Emulation as a Service (EaaS): A Plug-n-Play Framework for Benchmarking Network Analytics

Garima Mishra, Hemant Kumar Rath, Shameemraj M Nadaf

TCS Research & Innovation, India.
Email:{garima.mishra2, hemant.rath sm.nadaf}@tcs.com

***Abstract*—Real-time data generation and collection to analyse the network performance is difficult for large-scale networks having limited accessibility. In this paper we propose a framework which can provide realistic si/e-mulations, and generate synthetic data closer to real-time data that replaces the traditionally used deterministic and probabilistic models. This framework uses an emulation based platform to replicate real network scenarios. The emulator acts as a base layer with necessary APIs to enable customized inclusion of analytics services in a plug-and-play manner through the framework. This framework can be used to acquire data required for different Machine Learning (ML) models in order to reduce costly and time-consuming data collection effort in network analytics.**

I. Introduction

Simulation and Emulation are two crucial techniques used extensively today to imitate the real world operations. These techniques are used to understand the would be scenarios, conduct performance analysis, demonstrate capabilities and design new policies/methods which can be used in the real world. Network si/e-mulation is a proven technique which is being used today by both academia and industries for various purposes. While simulated networks are scalable and do not involve deployment cost, they are not suitable for any real-time analysis. Testbeds in other hand are used to evaluate any system in real-time, but are quite expensive and time consuming. Therefore, emulation has emerged as a hybrid solution of simulation and testbeds. In an emulated network, real devices are integrated with the simulated network and analysis is performed in real-time. Though emulation has become a critical method for verification and validation, achieving interoperability between emulators and ensuring credibility of results currently require significant efforts.

Traditional emulation paradigm is highly dependent on the industry partners as majority of the industries host vendor dependent devices and emulators. They require a significant investment on CAPEX and OPEX. However with the emergence of a wide range of Cloud services, Software Defined solutions and Virtualization, end-users can now interact remotely with the devices and emulators using (web-)clients or services. The web service access offers a chance to combine both remote emulation and cloud services into an integrated system. It allows users to connect to different emulators on a need basis and create services. The service based approach allows users to experiment on composable emulation environments which can be deployed and executed on demand basis. Further, emulation capabilities can be exposed as Application Program Interfaces (APIs) to the users for remote utilization. However the interoperability between the existing set of network devices and different emulators is a challenging task. Today, there is not a single framework available (to the best of our knowledge) which provides a common interface between users and different emulators. Hence, there is a need for an interface which operates independent of integrated emulators and network devices.

Keeping this in mind we propose an "Emulation as a Service (EaaS)" framework, that aims to provide remote users with discoverable services that are readily available on demand. This increases its operational benefits and supports integration of multiple simulated and real systems into a single unified emulation framework. The EaaS framework offers an agnostic interface which operates independent of integrated emulators and network devices. The proposed EaaS framework follows Service Oriented Architecture (SOA) [1], integrates reconfigurable modular components, viz., user interface layer, parsers, emulators, data collectors etc. The SOA based framework provides a layered abstraction for discovery, composition and execution of various services. The SOA approach is model-driven and can adapt reusable developments to emphasize on collaborative processes. The proposed framework provides plug-and-play services to a user. The emulator acts as a base layer with necessary APIs to enable customized inclusion of analytics module. The framework is used to automate the complete end-to-end emulation process.

The proposed framework helps a user to create a scenario specific simulation script, generates data imitating a real network and collects data in a specific format. It dynamically creates the necessary data flow paths to allow distributed data collection and storage in the database. The key contributions of this work are as follows:

1) The proposed EaaS framework is based on the SOA-Reference Architecture (SOA-RA). It introduces emulation functionality by adding a new layer called *Emulation Environment* layer at the bottom. This layer interfaces between real devices and the system software to exchange real-time data to-and-from the emulation environment.
2) The EaaS framework is agnostic to the choice of emulator; provides an interface layer to integrate different

978–1–6654–5136–9/22/$31.00 © 2022 IEEE

2022 National Conference on Communications (NCC) | 978-1-6654-5136-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/NCC55593.2022.9806721

si/e-mulators with the real hardware devices.
3) It supports plug-and-play services to a user; emulator acts as a base layer with necessary APIs to enable customized inclusion of analytics modules and to benchmark analytics performance.
4) The emulation results obtained for a smaller network can be extrapolated by the scalability module along with the data received from the trained Machine Learning (ML) model to predict the performance of a larger network.

