Kaala 2.0: Scalable IoT/NextG System Simulator

Udhaya Kumar Dayalan, Timothy J. Salo, Rostand A. K. Fezeu, and Zhi-Li Zhang

Abstract

The IoT world is evolving with the latest technology trends, like edge computing, augmented & virtual reality, machine learning, robotics, and 5G. With the digital transformation happening in Industry 4.0, many industries are moving toward private 5G networks. There are massive number (hundreds to thousands) of IoT devices in a single factory depending on the scale of the industry and these factories consists of critical IoT devices, like fire or gas sensors which need to operate reliably with less latency. To efficiently realize the capabilities, such as ultra reliable low latency communications (URLLC), enhanced mobile broadband (eMBB), and massive machine-type communications (mMTC) offered by 5G, the next generation IoT devices/applications need a paradigm shift in their design and need to be evaluated under simulation using 5G networks before getting deployed in the real-world. However, many IoT simulators run in isolation and do not interface with real-world IoT cloud systems or support 5G networks. This isolation makes it difficult to design, develop and evaluate IoT applications for industrial automation systems and for experiments to fully replicate the diversity that exists in end-to-end, real-world systems using 5G networks. Kaala 2.0 is the first scalable, hybrid, end-to-end IoT and NextG system simulator that can integrate with real-world IoT cloud services through simulated or real-world 5G networks. Kaala 2.0 is intended to bridge the gap between IoT simulation experiments and the real world using 5G networks. The simulator can interact with cloud IoT services, such as those offered by Amazon, Microsoft, and Google. Depending on the configuration, Kaala 2.0 supports simulation of User Equipment (UE), 5G Radio Access Network (RAN) and 5G Core and at the same time support real-world User Equipment (UE), 5G Radio Access Network (RAN) and 5G Core. Kaala 2.0 can simulate many diverse IoT devices to evaluate mMTC, simulate events that may simultaneously affect several sensors to evaluate URLLC and finally simulate large amount of data to evaluate eMBB.

Introduction

The IoT world is evolving with the latest technology trends like edge computing, augmented & virtual reality, machine learning, robotics and 5G. But still there is less business productivity due to the slower technical adoption in the industrial automation system. There is a tremendous need for autonomous networks in the manufacturing industry to increase productivity and allow communication between people, devices, and sensors. And there are massive number (hundreds to thousands) of IoT devices in a single factory depending on the scale of the industry. These factories consist of critical IoT devices like fire or gas sensors which need to operate reliably with less latency. But the existing wired and/or wireless networks are struggling to fulfil the computational and resources for operations demand of the emerging technologies. To address these needs of the industries with digital transformation happening in Industry 4.0, the evolution of private 5G and 5G standards opens bigger opportunities. The network architecture of 5G is designed for 3 key services in mind: mMTC, eMBB, and URLLC. The 5G promises and new capabilities trigger an important question: can 5G help manufacturing industry to catch-up with the technology changes shown in Fig. 1?

Cloud IoT service vendors, such as AWS [2], Azure [3], and Google [4], all aim to build their own IoT ecosystems, which currently do not interoperate with each other. According to the UNIFY IoT project, more than 360 IoT companies exist today [5]. While there are industry-led efforts to ensure interoperability between cloud IoT services, non-interoperable cloud IoT services are likely to remain the rule.

Motivations: Need and Limitations of Simulation Frameworks

Expensive and Cumbersome Real Setup: Prototyping, testing, and evaluating new IoT devices and systems can be expensive. Beyond the cost of the devices themselves, setting up, configuring, and maintaining a diverse collection of physical IoT devices can quickly become time-consuming, cumbersome, unwieldy, and expensive. As a result, simulators are often used to test and evaluate new product ideas and designs early in the development process. Since cloud services have become a critical component of many IoT systems, it is beneficial that these simulations be able to evaluate end-to-end systems, from the IoT devices to the IoT cloud services.

Vendor Locked: However, simulating end-to-end IoT systems as shown in Fig. 2 is complicated by the vendor specific nature of these cloud IoT services. Each vendor requires IoT devices to

Digital Object Identifier:
10.1109/MNET.002.2200498

Udhaya Kumar Dayalan is with Trane Technologies, USA, and University of Minnesota, USA; Timothy J. Salo, Rostand A.K. Fezeu, and Zhi-Li Zhang are with the University of Minnesota, USA.

