CloudNetSim++: A GUI Based Framework for Modeling and Simulation of Data Centers in OMNeT++

Asad W. Malik, Kashif Bilal, *Student Member, IEEE*, S. U. Malik, *Member, IEEE*, Zahid Anwar, Khurram Aziz, *Senior Member, IEEE*, Dzmitry Kliazovich, Nasir Ghani, *Senior Member, IEEE*, Samee U. Khan, *Senior Member, IEEE*, and Rajkumar Buyya, *Fellow, IEEE*

Abstract—State-of-the-art cloud simulators in use today are limited in the number of features they provide, lack real network communication models, and do not provide extensive Graphical User Interface (GUI) to support developers and researchers to extend the behavior of the cloud environment. We propose CloudNetSim++, a comprehensive packet level simulator that enables simulation of cloud environments. CloudNetSim++ can be used to evaluate a wide spectrum of cloud components, such as processing elements, storage, networking, Service Level Agreement (SLA), scheduling algorithms, fine grained energy consumption, and VM consolidation algorithms. CloudNetSim++ offers extendibility, which means that the developers and researchers can easily incorporate own algorithms for scheduling, workload consolidation, VM migration, and SLA agreement. The simulation environment of CloudNetSim++ offers a rich GUI that provides a high level view of distributed data centers connected with various network topologies. The package also includes an energy computation module that provides a fine grained analysis of energy consumed by each component. This paper shows the flexibility and effectiveness of CloudNetSim++ through experimental results demonstrated using real-world data center workloads. Moreover, to demonstrate the correctness of CloudNetSim++, we performed formal modeling, analysis, and verification using High-level Petri Nets, Satisfiability Modulo Theories (SMT), and Z3 solver.

Index Terms—Cloud computing, data center, OMNeT++, energy efficiency, cloud simulator, virtual machine migration

1 Introduction

The extensive adoption of high speed Internet and availability of low cost computers is changing the high performance computing paradigm. At the same time, enterprises need to analyze huge amounts of data to determine future trends and reason about adoption in their business policies. The entailed big data analysis and immense storage requirement is not possible without readily available cloud infrastructure. Cloud computing is an emerging paradigm that is growing rapidly [1] and offers virtualized scalable resources as services over the Internet.

Clouds are formed from a large set of geographically distributed data centers. Multinational Information Technology (IT) companies, such as Google, Facebook, Amazon, and Microsoft are pioneers of the cloud computing paradigm [2], currently providing a variety of cloud-hosted services, over the Internet. Cloud based applications are gaining more popularity due to the vast adoption of the cloud paradigm in various sectors of life [34] and quality of service (QoS) offered by cloud service providers to end users [3].

The cloud provides economical, flexible, and reliable deployment and hosting of services by offering pay-per-use and elastic (procurement and release of resource on the fly) model [3]. The cloud offers services to facilitate enterprises, small businesses and developers to utilize readily available hardware and software resources to save on huge Capital Investments (CaPex) in procurement of new equipment. On the cloud providers end, one of the most critical requirements of the cloud environment is to provide a guaranteed Quality of Service. In a multi-tenant based cloud environment, multiple VMs share the same physical computing node. Aggressive consolidation and workload unpredictability among VMs may results in decreased QoS, and ultimately violations of SLAs. Therefore, cloud providers must provide robust services to handle workload perturbations and to meet the required QoS and SLA while performing consolidation [4].

Due to diminishing reserves of fossil fuel, high energy demands, and increased environmental concerns, another

- *A. W. Malik and Z. Anwar are with the Department of Computing, National University of Sciences and Technology (NUST-SEECS), Islamabad, Pakistan. E-mail: {asad.malik, zahid.anwar}@seecs.edu.pk.*
- *K. Bilal and S. U. R. Malik are with the Department of Computer Science, COMSATS Institute of Information Technology, Pakistan. E-mail: kashif.bilal@my.ndsu.edu, saif_ur_rehman@comsats.edu.pk.*
- *K. Aziz is with the Department of Electrical Engineering, COMSATS Institute of Information Technology, Pakistan. E-mail: khurram.aziz@comsats.edu.pk.*
- *D. Kliazovich is with the Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg. E-mail: Dzmitry.Kliazovich@uni.lu.*
- *N. Ghani is with the Electrical Engineering Department, University of South Florida, FL. E-mail: nghani@usf.edu.*
- *S. U. Khan is with the Department of Electrical and Computer Engineering, North Dakota State University, Fargo. E-mail: samee.khan@ndsu.edu.*
- *R. Buyya is with the CLOUDS Laboratory, University of Mel-bourne, Australia. E-mail: rbuyya@unimelb.edu.au.*

Manuscript received 9 June 2015; revised 12 Sept. 2015; accepted 20 Oct. 2015. Date of publication 29 Oct. 2015; date of current version 9 Aug. 2017. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TSC.2015.2496164

major apprehension of cloud providers is to minimize energy consumption and reduce carbon emissions. The modern-day cloud data center consumes a considerable amount of energy [5], has significant operational cost and requires the availability of specialized cooling infrastructure to operate efficiently [6]. This can be mainly attributed to workload computation at each data center server causing it to consume a significant amount of power and disseminating heat that further requires specialized cooling equipment. Therefore, the estimated overall cost of cooling equipment is within a range of 2-5 million US$ per year [7].

The total energy consumption of IT equipment inside data centers is estimated to be around 71 percent [8]. Therefore, major IT companies are reluctant to expand to several data centers for improving capacity and performance. To efficiently utilize power, techniques such as Dynamic Voltage and Frequency Scaling (DVFS) and Dynamic Power Management (DPM) have been adopted [10]. Service providers will usually always over-provision resources to handle peak load. On the other hand, the average workload is around 30 percent of an entire data center's processing capacity [11]. The aforementioned facts imply that idle resources can be transitioned to sleep mode unless required to handle increased workload. Efficient utilization of data centers in term of computing and energy is still an open area for researchers to explore. At the same time, cloud infrastructure demands new scheduling, resource allocation, and performance optimization techniques to minimize the energy consumption. Such novel methodologies and optimization techniques require considerable testing and evaluation before deploying in a production environment. However, because of huge setup costs, it is infeasible to test the proposed schemes in real cloud testbeds. Moreover, use of virtualized cloud infrastructure and production environments, such as Amazon EC2, limit the experimental setup and make the reproduction of experimental results extremely difficult. The underlying architectural details are hidden and beyond the control of the tenant or deployed service. Therefore, the extensive study and evaluation of novel techniques and algorithms for a distributed environment can only be achieved through simulations.

To support the study of realistic cloud scenarios which allow fine grained control over individual data center components, we propose CloudNetSim++, a realistic network simulator that allows modeling, simulation, and evaluation of geographically distributed data centers, with multiple users, varying pricing models, energy auditing, and VM consolidation schemes based on real-world data center workloads. CloudNetSim++ is developed as an extension of the existing OMNeT++ discrete event simulator and provides an extensive framework that can be extended with ease and limited efforts. CloudNetSim++ also provides an infrastructure that allows researchers to focus on a specific problem without worrying about the underlying details. In particular, the salient features of CloudNetSim++ can be summarized as follows:

1) Provides support for modeling and simulation of individual or large scale geographically distributed data centers connected with a high speed network.
2) Resources are assigned to ensure SLA.
3) Includes different scheduling algorithms for resource allocation. Users can select the scheduling algorithm that seems most appropriate for their workload and tasks scheduling. The purpose of this feature is to provide a template to incorporate custom scheduling algorithms and analyze the resulting behavior.
4) Offers support for a heterogeneous computing environment. The user can define resources with varying CPU power, RAM, and storage space for individual compute nodes.
5) Heterogeneity in terms of different resources within a rack is also provided. Users can also specify different number of computational nodes in each rack.
6) Fine-grained energy auditing for computational resources is provided. Energy consumption by each VM and compute node is calculated.
7) Computes energy consumed by network elements, such as switches/routers.
8) Provides support for multiple users, delivering a more realistic multi-tenant cloud environment.
9) A user can specify VMs with high priority, to deal with and service high priority traffic
10) Provides different pricing modules for data centers services.
11) Provides a cloud usage monitor module to monitor and analyze usage patterns and notify users in case of SLA breach.
12) Provides a rich GUI based environment that allows users to visualize the entire network along with fine-grained packet flows. Thr GUI support further simplifies analysis and debugging.

Overall the proposed solution uses real physical network properties, provides distributed data center simulation support, features heterogeneous configuration of racks and servers connected through a three tier architecture. These contributions makes it a novel tool as compared to existing simulators. The rest of the paper is organized as follows: Section 2 briefly describes the most commonly used cloud simulators. In Section 3, CloudNetSim++ architecture design is explained. The experimental setup and simulation details are presented in Section 4. Finally, we conclude this paper by providing future research directions pertaining to the CloudNetSim++ simulator in Section 5.

2 Comparison with Other Cloud Computing Simulators

This section provides a comparison between CloudNetSim++ and other related cloud simulators. A brief summary of existing simulators is given below.