The rest of the paper is organized as follows: Section II presents the related work and motivation behind this contribution. Section III presents the system model. The framework implementation detail is presented in Section IV-B. We conclude the paper in Section VI.

II. Literature Survey

Modeling and Simulation as a Service (MSaaS) has attracted many researchers recently [2]. The definition of MSaaS, its architecture and deployment strategies are surveyed in [3]. Other service types such as, Network as a Service (NaaS) [4] and [5], Trust as a Service (TaaS) [6] and Authorization as a Service [7], etc., are also introduced in literature. These are derivations of Platform as a Service (PaaS) and Software as a Service (SaaS) in various combinations and forms. MSaaS can also be perceived as one of these derivatives.

The MSaaS framework proposed by North Atlantic Treaty Organization-Modeling and Simulation Group (NATO MSG)-136 [8] is a permanent service and cloud based Modeling and Simulation (M&S) ecosystem. A reference architecture is presented in [9] and [10]. The structure of the reference architecture is influenced by the open group SOA-RA where as the building blocks and patterns of the architecture are backed by the NATO C3 Taxonomy.

The Open Cloud Ecosystem Application (OCEAN) project [11] is a MSaaS platform which is developed under a technical agreement with the NATO Modeling and Simulation Center of Excellence (M&S CoE). The main objective of this project is to provide an experimentation environment for the cloud based technology, where it is possible to consume the available MSaaS services and/or deploy new M&S services.

A cloud based Simulation-as-a Service to address the challenges faced in cloud based computing is presented in [12], [13]. Authors of [12] demonstrate how lightweight solutions using Linux containers (e.g., Docker) are better suited to support such services instead of heavyweight hypervisor based solutions. Empirical results validating their claims are presented in the context of few case studies. The paper [13] proposes a Software as a Service (SaaS) model that provides access to applications hosted in a cloud environment, allowing users to use services at low cost and scale them as needed.

TotalSim [14] provides simulation as a service in aerospace, automotive and manufacturing domains. *SIMULIA* [15] of Dassault Systems helps in evaluating the performance of the products before developing the final prototypes. The Santa Clara based Ciespace adopts and delivers a SaaS based Computer-Aided Engineering (CAE) platform for advanced mechanical engineering design and analysis.

Defence and manufacturing industry have been extensively using the M&S approach and their applications have always been at the cutting edge of computing and networking. To the best of our knowledge, none of the previous works have addressed the problem of providing a single platform for emulation as a service in the network and communication domain - to analyse critical failures that can be identified and corrected before the physical prototype. Prior arts also do not address the interoperability between different emulatiors; some of them rather just propose an architecture to interact and provide bidirectional data feeds for data mining tools for actionable analysis. Our proposed EaaS framework is not a logical extension of the prior art. It provides a common Interface Layer between a user and an emulation tool, and automates the complete end-to-end emulation process through a single user request. In the following section, we explain a reference architecture for accommodating hardware devices and define a suitable framework for exposing emulation capabilities as services.

III. EaaS: Architecture

The EaaS framework is modeled by following the key features of layered Open Group SOA-RA [1]. The main design and deployment features of this framework are (i) **Generic**: It is a generic solution agnostic to different emulators and hardware devices, (ii) **Modular**: It comprises of modular building blocks which can be separated and recombined and (iii) **Scalable**: It allows users to extend the capabilities of a specific module or addition of blocks with other attributes.

A. The Architecture Concepts

As illustrated in Fig. 1, the EaaS framework is decomposed into multiple layers with different functional capabilities. Some of the layers are cross-cutting and have common functionalities that spans across the layers. The horizontal layers support the base functionalities of the architecture. In an emulator, along with the simulated nodes, one or more emulated nodes are to be created which can help real devices such as sensors, access points and stations etc., to participate in simulation and to send/ receive packets over the network in real-time. However the current SOA architecture does not have any support for this kind of dynamic device configuration and integration. Therefore we have added two new layers- namely an **Emulation Environment Layer** and an **Orchestration Layer** in the above mentioned SOA architecture. The Emulation Environment Layer in particular provides ability to create and configure all virtual interfaces, bridges and routes that are needed to connect real devices with the simulated nodes. The Orchestration Layer takes care of emulator/ container orchestration.

