [20]. 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 [35] 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 another LP. Two types of message flows are illustrated in Figure 6. 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, Simulation Entity $SE_0$ 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.

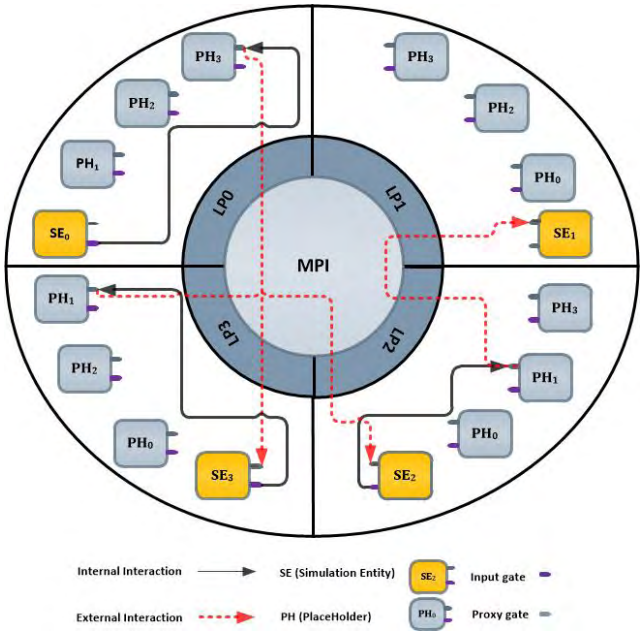


**FIGURE 6.** Remote communication among simulation entities (SEs).

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

Let $S_{LP}$ represents the set of LPs:

$$S_{LP} = \{LP_1, LP_2, LP_3, \ldots, LP_m\} \quad (1)$$

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

$$S_{SE} = \{SE_1, SE_2, SE_3, \ldots, SE_n\} \quad (2)$$

To distribute SEs across available LPs, it is required to find out the number of SEs (N) on each LP i.e.,

$$N = n/m \quad (3)$$

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:

$$LP_i = \{(i-1)(N)+1, (i-1)(N)+2 \ldots, (i-1)(N)+N\} \quad (4)$$

In Eq.4, $i$ represents index in $S_{LP}$. To execute simulation in parallel, SEs are required to be distributed across different logical processes (LPs). The allocation of specific range of PHs on different LPs is obtained using Eq.5.

$$LP_{PH} = S_{SE} \setminus LP_i = \{SE : SE \in S_{SE} \; and \; SE \notin LP_i\} \quad (5)$$

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**

### D. 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):

$$OEC = SUC + LIC \quad (6)$$

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.7:

$$OEC = SUC + GCC \quad (7)$$

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

$$OEC = SUC + (IC + SynC + MM) \quad (8)$$

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. 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.

$$IC = LIC + RIC \quad (9)$$

Thus overall execution cost can be calculated as given in Eq.10.

$$OEC = SUC + (LIC + RIC + SynC + MM) \quad (10)$$

The OEC depends on the ratio between local and remote communication. If LComm represents the size of local communication and RComm represents size of remote communication then LRR (i.e., Local to Remote Communication Ratio) can be obtained by the given formula (Eq.11).

$$LR_R = L_{Comm} \mid R_{Comm} \quad (11)$$

## IV. SIMULATION SETUP

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 [33]). 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 larger 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 simulation entities 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 [42] for sensor movement in underwater environment. Moreover, path loss propagation model [43] is employed to simulate the physical layer characteristics of the UWSNs. The opted routing protocol for UWSNs is Vector-Based Forwarding (VBF) [44]. 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 [45]. Table 2 summarizes all the parameters used in the OMNeT++ simulations.

**TABLE 2. 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 [44] |
| 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 [46]. 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 [47] and two workstations. Description of available instances and available workstations is depicted in Table 3.

**TABLE 3. 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 |

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 can send an interaction to 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 $\times$ 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.

## V. RESULTS AND DISCUSSION

### A. SIMULATION ON SIM-CUMULUS USING OMNET++

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 2. The experimental results regarding simulation execution time are presented in Figure 7.