CloudSim [12] is the most widely known cloud computing simulator developed at University of Melbourne. It is event based, implemented in JAVA and provides a simulation and experimentation environment for computing and application services. CloudSim can model network components, such as switches, but lacks fine-grained communication models of links and Network Interface Cards (NIC) causing VM migration and packet simulation to be network-unaware. Since the network is a critical component, which significantly affects performance, throughput, and makespan, the obtained simulation results cannot reflect a real cloud environment. DartCSim+ [13] is developed as an extension of CloudSim that supports the modeling of communication link delay. Similarly, NetworkCloudSim [14] is proposed as another extension to provide support for cloud environments running different types of applications communicating via message exchanges, such as HPC and E-Commerce. CDOSim [41] extends CloudSim and integrates with the cloud migration framework (CloudMIG) [42]. CDOSim and CloudMIG are based on the enterprise resource planning system model. CloudExp [26] is another recently developed simulator based on CloudSim as an enhancement to the latter. It is developed in Java, provides a Graphical User Interface (GUI), and focuses on big data management and mobile cloud computing. CloudExp includes a rain cloud workload generator from Berkeley [27].

CloudAnalyst [3] aims to achieve optimal scheduling among user groups and data centers based on the current configuration. Both CloudSim and CloudAnalyst are based on SimJava [15] and GridSim [16], [17], which treat a cloud data center as a large resource pool, and consider only application-level workloads. Tian et al. [18] propose CloudSched, a novel lightweight simulation tool for real-time VM scheduling in cloud data centers.

GreenCloud [8] is a relatively new simulator compared to CloudSim, built on top of the NS2 simulation framework [12]. GreenCloud was a collaborative project of the University of Luxembourg and North Dakota State University. The GreenCloud simulator uses a packet level communication model to simulate complex simulations. iCancloud [9] is designed to perform large cloud computing simulations and is built on top of OMNeT++. It allows users to measure the cost and performance of their applications running on a variety of hardware. It includes Amazon EC2 and hypervisor support, through which a user can include different policies and compare results.

GroudSim [19] is a discrete event simulator, developed using Java as the underlying programming language. It is designed to support grid and cloud computing. GroudSim provides a framework to execute complex simulation scenarios that simulate the job execution, cost calculation, and background workload. This simulator is mainly focused on the IaaS model of the cloud and provides an extensive framework to support flexible extension for other cloud services.

SimIC [20] is a discrete event simulation tool based on the SimJava package. SimIC models the inter-cloud activities that include exchange of services between cloud data centers. SIMCAN [21] and SIMSANs are OMNeT++ based simulators, developed to simulate storage networks with the objective of analyzing network performance and detecting system bottlenecks. These simulators allow researchers to develop their own algorithms to analyze the performance on the underlying I/O and network subsystems. SPECI [22] is designed on SimKit [23] with the goal of simulating large-scale data centers. EMUSIM [24] is designed on top of CloudSim and the Automatic Emulation Framework (AEF) and is targeted towards evaluating the performance of cloud services. DCSim [25] is designed to allow rapid development and analysis of data center management models largely focused on VM management techniques.

MDCsim [43] cloud simulates the features of large scale multi-tier data centers. It models power utilization as well as links and switches connected along with the nodes. SmartSim [44] uniquely addresses simulation of mobile cloud applications and considers resource provisioning and utilization of the mobile device processor as part of the model.

In summary, the main differences among existing simulators with CloudNetSim++ are described below:

Graphical user interface support. The original CloudSim did not support a graphical interface. A graphical interface for limited support for configuration and viewing of results is supported in CloudAnalyst. The same is the case for GreenCloud. CloudSched [45] and iCanCloud on the other hand allow the entire scheduling process to be viewed via the GUI.

CloudNetSim++ is a distributed data center simulator completely written in C++. It provides an extensive GUI for researchers to perform in-depth analysis of every single module including but not limited to routing protocols, link delays, bit error rate (BER) and system TCP/IP stack and distributed data centers.

Precision of physical models. Model details about the simulated components, such as servers and network components effect the precision of the simulation and the validity of the results. iCanCloud and CloudSched provide the ability of viewing detailed traces of resource utilization in physical servers. GreenCloud requires plug-ins to simulate and capture the packet loss. Physical server models are not provided in CloudSim.

CloudNetSim++ provides a platform for simulating distributed data centers with actual physical link properties. A user can configure a comprehensive set of properties including bit error rate, link delay and packet drop rate. CloudNetSim++ supports heterogeneous hardware, and VMs are assigned based on users' requests through SLAs. The heterogeneity of hardware is handled in terms of three parameters i.e. CPU, memory and hard drive. It also provides the provision to configure all the systems with the same specifications or explicitly define different specification for each individual system/rack by tuning the required parameters in a configuration file.

Compatibility with real cloud provider interfaces. The iCanCloud and CloudSched simulators use the scheduling specification models proposed by Amazon EC2. When creating these models, the EC2 default specifications are used. Similarly public cloud providers allow VM migration features to satisfy specific objectives, for instance, dealing with the overload scenario in load balancing applications. In some application scenarios where migration operations are triggered, CloudSim and CloudSched migrate the requests to other hosts to comply with predefined configurations.

The pricing module inside CloudNetSIM++ is used to track the resources consumed by different users and generate a bill accordingly. The billing mechanism in CloudNetSim++ is based on the concept used in Amazon EC2.

Parallel experiments and power consumption model. Support for multiple machines running experiments simultaneously to perform a task is a novel feature of iCanCloud and is under development in CloudSched. Power consumption models enable simulators to compare energy efficient strategies and energy consumption of components under research. iCanCloud does not support power consumption modeling; therefore, E-mc2 is designed on the top of iCanCloud to support a energy model [9]. The energy consumption model implemented in GreenCloud allows the tracing

TABLE 1
Summary OF Most Commonly Used Simulators

Parameters	CloudNetSim++	iCanCloud	CloudSim	GreenCloud	CloudSched	MDCSim	CloudAnalyst
OS	Any	Any	Any	Linux	Any	Linux	Any
Platform	OMNeT++	OMNeT++	SimJava	NS2	-	J2EE	CloudSim
Language	C++	C++	Java	C++/OTcl	Java	J2EE/SQL	Java
Availability	Open source	Open source	Open source	Open source	Open source	Open source	Open source
Simulation Time	Tens of minutes	Tens of minutes	Seconds	Tens of minutes	Tens of minutes	Tens of minutes	Tens of minutes
Graphical Support	Extensive GUI	GUI	Limited	Limited	GUI	Web	GUI
Application Models	Yes	Yes	Yes	Yes	No	Limited	Yes
Communication Models	Real physical network	Yes	No	Yes	No	Limited	No
Energy Model	Yes	Yes	Yes	Yes	Yes	No	Yes
Power Saving Mode	Yes	No	Yes	Yes	No	Yes	Yes
Distributed Datacenter Model	Yes	No	Yes	No	No	No	Yes
VM Support	Yes	Yes	Yes	Yes	Yes	No	Yes
SLA Support	Yes	No	Limited	No	No	No	Limited
Priorities Traffic	Yes	No	No	No	No	No	No
Multiple User Support	Yes	Yes	Yes	Yes	No	Yes	Yes
Heterogeneous Compute Node	Yes	No	Yes	Yes	Yes	No	Yes

of every single element in a data center. DVFS energy consumption model is proposed in CloudSim with the help of an extension tool. CloudSched provides power consumption metrics based on specifications suggested by Amazon.

The energy consumption module inside CloudNetSim++ provides complete information of energy auditing at different levels inside racks and switches. In addition, according to our survey most of the cloud simulators are mainly focused on VM management or applications/performance of individual data centers. CloudNetSim++ is the first simulator that provides a framework for distributed data center models. Table 1 shows a detailed comparative analysis of CloudNetSim++ with six other popular data center simulators iCanCloud, CloudSim, GreenCloud, CloudSched, MDCSim and CloudAnalyst. Section 3 presents the detailed architecture and functionality of CloudNetSim++.

3 CloudNetSim++: A Toolkit for Modeling and Simulation

CloudNetSim++ is a realistic network simulator that allows modeling and simulation of geographically distributed data centers, supporting multiple users, pricing models, energy-aware scheduling, and VMs consolidation. CloudNetSim++ is developed as an extension of OMNeT++ and uses the open source INET framework to provide support for different communication protocols.

CloudNetSim++ provides an extensive framework designed using a modular approach that can be extended with ease and limited effort. It provides an infrastructure that allows researchers to focus on studying a specific problem without worrying about the underlying architectural and communication details. CloudNetSim++ progresses a simulation through a well-defined time advancement module that supports equal and variable interval policy and allows users to schedule tasks at any interval of time. The high level architecture diagram of CloudNetSim++ is shown in Fig. 1. Another benefit of CloudNetSim++ over CloudSim is the support for modeling transmission links. CloudSim 3.0 does not support transmission links and only uses bandwidth that leads to the following issues.

1) It does not support transmission congestion and realistic delays/latency based on current network load. Without transmission link, the size of packet sent/received between switches and host may exceed the transmission capacity.
2) Cannot incorporate the aggregate and collective effects of traffic sent and received by multiple hosts sharing a single network link and path.
3) Power consumption of links and routing nodes cannot be measured.

CloudNetSim++, on the other hand, supports a simulation model of real physical network characteristics that caters network congestion, packet drops, bit error, and packet error rates. Researchers can define or modify physical characteristics of communication links to model realistic inter and intra-data center communication.