The proposed EaaS framework is agnostic to emulators. Hence there is a need to coordinate among multiple emulators through a central entity or a mediator. The EaaS framework also follows a modular architecture and needs collaboration

among different modules. The Orchestration Layer takes care of deployment, maintenance and synchronization activities for different emulators. It also arranges multiple tasks in different modules to optimize a workflow. The capabilities and the functionalities of each layer are described in Table I. It provides a summary on how the capabilities of each layer (listed in Column 2) are achieved through different functional implementation and deployment approaches.

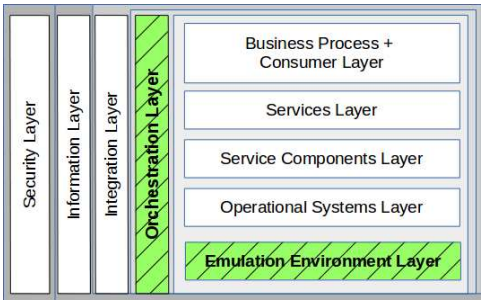

Fig. 1: EaaS Architecture Layers

TABLE I: Layers and architecture building blocks

Layers	Capabilities	EaaS Building Blocks
Emulation Environment Layer	Interactions between the hardware and the system software to exchange the real-time packets	Interface between the Emulated network nodes and the Simulated network nodes.
Operational Systems Layer	Supports the capabilities needed for running/executing all software	EaaS service configuration and Emulation server configuration.
Service Components Layer	Provides the implementation or the realization for services and their operations	Emulation Client layer and interface layer implementation.
Services Layer	Contains the service descriptions for business capabilities	Emulation specific services (separate APIs for start, stop, update, delete, execute).
Business Process + Consumer Layer	Covers process representation and composition to aggregate services as a sequencing process aligned with business goals	Executing different scenarios, Database generation.
Orchestration Layer	Supports collaboration between processes	Marathon can be used to orchestrate different types of Emulation services.
Integration Layer	Provides the capability to transport service requests from the service requester	Passing list of operational and configuration parameters, Socket based communication between emulation client and server.
Information Layer	Store and retrieve data in a uniform format	parameter translators and parsers
Security Layer	Processes policy definition, management and service lifecycle	Authentication and authorization of different category of users

B. EaaS Deployment Model

Emulation services are made accessible to the remote users via APIs which are implemented as RESTful web services. As per the user service request, resources such as multiple service instances or data storage are provisioned automatically. The block diagram as shown in Fig. 2 illustrates the deployment architecture of our proposed framework; major modules of the same are described as follows:

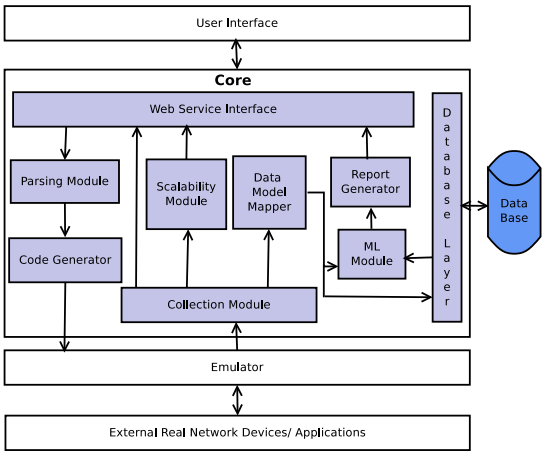

Fig. 2: Block Diagram for EaaS Framework

User Interface: Receives the user service requests and configuration parameters of a network on this interface. It authenticates the user, forwards the requests to the framework core, and provides a prescriptive report back to the user to enhance the network performance. In our current deployment user requests are provided in JavaScript Object Notation (JSON) format; other formats can also be used in practice.

Emulator: It is configured to execute the emulation script to get emulated nodes. Based on the inputs provided by the user, the simulated network environment is created. Along with the simulated nodes, one or more emulated nodes are also created to integrate real devices into the emulation, and allow them to send and receive packets over the network in real-time.