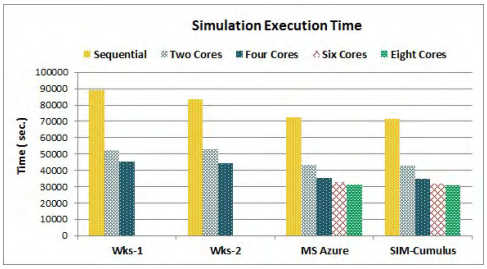


**FIGURE 7. Simulation execution time.**

Results in Figure 7 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 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

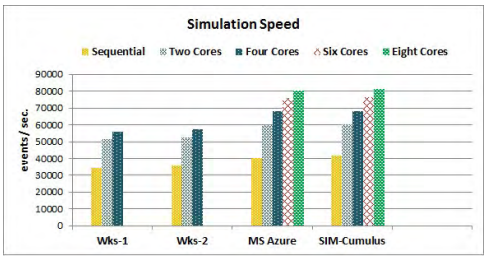


**FIGURE 8.** Simulation speed (events/second).

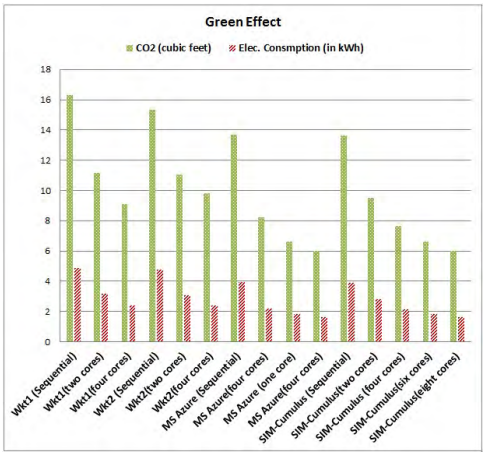


**FIGURE 9.** Simulation speed (events/second).

executions in terms of simulation speed is shown in Figure 8. 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 [47] 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 [48], [49]. Cloud computing is considered the appropriate choice for achieving green computing [50]. In this study, we have used Joulemeter [51] 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 9 presents $CO_2$ emission and energy-consumption of different employed multi-core 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

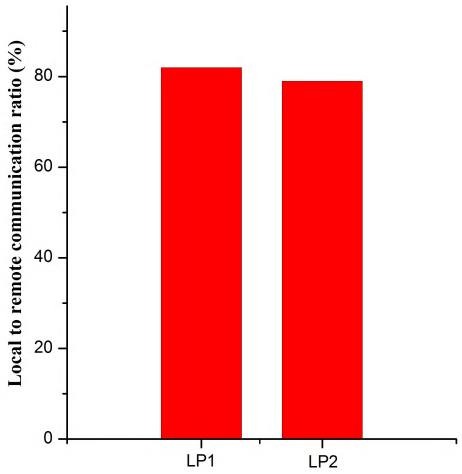


**FIGURE 10.** LRR with 2 LPs.

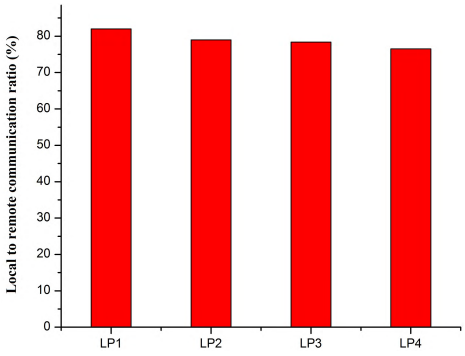


**FIGURE 11.** LRR with 4 LPs.

increase in number of LPs, look-ahead delay [52] 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 the share of costly remote communication involved in the simulation. To provide an insight into the performance gain, we obtained results regarding the local to remote communication ratio (LRR). LRR is the ratio of local communication of SEs to remote communication for the LPs. Figure 10 to Figure 13 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 10 and Figure 11 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 [52], required for the transmission of packets among SEs located on different LPs.