CloudNetSim++ is designed based on the compound module feature available in OMNeT++. The detailed working of compound module is available in [35]. The servers and routers are built using a modular approach. Researchers can link their own protocol schemes at different levels. In case of routers the physical and network layer is available

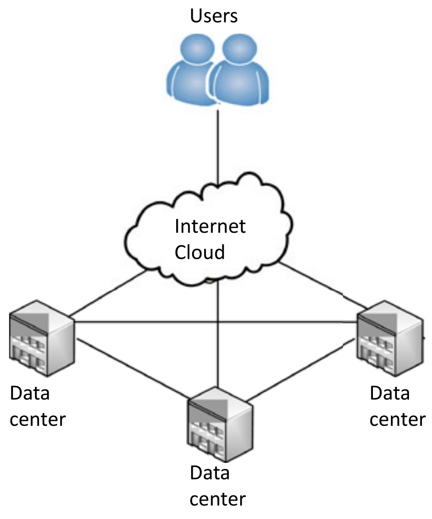


Fig. 1. High level architecture.

for the users and energy module for network elements is implemented as an extension of the network layer.

The application layer supports multiple applications running concurrently and communicating with other servers. The available application level protocols are HTTP, FTP, UDP and TCP. The nodes within CloudNetSim++ use DHCP to acquire dynamic IP addresses. The network layer inside servers and routers includes Address Resolution Protocol (ARP). This allows flexible increase in the number of servers and routers with ease and little effort.

CloudNetSim++ provides a framework for modeling and simulation of cloud infrastructure that allows simulation of geographically distributed data centers connected with a high speed network. The user requests services through SLA. The SLA includes service quality and other parameters, such as security, trust, obligation (service start, and end time), penalty, and required resources. Based on the SLA parameters, requested VMs are assigned to the end users. The Virtual Machine Manager (VMM) module is responsible for ensuring all of the requirements stated in the SLA.

Cloud computing data centers are increasingly popular for the provisioning of computing resources. These resources are billed on consumption. Therefore, efficient utilization of allocated resources is highly desirable for the end users. In CloudNetSim++, end users select the scheduling model used for incoming traffic, such as priority based and first-come-first-serve. This gives the user complete control over the allocated resources and helps in reducing the overall cost. At the time of SLA agreement, the user can specify the criteria for resource release. The available stopping criteria include:

1) User can release the assigned resources at any instant of time by sending a Service Termination Request (STR) message
2) Resources are assigned for a fixed time interval. On expiration, notification is send to the end user. This criteria allows users to keep using the assigned resources but billing cost includes the over usage cost plus the penalty defined in the SLA.

CloudNetSim++ provides access and location transparency to users as all of the traffic is managed through a centralized control mechanism. The CloudNetSim++ provides a framework to model multiple heterogeneous data centers. Here the term heterogeneous means that the numbers of VM hosts at each node, number of compute nodes in each rack, CPU, memory, storage capacity, and number of racks within each data center may vary. The cloud provider can configure these parameters through a simple user interface.

In CloudNetSim++, an energy computing module is placed at compute nodes and switches, which is used to compute energy consumption at different modules within data centers and the cloud. One of the objectives of CloudNetSim++ is to provide a platform to analyze the energy consumed at different components of distributed data centers. In CloudNetSim++, computing servers are the processing nodes. Each server has a computing power defined in terms of Million of Instructions Per Second (MIPS). A complete model of a data center is implemented where nodes are connected within racks using Top of the Rack (ToR) switches. These ToR switches are connected to aggregate switches; similarly aggregate switches are connected to core switches. The delay and Bit-Error-Rate is defined for each communication link. Finally, several geographically distributed data centers are connected to each other through several topologies. The module level diagram of the centralized interface is shown in Fig. 2.

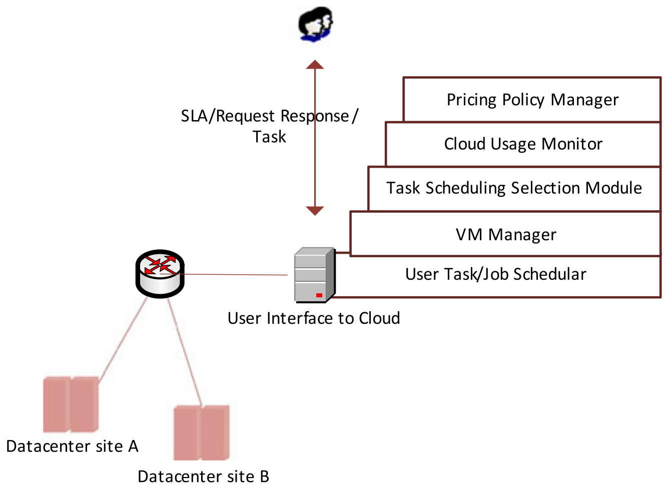


Fig. 2. Architecture of the CloudNetSim++ simulation environment.

The salient features of CloudNetSim++ are as follows:

- Geographically Distributed Datacenter GDC: CloudNetSim++ provides an extensive framework to simulate geographically distributed data centers networked together with the ability to control local and wide-area link parameters. The total number of data centers can be varied with ease.
- VM Manager VMM: The VM Manager module is responsible for VM assignment. It resides at the cloud interface and performs VM assignments based on received SLA requests. This module is responsible for VM consolidation, releasing of resources upon users' request or timeout. CloudNetSim++ categorizes users tasks into computation, communication intensive or hybrid. The VMM uses all these or other features (defined in SLA) for allocating VMs.
- User Task Scheduler UTS: The UTS receives all the incoming user traffic and directs it to the allocated VMs. While defining the SLA, users can select the scheduling algorithm from the existing set of algorithms for its incoming traffic.
- Energy Utilization EU: This module is placed at every VM compute node. It is responsible for calculating the energy consumed based on users' tasks. This module supports both Dynamic Voltage and Frequency Scaling, and Dynamic Network Shutdown - DNS. CloudNetSim++ measures the energy consumption at every VM, compute node, and data center.

Detailed features of CloudNetSim++ are explained below:

Service level agreement (SLA). A SLA is negotiated between a user and the resource provider and guarantees the availability of services during a specified time period. The SLA includes the number of resources, security, obligation, penalty and pricing. A vital parameter of SLA is QoS. The service provider must provide QoS that is continuously being monitored and deal with any perturbation that may happen

[28]. The detail of SLA parameters available in CloudNetSim++ are discussed below.

1) Security: CloudNetSim++ provides different levels of security. The VM manager uses this parameter to assign VMs on user request. Each data center provides a different level of security, based on location.
2) Trust: This factor is used while outsourcing user requests; researchers can extend CloudNetSim++ to incorporate data centers that belong to third parties.
3) Obligation: Two schemes are defined. A user can select a service for a limited time period after which services are automatically stopped, resources are released and the user is notified. In the other 'Pay-as-you consume' scheme services can be used for longer time periods.
4) Additional cost: This parameter indicates the additional cost a user needs to pay in case of a SLA breach or violation.
5) Required resources: This parameter is used to define the number of resources required in terms of CPU, disk, memory or the number of required VMs.
6) Task Scheduling Algorithm: A user can select the scheduling algorithm used for processing incoming traffic. This feature is especially made available for researchers so that they can incorporate and analyze their own scheduling algorithms on different data centers.
7) Dependent/Independent resources: A user can define the characteristics of the requested resources, either dependent or independent. In case of dependent resources, the requested VMs can communicate with each other, therefore they are placed either on the same node or inside the same rack or at least inside the same data center to minimize the communication overhead.
8) Max peak load per VM: This parameter is used to define the max peak load for each VM. The VM manager uses this parameter along with other parameters for VM consolidation and migration.
9) Pricing policy: A user can select a pricing model as:
    a) Lowest pricing - The most economical resources are required to host services. At the same time all other parameters must kept intact.
    b) Deadline based - This policy specifies that all the incoming tasks are deadline dependent, and must be completed within a specified time. In this case pricing would be on the higher side.
    c) Energy based - This policy means that the tasks are not deadline based; therefore, the resources can be assigned to minimize the energy consumption.

Pricing policy manager. This module calculates the billing cost for each individual user according to the agreement. It tracks the resources and timespan. The billing cost is calculated based on the resources occupied for the duration of time. CloudNetSim++ provides a user interface where cloud providers can set the billing rate per unit time for every data center based on its geographical location. The initial pricing policy is set close to the pricing plan of Amazon EC2 [3], [29].

Multi-tenancy. CloudNetSim++ provides a framework that allows multiple users to request resources. These resources are requested through SLA Request (SLA-R) message. Each user can request different numbers of virtual machines to run his application. The resource scheduler assigns resources keeping in view all of the SLA parameters. The SLA parameters are discussed shortly. In case of unavailability of resources, the SLA request message is put into the queue to serve later.

Energy consumption. The goal of the cloud computing paradigm is to utilize the computing power of data centers, which is considered as a replacement for office-based computing. However, to process a large amount of data, a significant amount of energy is required that is utilized by the switches, communication links, and cooling equipment. The energy consumed by the computing servers is around 30 percent; whereas, the rest of the energy is consumed by the network and cooling equipment [8], [30]. To efficiently utilize the resources to minimize energy consumption, several techniques are adopted, such as sleep scheduling and virtualization [11]. In CloudNetSim++, we present flexible data center models and compute detailed energy utilization of three components: (a) servers, (b) communication links, and (c) the data center network interconnects, such as router and switches. CloudNetSim++ also introduces the concept of distributed data centers, connected via highspeed physical network, where simulation of geographically distant data centers can be carried out by connecting the core nodes of these data centers through various topologies, while inside each data center, different architectures can be used. CloudNetSim++ allow researchers to explore different aspects of data center models through diverse traffic patterns. CloudNetSim++ is the first distributed data center simulator, developed over OMNeT++ and uses real physical network for communication. In CloudNetSim++, computing servers are the processing nodes. Each server has a computing power defined in terms of millions of instructions per second (MIPS). The complete model of a data center is implemented where racks are connected with ToR switches. These ToR switches are further connected with aggregate switches; similarly aggregate switches are connected with core switches. Finally, several geographically separated data centers are connected to each other through several network topologies.