Core: It is an intelligent network orchestration module; interacts with the user layer through various web services provided by the Web Interface Layer. These services are used for remote accessing of EaaS functionalities and to exchange data between EaaS web portal and other core modules. The core module has the following building blocks:

1) **Parsing Module:** It parses the received user request to define a network topology with the received configuration and to set up the network environment with the simulation parameters. In this work, we emulate a WiFi network with configuration parameters related to wireless mode, radio mode, desired channel frequencies, and simulation environment parameters such as: indoor/ outdoor scenario, path-loss model, mobility models etc.
2) **Code Generator:** It is configured to generate an emulation script based on the defined network topology. In this work, an emulation script is generated in C++, which is well understood by the integrated emulator.
3) **Data Collection Module:** It is configured to collect emulation logs in a predefined format. In this work, the collected logs are in a time series output and these are added in a Comma-Separated Value (CSV) file format. Any delimited file can also be used that has a field separator to separate each data point.
4) **Data Model Mapper Module:** It takes the CSV logs from the Data Collection Module, translates it into a

unified format, generates network performance report and stores it in the database.

5) **Machine Learning (ML) Module:** The historical data retrieved from the database are passed over the ML module for training. Data analysis and learning based algorithms are used to predict network health by utilizing important network Key Performance Indicators (KPIs) as key features for these ML models.
6) **Scalability Module:** The emulation results obtained for a smaller network can be extrapolated by the scalability module along with the information received from the trained ML model. The system further configures virtual interfaces, bridges, and routes for connecting real devices to the emulator and finally integrates emulated nodes with real hardware devices.

Next section provides the implementation details of the proposed EaaS framework.

IV. EaaS Framework Development

We consider Network Simulator-3 (NS3) [16], an open source discrete event simulator to demonstrate the EaaS framework capabilities for network monitoring, performance analysis and optimum policy building. Since emulation services can be invoked remotely and executed on demand, a user needs to first configure the emulation server to establish the connection. Post that, users can upload the desired configuration parameters for creating a simulation environment and integrating emulated nodes with the real devices. We use OSGi service platform for dynamic reconfiguration of the distributed environment. OSGi is a Java Framework which allows to develop and deploy modularised software program and libraries. The run-time dynamics and service-oriented concepts of OSGi enable plug-and-play capabilities for distributed simulation. A modular software component in OSGi is called bundle or plug-in, that can be deployed in a container and has an independent lifecycle, i.e., it can be started, stopped and removed independently. We describe the summary of an application lifecycle as follows:

A. Application Lifecycle

To execute an application and its services in an OSGi environment we build an OSGi bundle, define the application entry point and construct the service object. Once the bundle is started, the service is registered on the platform. The class diagram of the NS3 service implementation is shown in Fig. 3. We have created a dynamic collaborative emulator service model (*Ns3Model*) which is integrated with the OSGi life cycle. This service model is a Java object which is registered under different Java interfaces (*AnalysisNs3Service*) with the service registry. The OSGi bundle takes care of registering the emulator services and retrieving them while discovery.

1) **Bundle Entry Point**: We implement an *Activator* class which provides entry points for a bundle. A bundle is started and stopped by invoking the `start()` and `stop()` methods respectively by the OSGi platform.
2) **Building a Bundle**: We then leverage the *maven-bundle-plugin* to package the *NS3Service* class as an OSGi bundle with complete application.
3) **Service Registration**: The Container specification defines a dependency inclusion framework for OSGi. It is designed to deal with the dynamic nature of OSGi, where services can become available and unavailable at any time. The `XML` files that define and describe various components of an application are key to the programming model. We declare the *NS3Service* using *Blueprint XML* (*AnalysisNS3Service.xml*).
4) **Emulation Service**: We implement a service interface and different services are interfaced to upload configuration files (`uploadFileData`), to retrieve all the active NS3 simulation instances (`getAllNs3Config`) and to get NS3 configuration details by its instance id (`getNs3ConfigById`).
5) **Emulation Client**: We develop an emulation client (*NS3Client*) which communicates with the NS3 server (emulator) identified based on the received NS3 socket address (Server IP address and Port number) and transfers the received configuration file to the remote NS3 server using Socket communication.