The results shown in Figure 13 demonstrate imbalanced communication pattern between available LPs. The results

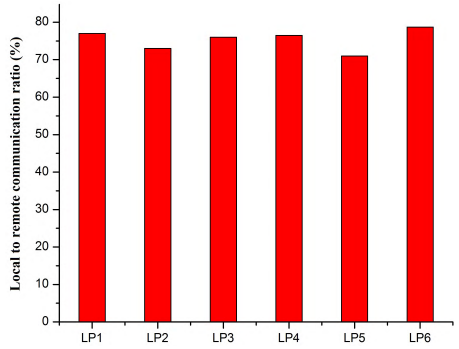


**FIGURE 12.** LRR with 6 LPs.

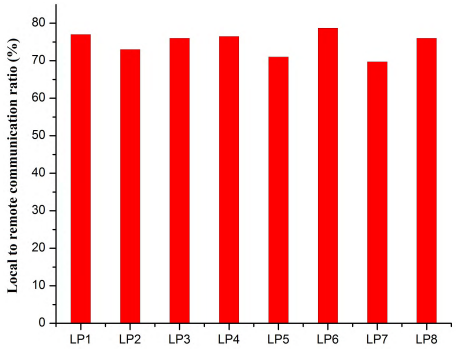


**FIGURE 13.** LRR with 8 LPs.

on the $LP_1$, $LP_4$, and $LP_6$ show average LRR 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.

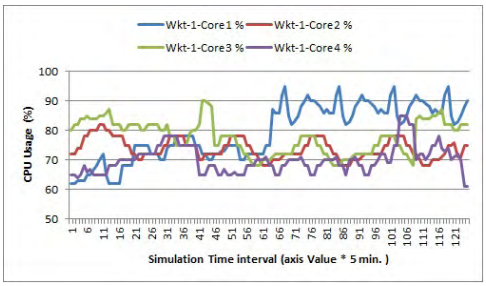


**FIGURE 14.** CPU usage on Wks-1.

Figure 14 and Figure 15 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 14. For the proposed academic cloud SIM-Cumulus, the CPU utilization observed is on average 72.4-85.5% as shown in Figure 15. 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

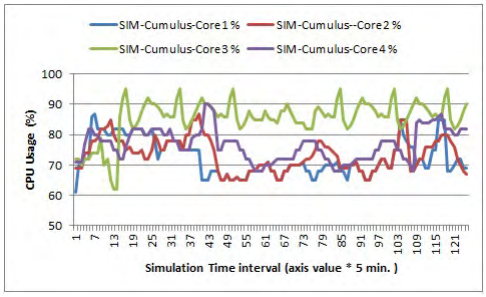


**FIGURE 15.** CPU usage on SIM-Cumulus.

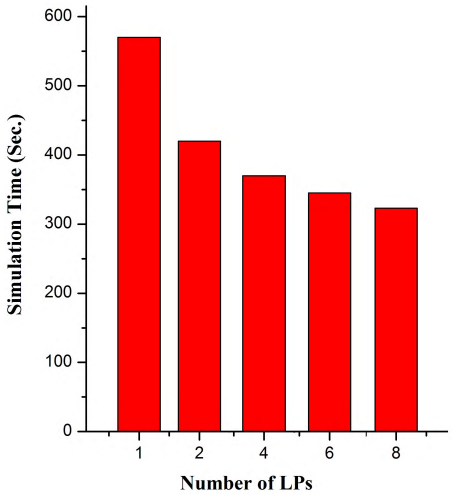


**FIGURE 16.** Simulation execution time.

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

### *B. SIMULATION ON SIM-CUMULUS USING ARTIS/GAIA*

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 16. 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 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.

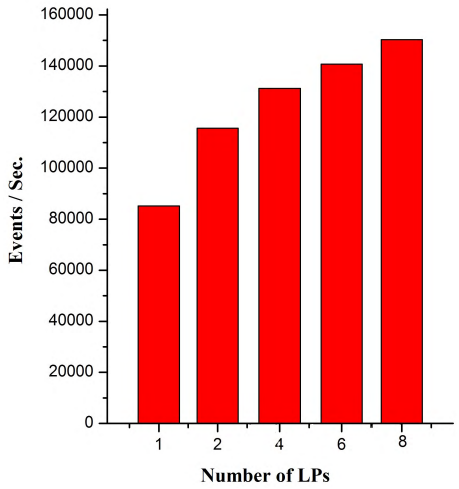


**FIGURE 17.** Simulation speed.

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 17. 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.