CloudNetSim++ provides a user interface for defining computing servers for each rack. Each computing server is composed of multiple modules. Researchers can execute different applications depending on their requirements at the application module. OMNeT++ supports applications through its UDP, TCP, and HTTP classes. Researchers can generate different types of traffic using these classes. All other OMNeT++ features and frameworks developed using OMNeT++ can easily be incorporated in CloudNetSim++ with limited effort. In CloudNetSim++, energy model is incorporated inside every computing server. This energy model is implemented in a standalone module termed EStandardHost. To use this, a researcher must inherit their computing machine modules through EStandardHost. Similarly, energy modules of router are implemented in ERouter module and all of the routing and switching devices are inherited from ERouter module. CloudNetSim++ at the application layer allows the user to generate traffic at constant or random intervals. In CloudNetSim++, tasks are generated though the user module, which are scheduled on different data centers depending on their computing requirement.

As stated in [8], [30], the idle server consumes 66 percent of energy. This energy is utilized in handling different modules of a computing system, i.e., memory, disk and I/O. The level of power consumption increases linearly with increase in workload. Therefore, this requires a central scheduler that can schedule tasks on different computing servers and efficiently utilize the energy. The energy aware scheduler is connected with core switches to distribute user tasks. The power management can be achieved with Dynamic Voltage Frequency Scaling - DVFS technique. In DVFS, switching power in chip de-creases proportional to where V = voltage and F = switching frequency. Although frequency downshift would result in decrease in voltage, but there are certain components which are not linked with frequency, such as bus, memory and disk. The average power consumption is stated as below:

$$P = P_C + CPU_f * f. \tag{1}$$

Initially DVS was used for power optimization, but this could downgrade the transmission rate [10]. Therefore, the options for transmission rate that can be utilized inside data centers are 10 Mbps, 100 Mbps, or 1 Gbps. As stated in [8], the energy consumed by a switch can be expressed as:

$$P_{switch} = P_{chassis} + n_{linecard} * P_{linecard} + \sum_{i=0}^{R} n_{i,r} * P_r, \tag{2}$$

where:

P_{switch} : Power consumed by switch hardware,

$P_{linecard}$: Power consumed by an active network line card

P_r : Power consumed by a port (transceiver) running at the rate r, and

$n_{(i,r)}$: Number of port running at rate r

As indicated in above equation, only the last component is dependent on transmission rate, whereas the rest are independent. In case there is no transmission, the switch still utilizes power. This can be prevented by turning off the switch or using a sleep mode technique. CloudNetSim++ implements the energy consumption model as indicated in equation (2). In CloudNetSim++ we present a distributed data center model, where data centers are located at different geographical locations; and each data center may implement a different architecture. In this paper, our focus is on distributed three-tier architecture. CloudNetSim++ provides several options of connecting remotely located data center networks using a variety of components and topologies. Currently, the mesh, star, and ring topologies are supported. As an extension of CloudNetSim++, we would like to incorporate other data center models, such as BCube [31] and DCell [32] to introduce the concept of distributed energy aware scheduling between dissimilar architectures.

VM migration policy. Virtualization is widely used due to the benefits it offers including resource management, allocation and hardware utilization. A virtualization allows several operating systems to run on a single server simultaneously. It is easy to manage and deploy. A virtual machine (VM) is a software implementation that is independent of underlying hardware; this makes it easy to migrate from one server to another. The VM migration is performed to efficiently utilize hardware and dynamically handle growing users requirements. There are two modes involved in VM migration: 1) offline and 2) live migration. The offline migration causes more delay, as it is based on suspend and resume operation; whereas, live migration is based on pre-copy method.

CloudNetSim++ supports live virtual machine migration technique. Here we need to define a few terms necessary to understand the algorithm:

h_i : Compute Node i

H_n : All Compute Nodes

vm_j :Virtual Machine j

UsP_k : kth user preference defined through SLA

h_iw_i : workload on host i

e_i : energy consumption at host i

$list_v$: list of VMs for migration

$list_h$: list of potential servers

w_f: workload factor

In CloudNetSim++, migration is implemented through two modules, i.e., VM selection and server selection. The first module is responsible for selecting the VM for migration and latter module selects the target server. In this migration mechanism, the centralized load manager keeps track of all the VMs, their workloads (h_iw_i), energy consumption (e_i) and workload factor (w_f) of simulation model. VMs are assigned based on user request; multiple users can have VMs co-located on the same node depending on the type requested. Each node h_i can hold up to N VMs, i.e., (vm_0, $vm_1, \ldots,$ vm_n). Therefore, the selection of VM for migration (vm_i from node h_i to migrate to h_k) is dependent on preference criteria (UsP_k). A preference is given to the user whos vm is already executing on h_k; thus, reduce the VM-to-VM communication traffic. This approach is termed as co-location of communicating VMs [40] . The VM manager ensures that the migration does not violate SLA. Therefore, the server selection module generates a list of servers that can accommodate VMs. The VM manager filters the list based on the user requirement defined in terms of SLA. On the filtered list, a first fit scheduling algorithm is used to find a place for the migrated VM. Users can incorporate their own selection algorithms (by overriding the function call *VMServerSelectionPolicy*). The other technique is load based migration; where the load per node is monitored to handle the peek workload by migrating VMs to a different node. CloudSimNet++ offers a handy VM migration module for the researchers to implement their own VM migration techniques (override the function called *VMMigratePolicy*) to observe the details of the overall performance and energy consumption. In our default VM migration technique, we focus on the workload consolidation mechanism to achieve energy efficiency (see Table 2). The VM manager monitors the workload of each VM by inspecting their input queues and invokes the VM migration module based on the workload. The migration module is also invoked when a new VM request is received or at the exit of the VM, to consolidate the VMs on a minimum set of machines and to reduce the energy consumption. Currently CloudNetSim++ does not support dynamic addition of servers but we plan

TABLE 2
VM Migration Algorithm

VM Placement Algorithm

```
Step 1: VM Selection
  foreach $h_i$ in $H_n$ do
    if $h_i w_i > w_f$ or $e_i$ then
      foreach $vm_i$ in $h_i$ do
        $list_v \leftarrow vm_k$ based on $UsP_l$
      loop end
    endif
  loop end
Step 2: Host Selection
  foreach $h_i$ in $H_n$ do
    foreach $vm_i$ in $list_v$ do
      if $h_i$ accommodate $vm_i$ then
        $list_h \leftarrow h_i$
      loop end
    loop end
  loop end
Step 3: Migration Policy
  foreach $h_i$ in $list_h$ do
    if $h_i$ fulfil $UsP_l$ for $vm_i$ then
      migrate $h_i \leftarrow vm_i$
      remove $vm_i$ from $list_v$
    endif
  loop end
  if $list_v$ count $> 0$
    goto step 1, reselect $vm$ for migration
  end if
```

to implement a VM migration algorithm invoked on addition of servers in the future.

4 CloudNetSim++: Formal Verification and Performance Evaluation

CloudNetSim++ is designed to offer comprehensive cloud functionality and capabilities to its users to help them in understanding different aspects of cloud computing environments. Users can simulate experiments by changing infrastructure configuration through the *omnet.ini* file. To demonstrate the functionalities of CloudNetSim++, different set of experiments are conducted as examples and the results obtained using CloudNetSim++ are presented in this section.

4.1 CloudNetSim++ Formal Verification

Verification is the process of demonstrating the correctness of an underlying system. To determine the correctness of the model or a system, two parameters are required: (a) specification and (b) properties (that determines the correctness) [36]. In this study, we use bounded model checking [37] technique to perform the verification of CloudNetSim++, using SMT-Lib and Z3 solver.

4.1.1 High Level Petri Nets

Petri nets are very useful for mathematical and graphical modeling of wide range of systems [38]. In this work we have used a variant of conventional Petri nets, termed as High-Level Petri Nets (HLPN) for the formal modeling of CloudNetSim++. The HLPN is a 7-tuple structure represented as: $N = (P, T, F, \varphi, R, L, M_0)$ where:

1) P: refers to a set of places.
2) T: is the set of transitions such that $P \cap T = \phi$.
3) F: refers to the flow relation such that $F \subseteq (P \times T) \cup (T \cup P)$.
4) φ: is a mapping function used to map P to data types.
5) R: defines the transition rules used to map T to predicate formulas such that $R : T \rightarrow$ Formula.
6) L: defines a label that is used to map F to labels ($L : F \rightarrow$ Label).
7) $M0$: is the initial marking such that $M : P \rightarrow$ Tokens.

The HLPN models can efficiently represent the static and operational semantics of the system. The first three variables in the tuple (P, T, F) represent the static aspects, while the operational semantics are represented by (φ, R, L). The pre-conditions and post-conditions are specified to determine the flow of the system. Once the pre-condition(s) are met, the transitions are fired, causing post conditions to execute. In HLPN, places can have tokens of different types and can also be a cross product of two or more types, the places are mapped to the types: $\varphi(P_1) =$Boolean, $\varphi(P_2) =$ID, $\varphi(P_3) = P$(Integer), and $\varphi(P_1) =$Char. The set $P = \{P_1, P_2, P_3, P_4\}$ and $T = \{t_1, t_2, t_3\}$.