B. Web Services

The proposed framework provides various plug-and-play services to a user. The emulator acts as a base layer with necessary APIs to enable customized inclusion of analytics modules. It creates a network environment specific simulation script; generates data imitating real network; translates/maps the data in a specific format; and finally stores it in a database. It creates the necessary data flow paths to allow distributed data collection and storage by the individual nodes. It addresses the problem of scarcity of training data and inability to test the analytics algorithms due to lack of testbed. The current deployment of this framework offers the following web services: (i) Generation of user specific network simulation scripts, (ii) Customizable data collection points, (iii) Insertion of the data analysis modules based on user specific needs and (iv) Creation of the run-time feedback loop to the base framework from analytics function. In the following section our aim is to provide the benchmarking service of EaaS.

V. EaaS: Benchmarking

The EaaS framework offers many advantages including stand-alone and remote access to emulators; a large number of users can access emulation services from anywhere and anytime at a reduced cost. Therefore, benchmarking of the proposed framework is very important. This section explains the procedure of benchmarking the emulation data by evaluating the accuracy of the collected data and then modeling of the learning based algorithms.

A. Accuracy of the Emulation Results

The emulation result accuracy depends on the calibration of emulation models, which is achieved by tuning some of the simulation environment parameters. The sensitivity analysis

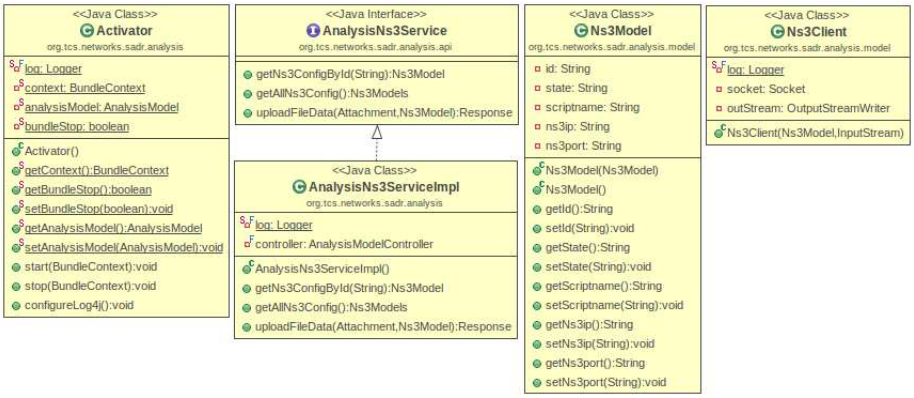

Fig. 3: Class diagram of NS3 Service Architecture

is applied over the estimated simulation results in order to specify more accurate values for those input parameters that have greater impact on the output. A user needs to define affecting parameters and set default values for the variables which have no or the least impact on the output. Calibration of the simulation model is to minimize the difference between the predicted and the measured simulation results. It is implemented as part of the data collection process to minimize the error between real and emulated results.

The sensitivity analysis technique is used to estimate the error in the simulation as a consequence of input uncertainties. Fig. 4 explains the process of analysing simulation outcome and re-calculating the parameters which are the source of uncertainty in the input. The extent to which a variable's input affects the output is the sensitivity of the output to that input. By taking the sample of the model at multiple points of the input space X, it becomes possible to fit a much simpler emulator model $\eta(X)$, such that $\eta(X) \approx f(X)$; where $f(X)$ is the testbed output. The emulator is modeled such that the approximation remains within an acceptable error margin (ε). During this process, the emulator parameters are adjusted until they resemble the testbed.

To assess emulator accuracy, we compare the results of the emulation with those of a small laboratory-based testbed. For this, we set-up a simple WiFi based network environment. We consider a lab-based testbed with a single Access Point (AP) connected with two mobile users (STAs). An emulation setup that uses NS3 is also created. The configuration parameters (transmission power, operating channel, radio mode, etc.,) of the WiFi devices are kept same for both the emulation and the testbed. The indoor path loss model [17] is used to design a accurate indoor communication network. The results presented in Fig.5 and Fig.6 demonstrate the comparison between emulation and testbed output. From these figures, we observe that in both, emulation and the testbed, the mean Received Signal Strength Indicator (RSSI) for a moving object is around -40 dBm. To bring the emulation results closer to the testbed results, we calibrate the emulation model by tuning the indoor propagation model (`ns3::PropagationLossModel`) parameters used in the simulator. Due to the real-time testbed limitations, we have only demonstrated accuracy check and data collection of RSSI but the framework is not only limited to the signal strength rather it can be utilized for other parameters like, throughput, latency, bandwidth utilization, etc.