## VI. CONCLUSIONS AND FUTURE WORK

Over the past few years, Cloud computing is deemed as a viable solution for solving 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 assist researchers. The aim of the proposed work is to assist network researchers by provisioning high computing power and large memory instances to perform large-scale network simulations. The simulation results have depicted remarkable effectiveness and efficiency of SIM-Cumulus in terms of simulation elapsed time, cost, and green IT effect. In future, we intend to extend SIM-Cumulus capabilities through distributed approach for the simulation of large-scale wireless networks. In addition, adaptive approach based on the migration of simulation entities will further improve the performance of large-scale network simulations. Conclusively, the SIM-Cumulus can 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.

## REFERENCES

[1] K. Wehrle, M. Günes, and J. Gross, *Modeling and Tools for Network Simulation*. Berlin, Germany: Springer-Verlag, 2010.
[2] T. Issariyakul and E. Hossain, *Introduction to Network Simulator NS2*. New York, NY, USA: Springer, 2011.
[3] G. F. Riley and T. R. Henderson, "The $ns-3$ network simulator," in *Modeling and Tools for Network Simulation*. Berlin, Germany: Springer-Verlag, 2010, pp. 15–34.
[4] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Proc. 1st Int. Conf. Simulation Tools Techn. Commun., Netw. Syst. Workshops*, 2008, Art. no. 60.
[5] M. A. Iqbal, M. Aleem, M. A. Islam, and M. Hassan, *Computer Network Simulation Using OMNeT++*. Islamabad, Pakistan: Higher Education Commission of Pakistan, 2017.
[6] A. S. Sethi and V. Y. Hnatyshin, *The Practical OPNET User Guide for Computer Network Simulation*. Boca Raton, FL, USA: 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 *Proc. 3rd Int. ICST Conf. Simulation Tools Techn. (SIMUTools)*, 2010, Art. no. 25.
[8] A. Lancin and D. Papadimitriou, "DRMSim: A routing-model simulator for large-scale networks," *ERCIM News*, vol. 94, pp. 31–32, Jul. 2013.
[9] R. Riebl, H.-J. Günther, C. Facchi, and L. Wolf, "Artery: Extending veins for VANET applications," in *Proc. Models Technol. Intell. Transp. Syst. (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 Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1313–1324, May 2017.
[11] JIST/SWANs. *Jist/Swans Simulator*. Accessed: May 2018. [Online]. Available: http://jist.ece.cornell.edu/
[12] R. Fernandes and P. M. d'Orey, and M. Ferreira, "DIVERT for realistic simulation of heterogeneous vehicular networks," in *Proc. IEEE 7th Int. Conf. Mobile Adhoc Sensor Syst. (MASS)*, Nov. 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," *Simul. Model. Pract. Theory*, vol. 26, pp. 151–167, Aug. 2012.
[14] V. Kumar *et al.*, "iTETRIS: Adaptation of ITS technologies for large scale integrated simulation," in *Proc. IEEE Veh. Technol. Conf. (VTC-Spring)*, May 2010, pp. 1–5.
[15] *Working With Network Simulators*. Accessed: Mar. 2018. [Online]. Available: https://goo.gl/pnzYNW
[16] J. D. Walker and S. C. Chapra, "A client-side Web application for interactive environmental simulation modeling," *Environ. Model. Softw.*, vol. 55, pp. 49–60, May 2014.
[17] P. Lorenz, T. J. Schriber, H. Dorwarth, and K.-C. Ritter, "Towards a Web based simulation environment," in *Proc. 29th Conf. Winter Simulation*, 1997, pp. 1338–1344.
[18] 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 *Proc. Symp. Theory Modeling Simulation-DEVS Integrative M&S Symp.*, 2013, Art. no. 34.
[19] D. Zehe, A. Knoll, W. Cai, and H. Aydt, "SEMSim cloud service: Large-scale urban systems simulation in the cloud," *Simul. Model. Pract. Theory*, vol. 58, pp. 157–171, Nov. 2015.
[20] Y. Qu and X. Zhou, "Large-scale dynamic transportation network simulation: A space-time-event parallel computing approach," *Transp. Res . C, Emerg. Technol.*, vol. 75, pp. 1–16, Feb. 2017.
[21] H. Liu, K. Wang, Z. Chen, B. Yang, and R. He, "Large-scale reservoir simulations on distributed-memory parallel computers," in *Proc. 24th High Perform. Comput. Symp.*, 2016, Art. no. 8.
[22] J. D. Rhodes *et al.*, "Experimental and data collection methods for a large-scale smart grid deployment: Methods and first results," *Energy*, vol. 65, pp. 462–471, Feb. 2014.
[23] P. Janovsky and S. A. DeLoach, "Multi-agent simulation framework for large-scale coalition formation," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI)*, Oct. 2016, pp. 343–350.