4.1.2 SMT-Lib and Z3 Solver

Satisfiability Modulo Theories (SMT) is an area of automated deduction for checking the satisfiability of formulas over some theories of interest and has the roots from Boolean Satisfiability Solvers (SAT) [36]. The SMT-Lib is an international initiative that provides a standard benchmarking platform that works on common input/output framework. In this work, we used Z3, a high performance theorem solver and satisfiability checker developed by Microsoft research [39].

4.1.3 Modeling, Analysis, and Verification

The HLPN model for the CloudNetSim++ is shown in Fig. 3. The model starts when the user requests the scheduler for the resources. The scheduler evaluates the request and decides whether to grant or deny the resources requested in the message.

The transition *Req-F* is mapped on to the formulas that represent the scenario, where the resource request is denied. The requests are denied (a) if the resources requested by the user are unavailable or (b) if the available resources are less than the resources requested, as shown in (3). Alternatively, the transition *Req-S* depicts the successful execution of the request, where the scheduler grants the request (shown in (4)) is mapped on to the formulas if the demand and availability of resources are matched, then the request is granted by the scheduler, as

$$\begin{aligned} &R(Req - F) = \forall cr \in Chk - Req \bullet cr \neq NULL \wedge \\ &\forall vr \in Val - Res \,|\, \nexists vr[1] = cr[2] \vee cr[3] > vr[3] \wedge \\ &Chk - Req' = Chk - Req \wedge \\ &Val - Res' = Val - Res \end{aligned} \tag{3}$$

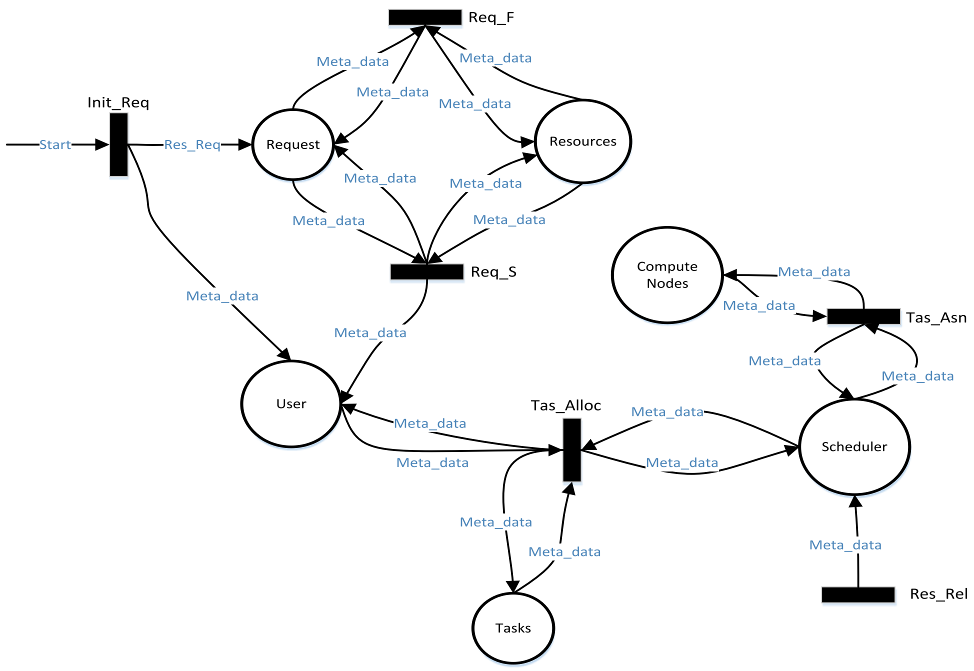


Fig. 3. The HPNL model for CloudNetSim++.

$$R(Req-S) = \forall cr \in Chk-Req \bullet cr \neq NULL \wedge \forall vr \in Val-Res | \exists vr[1] = cr[2] \vee cr[3] \leq vr[3] \wedge \forall rrt \in Req-Res-T \,|\, rrt[5] := True \wedge rrt[2] := cr[2] \wedge rrt[3] := cr[8] \wedge rrt[4] := cr[9] \wedge rrt[6] := VM-Assignment(rrt[1], cr[2]) \wedge Req-Res-T' = Req-Res-T \cup \{(rrt[1], rrt[2], rrt[3], rrt[4], rrt[5], rrt[6])\}. \tag{4}$$

As soon as the request is granted by the scheduler, the user can send tasks to the scheduler to allocate resources. The next transition Tas-Alloc is mapped on to the rules, where the scheduler receives the task from the user. In (5), the scheduler first authenticates the user. Once the user is verified, then the user id (encapsulated within the task message) is exploited to route the tasks to the assigned VM(s).

$$R(Tas-Alloc) = \forall gr \in Get-Rq \wedge \forall gq \in Get-Q \bullet (gq[5] = True \wedge gq[1] = gr[1] \wedge gq[2] = gr[2]) \rightarrow \forall gt \in Get-T \,| gt[4] = gq[1] \wedge \forall ats \in Alloc-T-Sc \,|\, ats[1] := gt[1] \wedge ats[2] := gt[3] \wedge ats[3] := gt[2] \wedge ats[4] := gr[8] \wedge ats[5] := gr[9] \wedge ats[6] := gr[2] \wedge ats[7] := gt[1] \wedge Alloc-T-Sc' = Alloc-T-Sc \cup \{(ats[1], ats[2], ats[3], ats[4], ats[5], ats[6], ats[7])\} \tag{5}$$

The transition *Tas-Asn* explains the process of assigning the tasks on to the compute nodes, as in (6). The compute nodes are scrutinized to execute the tasks based on the required VM configuration, such as CPU, hard disk, and memory requirement,

$$R(Tas-Asn) = \forall gs \in Get-Spec \wedge \forall asn \in Asn-t \wedge \forall au \in Auth \wedge gs[1] = au[1] \rightarrow asn[7] := au[6] \wedge asn[2] := gs[1] \wedge asn[3] := gs[2] \wedge asn[4] := gs[3] \wedge asn[5] := gs[4] \wedge asn[6] := gs[5] \wedge Asn-T' = Asn-T \cup \{(asn[1], asn[2], asn[3], asn[4], asn[5], asn[6]\}. \tag{6}$$

The last transition *Bill* depicts process of notifying the user about the dues payable by the user. As shown in (7), when the user generates a request for the resource release, then the scheduler releases the resources along with the bill payable by the user to the cloud. The dues are calculated based on the SLA between the user and the cloud,

$$R(Bill) = \forall gu \in Get-U \wedge \forall gi \in Get-I \bullet Res-Release(gu[1]) = True \rightarrow gu[7] := Bill(gi[6]) \wedge Get-U' = Get-U \cup \{(gu[1], gu[2], gu[3], gu[4], gu[5], gu[6], gu[7])\}. \tag{7}$$

To perform the verification process, the HLPN model is first translated to the respective SMT code. As stated above, to analyze the correctness of the model or the system, some properties must be specified. In the said perspective, the two properties we verified are (a) Request Verification (the

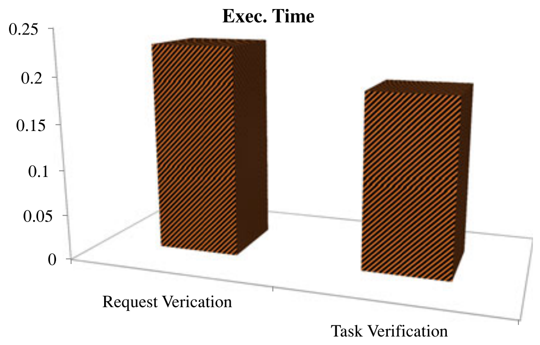


Fig. 4. Execution time for request and task verification properties.

TABLE 3
Machine Specification

Parameters	Value
CPU(s)	4
Thread(s) per core	2
Core(s) per socket	2
CPU family	6
CPU MHz	1801.000
Memory	4 Gb

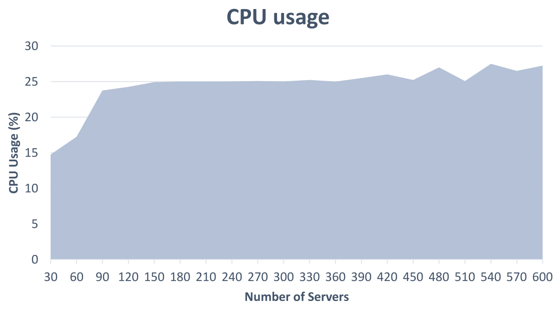


Fig. 5. CloudNetSim++ CPU usage.

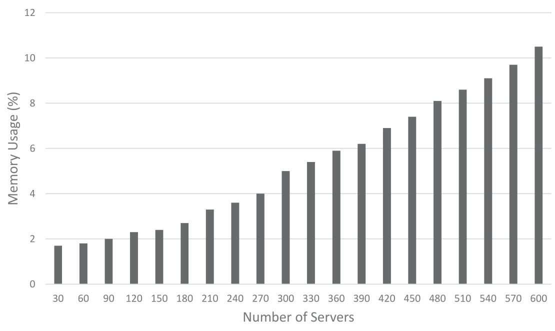


Fig. 6. CloudNetSim++ memory usage.