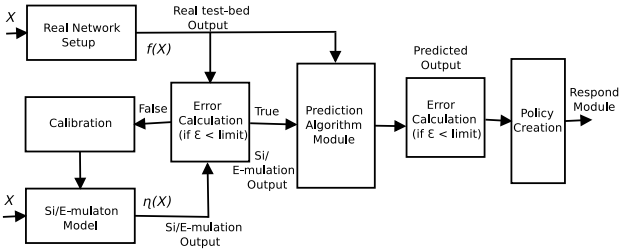

Fig. 4: Block Diagram: Error Estimation and Model calibration

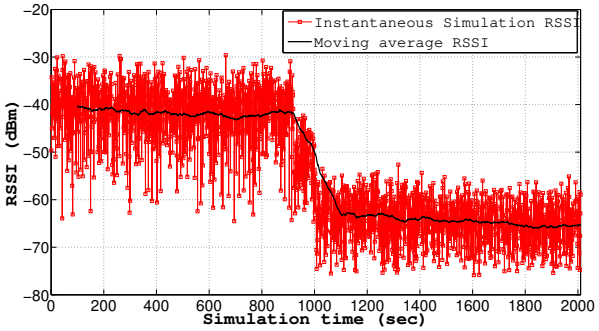

Fig. 5: RSSI Variations for a Moving Object- Emulation

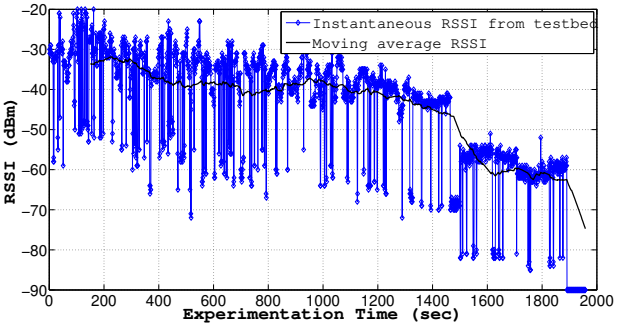

Fig. 6: RSSI Variations for a Moving Object- Testbed

B. Benchmarking of Network Analytics with Si/E-mulation

In practice the scarcity of real-time network data hinders benchmarking of various Machine Learning (ML) models to

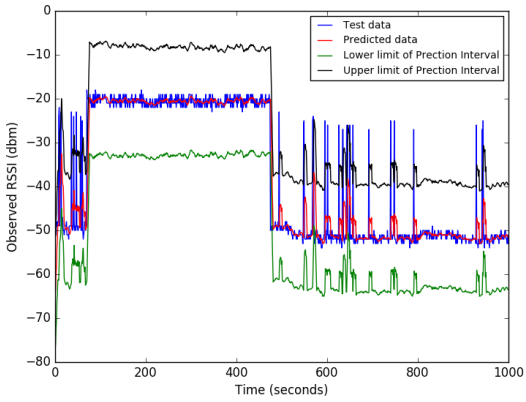

Fig. 7: Prediction Accuracy

be used for network provisioning and management. This challenge is handled by the proposed EAAS framework. Further, this framework not only provides use-case specific network simulations, but also allows network analytics services such as predictions, anomaly detection etc., to be used as plug-n-play services. This framework helps in generating synthetic data (log traces) mimicking a real network or a testbed; synthetic data is further used for network analytics algorithms. The emulation model parameters are tuned to generate synthetic data close to the testbed data for higher prediction accuracy. The ML models can be trained over both the historical data stored in a database as well as the synthetic data generated in real-time. These trained models respond to new data, which are used for data predictions, prescriptive report generation and new policy creation. The generated reports are used to provision the network resources and to improve its performance. For verification and testing of these ML models, real-time or testbed data, if any, are used. In this deployment, we use EaaS generated synthetic data to train the ML models and testbed data to verify and test the ML models.

In this paper, we predict the quality of a wireless link by estimating its Received Signal Strength (RSSI) values. For this prediction process, we propose to use the synthetic data set to train the ML model and the testing process is performed over the testbed data. Training is performed with sliding window based supervised learning. We plot both, the observed RSSI data of the testbed and the predicted RSSI data in Fig. 7. Through this figure, we demonstrate the possible use of synthetic data generated by a huge number of simulations to train ML models and use it with testbed data for online verification and testing. The performance of ML models are restricted by both amount and quality of training data. Hence, the emulation model parameters are pre-tuned to generate synthetic data which is close to the real-time data for achieving higher prediction accuracy. We observe from the results that model predicts the RSSI of the wireless link with small mean square error of 0.64.