[24] S. Coakley *et al.*, "Large-scale simulations with flame," in *Intelligent Agents in Data-Intensive Computing*. Cham, Switzerland: Springer, 2016, pp. 123–142.
[25] 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 Comput., Pract. Exper.*, vol. 28, pp. 2024–2030, May 2016.
[26] G. D'Angelo and M. Marzolla, "New trends in parallel and distributed simulation: From many-cores to cloud computing," *Simul. Model. Pract. Theory*, vol. 49, pp. 320–335, Dec. 2014.
[27] S. Guo, F. Bai, and X. Hu, "Simulation software as a service and service-oriented simulation experiment," in *Proc. IEEE Int. Conf. Inf. Reuse Integr.*, Aug. 2011, pp. 113–116.
[28] E. Cayirci, "Modeling and simulation as a cloud service: A survey," in *Proc. Winter Simulation Conf. (WSC)*, 2013, pp. 389–400.
[29] R. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exper.*, vol. 41, no. 1, pp. 23–50, 2011.
[30] A. W. Malik *et al.*, "CloudNetSim++: A GUI based framework for modeling and simulation of data centers in OMNeT++," *IEEE Trans. Serv. Comput.*, vol. 10, no. 4, pp. 506–519, Jul./Aug. 2017.
[31] U. ur Rahman, O. Hakeem, M. Raheem, K. Bilal, S. U. Khan, and L. T. Yang, "Nutshell: Cloud simulation and current trends," in *Proc. IEEE Int. Conf. Smart City/SocialCom/SustainCom (SmartCity)*, Dec. 2015, pp. 77–86.
[32] 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 Generat. Comput. Syst.*, vol. 74, pp. 104–118, Sep. 2017.
[33] G. D'Angelo, "The simulation model partitioning problem: An adaptive solution based on self-clustering," *Simul. Model. Pract. Theory*, vol. 70, pp. 1–20, Jan. 2017.
[34] S. J. E. Taylor, T. Kiss, G. Terstyanszky, P. Kacsuk, and N. Fantini, "Cloud computing for simulation in manufacturing and engineering: Introducing the cloudsme simulation platform," in *Proc. Annu. Simulation Symp.*, 2014, Art. no. 12.
[35] M. Mubarak, C. D. Carothers, R. B. Ross, and P. Carns, "Enabling parallel simulation of large-scale HPC network systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 1, pp. 87–100, Jan. 2017.
[36] M. A. Iqbal, M. Aleem, M. A. Islam, M. Ibrahim, and S. Anwar, "Amazon cloud computing platform EC2 and VANET simulations," *Int. J. Ad Hoc Ubiquitous Comput.*, to be published.
[37] Eucalyptus Systems, Inc. *Eucalyptus Community Cloud*. Accessed: May 2018. [Online]. Available: http://open.eucalyptus.com/try/community-cloud
[38] *JavaSysMon*. Accessed: May 2018. [Online]. Available: https://github.com/jezhumble/javasysmon/wiki
[39] A. C. Zhou, B. He, and S. Ibrahim, "A taxonomy and survey of scientific computing in the cloud," in *Big Data: Principles and Paradigms*. San Mateo, CA, USA: Morgan Kaufmann, 2016.
[40] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and Linux containers," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Mar. 2015, pp. 171–172.
[41] G. A. Wainer and P. J. Mosterman, *Discrete-Event Modeling and Simulation: Theory and Applications*. Boca Raton, FL, USA: CRC Press, 2016.
[42] S. Cloudin and P. M. Kumar, "Challenges on mobility models suitable to vanet," *J. Softw.*, vol. 12, no. 2, pp. 91–101, 2017.
[43] S. Kurt and B. Tavli, "Path-loss modeling for wireless sensor networks: A review of models and comparative evaluations," *IEEE Antennas Propag. Mag.*, vol. 59, no. 1, pp. 18–37, Feb. 2016.
[44] P. Xie, J.-H. Cui, and L. Lao, "VBF: vector-based forwarding protocol for underwater sensor networks," in *Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems*, vol. 3976. Berlin, Germany: Springer, 2006, pp. 1216–1221.
[45] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin, "Wireless sensor networks: A survey on recent developments and potential synergies," *J. Supercomput.*, vol. 68, no. 1, pp. 1–48, 2014.
[46] A. Virdis, C. Vallati, and G. Nardini. (2016). "Automating large-scale simulation and data analysis with OMNET++: Lession learned and future perspectives." [Online]. Available: https://arxiv.org/abs/1609.04603
[47] *Microsoft Azure*. Accessed: May 2018. [Online]. Available: https://azure.microsoft.com/en-us/
[48] U. Wajid *et al.*, "On achieving energy efficiency and reducing $CO_2$ footprint in cloud computing," *IEEE Trans. cloud comput.*, vol. 4, no. 2, pp. 138–151, Apr./Jun. 2016.
[49] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong, "Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing," *J. Netw. Comput. Appl.*, vol. 59, pp. 46–54, Jan. 2016.
[50] S.-Y. Jing, S. Ali, K. She, and Y. Zhong, "State-of-the-art research study for green cloud computing," *J. Supercomput.*, vol. 65, no. 1, pp. 1–24, 2013.
[51] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *Proc. 1st ACM Symp. Cloud Comput.*, 2010, pp. 39–50.
[52] A. Varga and A. Sekercioglu, "Parallel simulation made easy with OMNeT++," in *Proc. 15th Eur. Simulation Symp. Exhib.*, 2003, pp. 493–499.