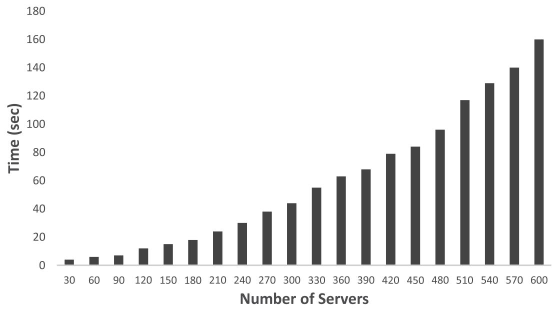


Fig. 7. Delay to construct enriched GUI.

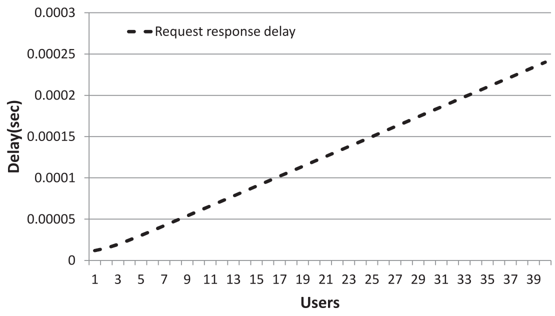


Fig. 8. Request response delay by varying the number of users.

resources allocated are according to the request or not) and (b) Task Verification (tasks are routed to the specified compute nodes or not). To prove the correctness of the model, the Z3 solver must not find any values (or combination of values) that violates the aforesaid properties. As stated above, we used bounded model checking approach for the verification of our system. In our case, the execution time (shown in Fig. 4) serves as a bound, which means that after the bound (execution time) the solver was unable to find any counter example to prove that the properties are not satisfied.

4.2 CloudNetSim++ Overhead Evaluation

To measure the CPU and memory usage, we conducted a series of tests by varying the number of nodes. In this experimental setup, two data centers are deployed at distant locations and all the nodes are symmetrically distributed across these data centers. The overhead is measured as the total delay in instantiating the simulation environment, CPU, and memory usage at the time of constructing a GUI. The profiling of CloudNetSim++ is carried out using linux top command to monitor the resources usage while constructing a GUI.

The machine specification on which the is conducted is shown in Table 3, and results of CPU, memory, and time are shown in Figs. 5, 6 and 7. We observed that the CPU and memory usage grow in steps when the number of hosts are varied in the experiment. The CloudNetSim++ provides a rich GUI and the obtained results illustrated that the time to instantiate an experiment setup with 600 hosts is around 150 s. This is a one-time overhead, at the time of initialization.

The other overhead parameters, CPU, and memory usage revealed that CloudNetSim++ consumes minimal amount of CPU and memory.

4.3 CloudNetSim++ Users Scalability Evaluation

CloudNetSim++ supports multiple concurrent users. The user sends a request to the centralized cloud Access Point (CAP) or centralize scheduler indicating the number of required resources. The CAP executes its resource assignment algorithm to allocate the requested resources. CloudNetSim++ provides a complete access and location transparency to the end users. The users are unaware of the location of nodes executing their jobs. The first experiment is conducted to measure the delay of requested resource assignments by varying the number of users. The delay measured in a CloudNetSim++ is shown in Fig. 8. This delay includes the time when the request is initiated by the user, and either the positive or negative response is

TABLE 4
Resource Requested

Type of Resources	Requested
VMs	2
Scheduling algorithm	FCFS
Pricing policy	Lowest
Priority VMs	01
Resources requested for time	60 minutes
Hard disk	100 GB
RAM	1 GB

TABLE 5
Lowest Pricing Policy

VMs	Memory	Storage	Price
1	1 GB	100 GB	$0.10 per Hour
1	2 GB	100 GB	$0.20 per Hour
2	4 GB	100 GB	$0.40 per Hour
4	4 GB	100 GB	$0.80 per Hour
8	4 GB	100 GB	$1.60 per Hour

TABLE 6
Priority Pricing Policy

VMs	Memory	Storage	Price
1	1 GB	100 GB	$0.20 per Hour
1	2 GB	100 GB	$0.40 per Hour
2	4 GB	100 GB	$0.80 per Hour
4	4 GB	100 GB	$1.60 per Hour
8	4 GB	100 GB	$3.20 per Hour

generated from the CAP module. In this experimental setup, each user requested the same number of resources; the requested resources are shown in Table 4. The delay in fulfilling the request increases with increase in the number of users, but the delay is still significantly less than a second. The experiment is conducted where up to 40 users generate a request concurrently and the max delay obtained is 0.00023 s.

4.4 CloudNetSim++ Pricing Scheme

CloudNetSim++ provides different pricing schemes. It categorizes data centers on lowest and highest priority schemes. During SLA, the user can request for highest, lowest pricing VMs, or hybrid pricing scheme. The pricing table of lowest and highest pricing data centers is shown in Tables 5 and 6. These pricing values are configurable and a user can change the values according to their requirements. This pricing table is available in *pricing.policy* file inside CloudNetSim++.

4.5 CloudNetSim++ Energy Module

CloudNetSim++ energy module computes the energy of individual components as well as the complete cloud infrastructure that includes servers, access, aggregate, and core switches. The energy is measured by executing the work load on multiple data centers. For the simulation, a real data center workload comprising of 22,385 jobs (around 128,000 tasks) is used. The data center workload is obtained from the Center for Computational Research of State University

TABLE 7
Simulation Parameters

S.no.	Parameters	Value
1	Inter-Data Center (DC) topology	Star/Mesh
2	Intra-DC topology	three-tier
3	Inter-DC link	100 Gbps
4	Data center to data center link (Bit Error Rate)	10^{-12}
5	Core to aggregate link	10 Gbps
6	Aggregate to access link	1 Gbps
7	Access to servers link	1 Gbps
8	Core to aggregate link (BER)	10^{-12}
9	Aggregate to access link (BER)	10^{-12}
10	Access link to computing servers (BER)	10^{-5}
11	Packet size	1,500 bytes
12	Core nodes	8
13	Aggregate nodes	16
14	Access nodes	256
15	Computing server	2,200 - 9,000

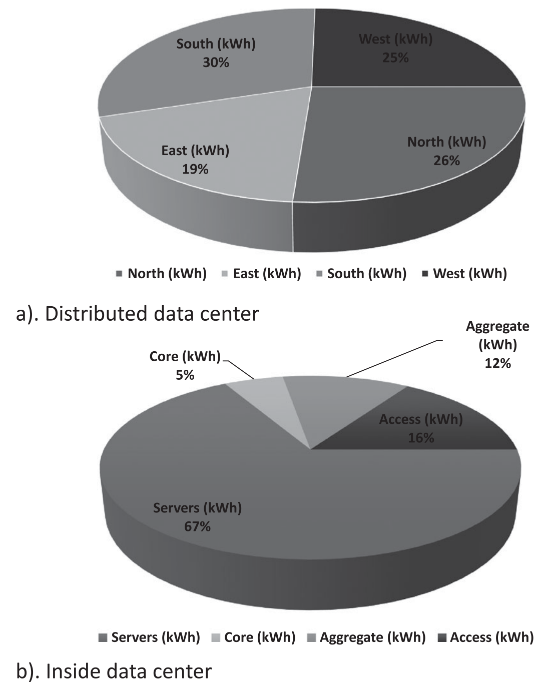


Fig. 9. Energy consumption a). multiple data centers b). inside data center.

of New York at Buffalo for evaluation purpose. The data center workload falls into two categories: (a) homogeneous and (b) heterogeneous resources. The resources in homogeneous systems are similar in terms of size and capacity, in which a job executes in similar capacity, whereas the resources in heterogeneous system are organized with different specification. The workloads were collected during a 30-day time period from February 20, 2009 to March 22, 2009 [33]. The other simulation parameters are shown in Table 7, and the calculated energy consumption is shown in Fig. 9.

CloudNetSim++ is designed using a modular approach, where each of the modules can easily be replaced with custom modules. This is one of the features of CloudNetSim++ offered to the research community.

Fig. 10 shows the inside view of “DistributedDataCenter”, where different data centers are shown connected with network topology and the right side of Fig. 10 shows the in-

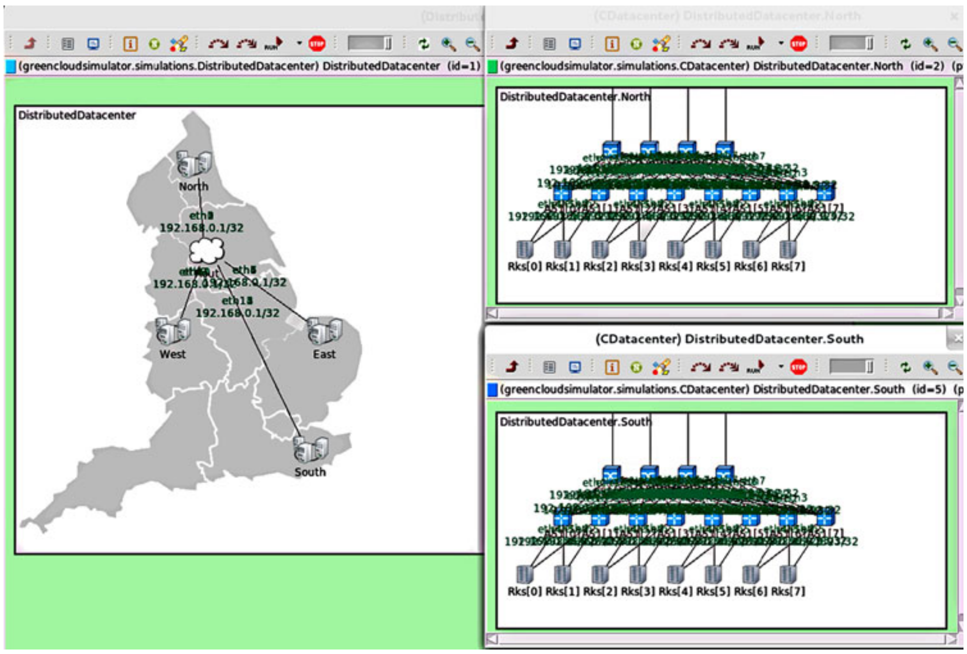

(greencloudsimulator.simulations.DistributedDatacenter) DistributedDatacenter (id=1)
DistributedDatacenter
North
West
East
South
192.168.0.1/32
(CDatacenter) DistributedDatacenter.North
(greencloudsimulator.simulations.CDatacenter) DistributedDatacenter.North (id=2)
DistributedDatacenter.North
Rks[0] Rks[1] Rks[2] Rks[3] Rks[4] Rks[5] Rks[6] Rks[7]
(CDatacenter) DistributedDatacenter.South
(greencloudsimulator.simulations.CDatacenter) DistributedDatacenter.South (id=5)
DistributedDatacenter.South
Rks[0] Rks[1] Rks[2] Rks[3] Rks[4] Rks[5] Rks[6] Rks[7]