VI. Conclusion

There is an increasing demand of emulators which replace costly or impractical testbed. The 'Emulation as a Service' is a plug-n-play framework to benchmark network analytics algorithms. In order to provide a wide range of remote emulation services, the distributed architecture of EaaS framework is helpful to deploy and execute emulation environment on-demand basis. It provides a support to integrate multiple simulation tools and real devices over a common emulation framework. It further offers an agnostic interface which operates independent of integrated emulators and network devices. The EaaS framework addresses the problem of scarcity of training data for different networking domains and inability to test the analytics algorithms due to lack of test bench.

Acknowledgment

We thank Dr. Samar Shailendra for his valuable suggestions during the planning and development phase of this work.

References

[1] "The Open Group, SOA Reference Architecture Technical Standard, doc. no. C119 (2011)," Tech. Rep.

[2] E. Cayirci and Chunming Rong, "Intercloud for simulation federations," in *International Conference on High Performance Computing Simulation*, July 2011, pp. 397–404.

[3] E. Cayirci, "Modeling and simulation as a cloud service: A survey," in *Winter Simulations Conference (WSC)*, Dec 2013, pp. 389–400.

[4] Q. Duan, "Network-as-a-Service in Software-Defined Networks for end-to-end QoS provisioning," in *23rd Wireless and Optical Communication Conference (WOCC)*, 2014, pp. 1–5.

[5] R. Pries, H.-J. Morper, N. Galambosi, and M. Jarschel, "Network as a Service - A Demo on 5G Network Slicing," in *28th International Teletraffic Congress (ITC 28)*, vol. 01, 2016, pp. 209–211.

[6] N. T.H. and S. Q.Z, *Trust as a Service: A Framework for Trust Management in Cloud Environments*. Lecture Notes in Computer Science, Springer, 2011, vol. 6997, pp. 314–321.

[7] A. Alsubaih, A. Hafez, and K. Alghathbar, "Authorization as a Service in Cloud Environments," in *International Conference on Cloud and Green Computing*, 2013, pp. 487–493.

[8] S. Wang and G. Wainer, "Modeling and Simulation as a Service Architecture for Deploying Resources in the Cloud," *International Journal of Modeling, Simulation, and Scientific Computing*, vol. 7, 01 2016.

[9] D. Procházka and J. Hodický, "Modelling and simulation as a service and concept development and experimentation," in *International Conference on Military Technologies (ICMT)*, May 2017, pp. 721–727.

[10] J. Hannay and T. van den Berg, "The NATO MSG-136 Reference Architecture for M&S as a Service," 10 2017.

[11] M.-M. Neag, L. Ispas, and C. Grindeanu, "The Comprehensive Approach Concept in Multinational Operations," *Land Forces Academy Review*, vol. 22, 12 2017.

[12] S. Shekhar, H. Abdel-Aziz, M. Walker, F. Çağlar, A. Gokhale, and X. Koutsoukos, "A simulation as a service cloud middleware," *Annals of Telecommunications*, vol. 71, pp. 93–108, 01 2016.

[13] T. Bitterman, P. Calyam, A. Berryman, D. Hudak, L. Li, A. Chalker, S. Gordon, D. Zhang, D. Cai, C. Lee, and R. Ramnath, "Simulation as a service (SMaaS): a cloud-based framework to support the educational use of scientific software," *Int. J. of Cloud Computing*, vol. 3, pp. 177–190, 01 2014.

[14] "Advanced CFD Technology Expertly Applied to Solve Real-World Problems," https://www.totalsim.us, accessed: 2020-10-28.

[15] "Discover SIMULIA," https://www.3ds.com/products-services/simulia/, accessed: 2020-10-28.

[16] The ns-3 network simulaton, http://www.nsnam.org/.

[17] H. K. Rath, S. Timmadasari, B. Panigrahi, and A. Simha, "Realistic indoor path loss modeling for regular WiFi operations in India," in *Twenty-third National Conference on Communications (NCC)*, March 2017, pp. 1–6.