**MUHAMMAD IBRAHIM** received the M.S. degree in information technology from the Shaheed Zulfikar Ali Bhutto Institute of Science and Technology, Islamabad, in 2009. He is currently pursuing the Ph.D. degree from the Capital University of Science and Technology, Islamabad, Pakistan. He is currently an Instructor with the Virtual University of Pakistan. His area of research include large-scale network simulation and modeling in Cloud.

**MUHAMMAD AZHAR IQBAL** received the Ph.D. degree in communication and information systems from the Huazhong University of Science and Technology, Wuhan, China, in 2012. He is currently an Assistant Professor with the Department of Computer Science, Capital University of Science and Technology, Islamabad, Pakistan. His research areas include coding-aware routing in vehicular ad hoc networks, energy-efficient MAC for wireless body area networks, and large-scale simulation modeling and analysis of computer networks in Cloud.

**MUHAMMAD ALEEM** received the Ph.D. degree in computer science from Leopold-Franzens-University, Innsbruck, Austria, in 2012. He is currently an Assistant Professor with the Department of Computer Science, Capital University of Science and Technology, Islamabad, Pakistan. His research interests include parallel and distributed computing comprises programming environments, multi-/many-core computing, performance analysis, Cloud computing, big data processing, and scheduling.

**MUHAMMAD ARSHAD ISLAM** received the Ph.D. degree from the University of Konstanz, Germany, in 2011. His dissertation is related to routing issues in opportunistic network. He is currently an Assistant Professor with the Capital University of Science and Technology, Islamabad, Pakistan. His current research interests are related to MANETs, DTNs, social-aware routing, and graph algorithms.