Fig. 10. Detailed view of data centers inside CloudNetSim++.

depth view of data centers, their architecture, switches, and racks. Finally, Fig. 11 shows the inner arrangement of racks, number of servers connected within the racks. During simulation, CloudNetSim++ shows the complete flow of packets and the user can debug or inspect the values of the various parameters.

5 Conclusion

In this paper, we presented CloudNetSim++, a comprehensive cloud simulator designed for fine-grained network analysis to meet the users requirements. CloudNetSim++ is easy-to-use and efficient cloud computing modeling and simulation toolkit. CloudNetSim++ provides a rich GUI, energy module, VM migration, and distributed data center framework. It also supports distributed data centers and provides a mechanism to connect the data centers with different network topologies. CloudNetSim++ is a collaborative effort between researchers associated with different schools, where the researchers are constantly working to improve the simulator by incorporating different scheduling and VM migration algorithms. The unique features of CloudNetSim++ includes its support for researchers to incorporate their own algorithms. CloudNetSim++ provides a wide spectrum of cloud components, such as processing elements, storage, networking, Service Level Agreement,

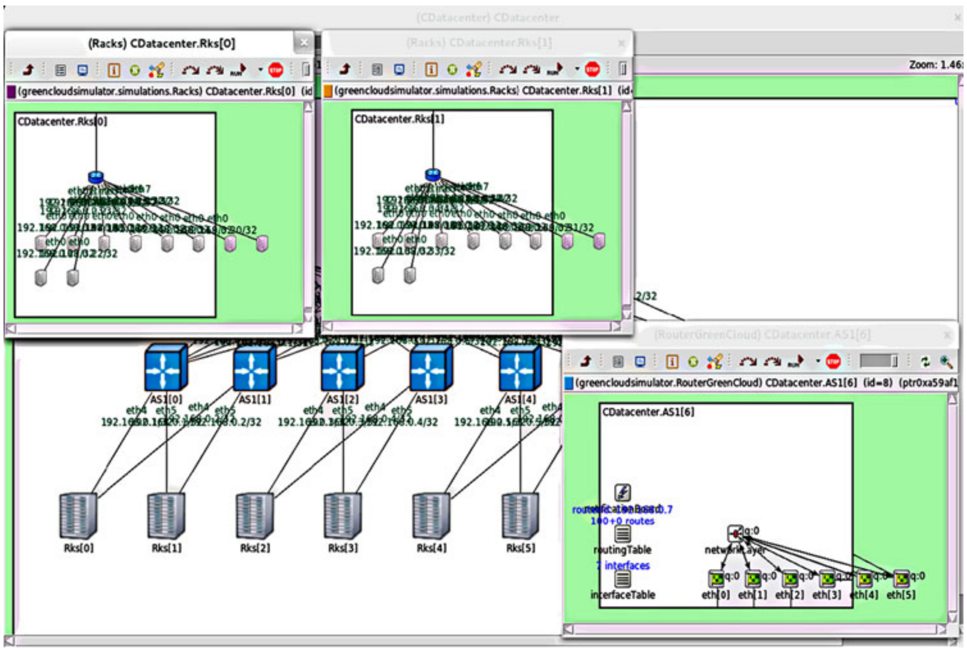

(CDatacenter) CDatacenter
(Racks) CDatacenter.Rks[0]
(greencloudsimulator.simulations.Racks) CDatacenter.Rks[0]
CDatacenter.Rks[0]
(Racks) CDatacenter.Rks[1]
(greencloudsimulator.simulations.Racks) CDatacenter.Rks[1]
CDatacenter.Rks[1]
Zoom: 1.46
AS1[0]
AS1[1]
AS1[2]
AS1[3]
AS1[4]
Rks[0]
Rks[1]
Rks[2]
Rks[3]
Rks[4]
Rks[5]
(RouterGreenCloud) CDatacenter.AS1[6]
(greencloudsimulator.RouterGreenCloud) CDatacenter.AS1[6] (id=8)
CDatacenter.AS1[6]
routingTable
interfaces
interfaceTable
networkLayer
eth[0] eth[1] eth[2] eth[3] eth[4] eth[5]

Fig. 11. Internal view of data centers inside CloudNetSim++.

scheduling algorithms, fine grained energy consumption, and VM consolidation algorithms. The other core feature includes an energy computation module that provides a detailed and fine grained analysis of energy consumed by each component. The correctness of CloudNetSim++ is performed through formal modeling, analysis, and verification using High-level Petri Nets, Satisfiability Modulo Theories, and Z3 solver.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, 2009.

[2] A. Vahdat, M. Al-Fares, N. Farrington, R. N. Mysore, G. Porter, and S. Radhakrishnan, "Scale-Out networking in the data center," *IEEE Micro*, vol. 30, no. 4, pp. 29–41, Jul.-Aug. 2010.

[3] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "CloudAnalyst: A CloudSim-based visual modeller for analysing cloud computing environments and applications," in *Proc. 24th IEEE Int. Conf. Adv. Inf. Netw. Appl.*, 2010, pp. 446–452.

[4] V. Goswami, S. S. Patra, and G. B. Mund, "Performance analysis of cloud with Queue-dependent virtual machines," in *Proc. 1st Int. Conf. Recent Adv. Inf. Technol.*, 2012, pp. 357–362.

[5] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "Energy aware network operations," in *Proc. 28th IEEE Int. Conf. Comput. Commun.*, 2009, pp. 25–30.

[6] R. Beik, "Green cloud computing: An energy-aware layer in software architecture," in *Proc. Spring Congress Eng. Technol.*, 2012, pp. 1–4.

[7] M. Pedram, "Energy-efficient datacenters," *IEEE Trans. Comput.-Aided Des. Integrated Circuits Syst.*, vol. 31, no. 10, pp. 1465–1484, Oct. 2012.

[8] D. Kliazovich, P. Bouvry, Y. Audzevich, and S. U. Khan, "GreenCloud: A packet-level simulator of energy-aware cloud computing data centers," in *Proc. IEEE Global Telecommun. Conf.*, 2010, pp. 1–5.

[9] G. G. Castane, A. Nunez, P. Llopis, and J. Carretero, "E-mc2: A formal framework for energy modeling in cloud computing," *Simul. Modelling Practice Theory*, vol. 39, pp. 56–75, Dec. 2013.

[10] S. Li, P. Li-Shiuan, and N. K. Jha, "Dynamic voltage scaling with links for power optimization of interconnection networks," in *Proc. 9th Int. Symp. High-Perform. Comput. Archit.*, 2003, pp. 91–102.

[11] G. Valentini, W. Lassonde, S. U. Khan, N. M. Allah, S. A. Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej, H. Li, A. Y. Zomaya, C. Z. Xu, P. Balaji, A. Vishnu, F. Pinel, J. E. Pecero, D. Kliazovich, and P. Bouvry, "An overview of energy efficiency techniques in cluster computing systems," *Cluster Comput.*, vol. 16, no. 1, pp. 3–15, 2013.

[12] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: Challenges and opportunities," *Int. Conf. High Performance Comput. & Simul.*, pp. 1–11, 2009.

[13] L. Xiang, J. Xiaohong, Y. Kejiang, and P. Huang, "DartCSim+: Enhanced cloudsim with the power and network models integrated," in *Proc. IEEE 6th Int. Conf. Cloud Comput.*, 2013, pp. 644–651.

[14] S. K. Garg, and R. Buyya, "NetworkCloudSim: Modelling parallel applications in cloud simulations," in *Proc. 4th IEEE Int. Conf. Utility Cloud Comput.*, 2011, pp. 105–113.

[15] W. Kreutzer, J. Hopkins, and M. V. Mierlo, "SimJAVA: A framework for modeling queueing networks in Java,"in *Proc. 29th Conf. Winter Simul.*, 1997, pp. 483–488.

[16] A. Sulistio, G. Poduval, R. Buyya, et al., "On incorporating differentiated levels of network service into GridSim," *Future Gener. Comput. Syst.*, vol. 23, no. 4, pp. 606–615, 2007.

[17] A. Sulistio, U. Cibej, S. Venugopal, R. Borut, and R. Buyya, "A toolkit for modelling and simulating data Grids: An extension to GridSim,"*Concurrency Comput.: Practice Exp.*, vol. 20, no. 13, pp. 1591–1609, 2008.

[18] W. Tian, Y. Zhao, M. Xu, Y. Zhong, and X. Sun, "A toolkit for modeling and simulation of real-time virtual machine allocation in a cloud data center," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 1, pp. 153–161, Jan. 2013.

[19] S. Ostermann, K. Plankensteiner, R. Prodan, and T. Fahringer, "GroudSim: An Event-based simulation framework for computational grids and clouds," in *Proc. Conf. Parallel Process.*, 2010, pp. 305–313.

[20] A. Ahmed and A. S. Sabyasachi, "Cloud computing simulators: A detailed survey and future direction," in *Proc. IEEE Int. Adv. Comput. Conf.*, 2014, pp. 866–872.

[21] A. Nez, J. Fernndez, R. Filgueira, F. Garca, and J. Carretero, "SIMCAN: A flexible, scalable and expandable simulation platform for modelling and simulating distributed architectures and applications," *Simul. Modelling Practice Theory*, vol. 20, no. 1, pp. 12–32, 2012.

[22] M. Jaatun, G. Zhao, and C. Rong, "SPECI, a simulation tool exploring Cloud-scale data centres," in *Proc. 1st Int. Conf. Cloud Comput.*, 2009, pp. 381–392.

[23] A. Buss, "Component based simulation modeling with Simkit," in *Proc. Winter Simul. Conf.*, 2002, vol. 1, pp. 243–249.

[24] R. N. Calheiros, M. A. S. Netto, C. A. F. De Rose, and R. Buyya, "EMUSIM: An integrated emulation and simulation Envi-ronment for modeling, evaluation, and validation of performance of Cloud computing applications," *Softw.: Practice Exp.*, vol. 43, pp. 595–612, 2013.

[25] M. Tighe, G. Keller, M. Bauer, and H. Lutfiyya, "DCSim: A data centre simulation tool for evaluating dynamic virtualized resource management," in *Proc. 8th Int. Conf. Workshop Syst. Virtualiztion Manage. Netw. Serv. Manage.*, 2012, pp. 385–392.

[26] Y. Jararweh, M. Jarrah, M. kharbutli, Z. Alshara, M. N. Alsaleh , and M. Al-Ayyoub, "CloudExp: A comprehensive cloud computing experimental framework," *Simul. Modelling Practise Theory*, vol. 49, pp. 180–192, 2014.

[27] A. Beitch, B. Liu, T. Yung, R. Griffith, A. Fox, and D. Patterson, "Rain: A workload generation toolkit for cloud computing applications," Elect. Eng. Comput. Sci. Dept., University of California, Berkeley, USA, Tech. Rep. UCB/EECS-2010-14, 2010.

[28] Z. Fengchuan, L. Hao, and J. Lu, "A service level Agree-ment framework of cloud computing based on the Cloud Bank model," in *Proc. IEEE Int. Conf. Comput. Sci. Autom. Eng.*, 2012, pp. 255–259.

[29] Amazon. (2015, Aug. 30). Amazon elastic compute cloud (Amazon EC2) [Online]. Available: http://aws.amazon.com/ec2/

[30] K. Bilal, S. Rehman, O. Khalid, A. Hameed, E. Alvarez, V. Wijaysekara, R. Irfan, S. Shrestha, D. Dwivedy, M. Ali, U. Shahid, A. Abbas, N. Jalil, and S. U. Khan , "A taxonomy and survey on green data center networks," *Future Gener. Comput. Syst.*, vol. 36, pp. 189–208, 2014.

[31] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C.Tian, Y. Zhang, and S. Lu, "BCube: A high performance, Server-centric net-work architecture for modular data centers," in *Proc. ACM SIGCOMM Conf. Data Commun.*, 2009, pp. 63–74.

[32] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: A scalable and fault-tolerant network structure for data centers," in *Proc. ACM SIGCOMM Conf. Data Commun.*, pp. 75–86, 2008.

[33] A. A. Chandio, K. Bilal, N. Tziritas, Z. Yu, Q. Jiang, S. U. Khan, and C.-Z. Xu, "A comparative study on resource allocation and energy efficient job scheduling strategies in Large-scale parallel computing systems," *Cluster Comput.*, vol. 17, no. 4, pp. 1349–1367, 2014.

[34] K. Bilal, S. Malik, S. U. Khan, and A. Zomaya, "Trends and challenges in cloud data centers," *IEEE Cloud Comput. Mag.*, vol. 1, no. 1, pp. 10–18, May 2014.

[35] A. W. Malik and S. U. Khan, "Data center modeling and simulation using OMNET++," *In Handbook on Data Centers*, S. U. Khan and A. Y. Zomaya, Eds. New York, NY, USA: Springer-Verlag, 2015, Chapter 28, pp. 839–855.

[36] S. U. R. Malik, S. U. Khan, and S. K. Srinivasan, "Modeling and analysis of State-of-the-art VM-based cloud management platforms," *IEEE Trans. Cloud Comput.*, pp. 50–63, 2013.

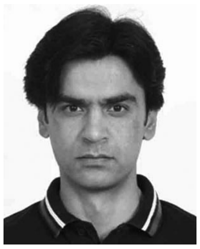
[37] A. Biere, A. Cimatti, E. Clarke, O. Strichman, and Y. Zhu, "Bounded model checking," in *Advances in Computers*, vol. 58. New York, NY, USA: Academic, 2003.

[38] Y. Quemener and T. Jeron, "Model checking of CL on infinite kripke structures defined by simple grammers," INRIA, France, Tech. Rep. RR-2563, 1995.

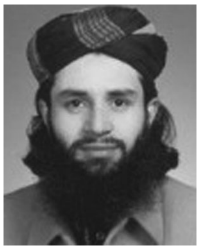
[39] S. U. R. Malik, S. K. Srinivasan, S. U. Khan, and L. Wang, "A methodology for OSPF routing protocol verification," in *Proc. Conf. Scalable Comput. Commun.*, pp. 1–5, Dec. 2012.

[40] R.W. Ahmad, A. Gani, S. H. Ab. Hamid, M. Shiraz, A. Yousafzai, and F. Xia, "A survey on virtual machine migration and server consolidation frameworks for cloud data centers," *J. Netw. Comput. Appl.*, vol. 52, pp. 11–25, 2015.

[41] F. Fittkau, S. Frey, and W. Hasselbring, “CDOSim: Simulating cloud deployment options for software migration support,” *Proc. IEEE 6th Int. Workshop Maintenance Evol. Serv.-Oriented Cloud-Based Syst.*, 2012, pp. 37–46.
[42] S. Frey, W. Hasselbring, and B. Schnoor, “Automatic conformance checking for migrating software systems to cloud infrastructures and platforms,” *J. Softw. Maintenance Evol.: Res. Practice*, vol. 25, no. 10, pp. 1089–1115, 2012.
[43] S. Lim, B. Sharma, G .Nam, E. K. Kim, and C. R. Das, “MDCSim: A multi-tier data center simulation platform,” in *Proc. IEEE Int. Conf. Cluster Comput. Workshops*, Aug. 2009, pp. 1–9.
[44] M. Shiraz, A. Gani, R. H. Khokhar, and E Ahmed, “An extendable simulation framework for modeling application processing potentials of smart mobile devices for mobile cloud computing,” in *Proc. 10th Int. Conf. Frontiers Inf. Technol.*, 2012, pp. 331–336.
[45] W Tian, Y. Zhao, M. Xu, Y. Zhong, and X. Sun, “A toolkit for modeling and simualtion of Real-time virtual machine allocation in a cloud data center,” *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 1, pp. 153–161, Jan. 2015



Asad W. Malik received the PhD degree in computer software engineering from the NUST, Pakistan. He is an assistant professor at the NUST School of Electrical Engineering and Computer Science, Pakistan. His research interests include parallel and distributed simulation, cloud computing, and large-scale networks.



Kashif Bilal received the PhD degree in electrical and computer engineering from the North Dakota State University. His research interests include data center networks, distributed computing, and energy efficiency. He is a student member of the IEEE.

Saif U. Malik received the PhD degree in electrical and computer engineering from the North Dakota State University. His research interests include data intensive cloud computational systems, formal verification, protocols in DataCenters. He is a member of the IEEE.

Zahid Anwar is an assistant professor at the NUST School of Electrical Engineering and Computer Science, Pakistan and an associate member of the Graduate Faculty at the University of North Carolina at Charlotte. His research is in cloud security and distributed systems.

Khurram Aziz is an assistant professor at the Comsats Institute of Information Technology, Pakistan. His research interests include optical networks. He is a senior member of the IEEE.

Dzmitry Kliazovich is a research fellow at the Faculty of science, technology, and communication of the University of Luxembourg. He holds an award-winning PhD in information and telecommunication technologies from the University of Trento, Italy and a large number of scientific awards, mainly from the IEEE communications society.

Nasir Ghani is a professor at the University of South Florida. His research interest include topics, such as cyberinfrastructure, cloud computing, and power grids. He is a senior member of the IEEE.

Samee U. Khan is an associate professor at the North Dakota State University. His research interest include topics, such as sustainable computing, social networking, and reliability. He is a senior member of the IEEE, and a fellow of IET and BCS.

Rajkumar Buyya is a professor of computer science and software engineering, future fellow of the Australian Research Council, and the director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is a fellow of the IEEE