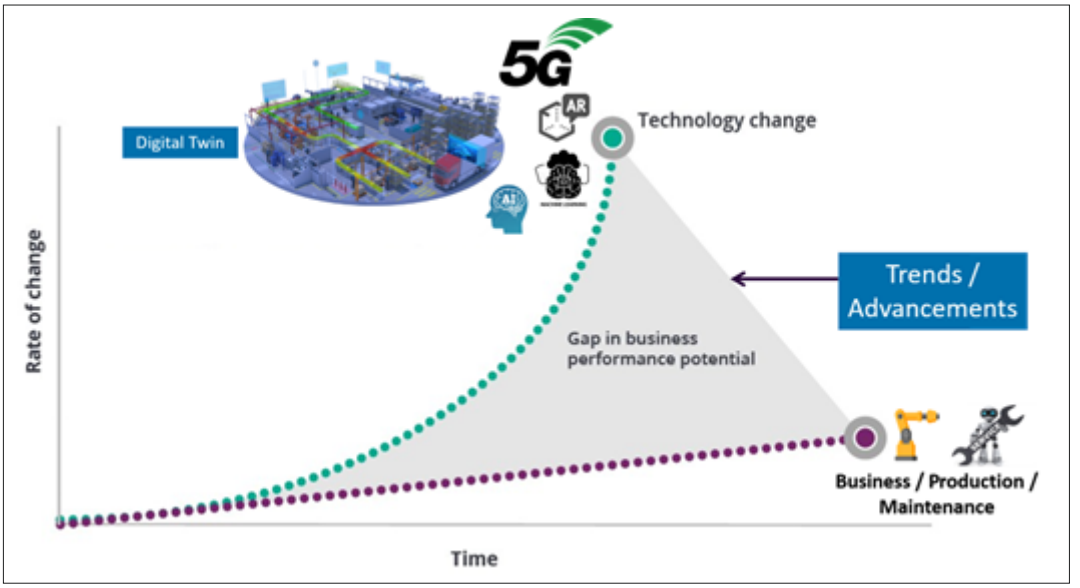

FIGURE 1. Gaps in manufacturing industry and trends [1].

implement a different set of protocols to authenticate and collect data from these devices. As a result, IoT devices are often locked into one cloud IoT service provider [6, 7].

Limited Capabilities of Existing Simulation Frameworks: An alternative to waiting until physical devices have been developed and constructed, is to use IoT simulators to test and evaluate prospective IoT devices, systems, and designs. If these simulators and experiments are designed properly, simulation can significantly reduce the gap between proof-of- concept (PoC) implementations and real-world deployments [8]. Unfortunately, existing IoT simulators are limited in their capabilities and scopes. Many simulators are designed to run on a single laptop, desktop, or a server, and are therefore poorly positioned for large-scale simulations that require significant computational power. Most of simulators fail to capture the large variety and diversity of IoT devices that exist today. For example, many are tailored to simulating only small sensors with low bandwidth requirements, ignoring a variety of IoT devices (e.g., surveillance cameras and autonomous vehicles) that consume large amounts of network bandwidth and require real-time cloud connectivity. Perhaps more importantly, existing IoT simulators operate in isolation: they interface with only a limited number of types of IoT devices and cannot be integrated with existing cloud services.

In short, existing IoT simulators cannot be used to effectively test and evaluate prototype IoT systems, especially those that require computationally intensive subsystems, such as machine learning algorithms, Cloud IoT services, or IoT control mechanisms running on edge computing facilities. Chernyshev *et al.* [8] in a recent survey highlighted that an all-in-one simulator capable of supporting an end-to-end IoT service with 5G networks has yet to be developed. New simulation tools and test-and-evaluation environments are sorely needed for unit testing and for the systematic evaluation of intelligent IoT services that include diverse, integrated devices, edge computing and cloud computing components.

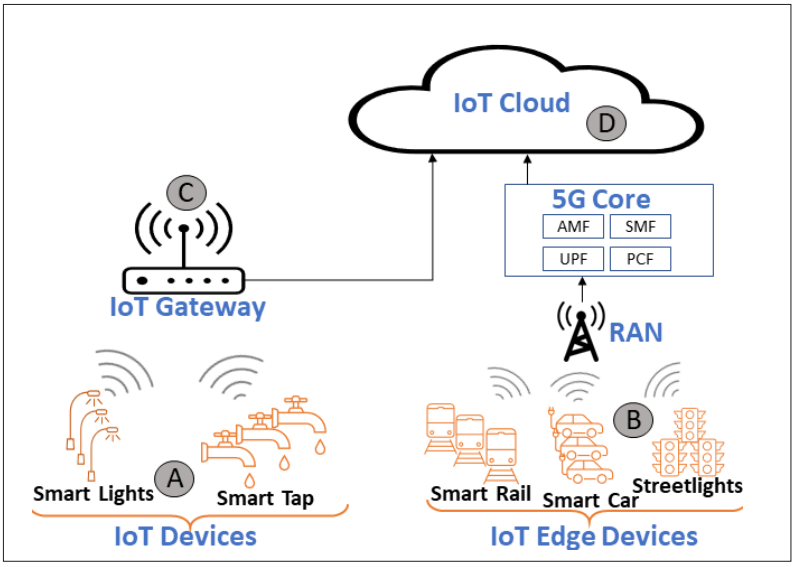

FIGURE 2. IoT (Edge) Devices, IoT Gateway, and IoT Cloud.

The Scope of This Article

We present *Kaala 2.0*—a modeling, simulation, and emulation platform that can specify IoT devices of various types, from low-powered sensors to smart IoT devices requiring high bandwidth, such as IoT devices that anticipate emerging 5G networks. Kaala 2.0 is an extension of Kaala [9] and simulates UE, RAN and 5G Core at the same time connect to real-work UE, RAN, and 5G Core. In addition to simulating IoT devices, an important feature of Kaala 2.0 is its ability to interface and connect with real-world cloud IoT services in an integrated fashion. The initial version of Kaala 2.0 can use Amazon AWS [2], Microsoft Azure [3] and Google [4] IoT cloud services. The main design goal of Kaala 2.0 is to help researchers and practitioners to prototype various IoT scenarios, including those that use high bandwidth 5G services, generate massive amounts of data, and to help bridge the gap that exists between simulators and real-world system. The major contributions are summarized below:

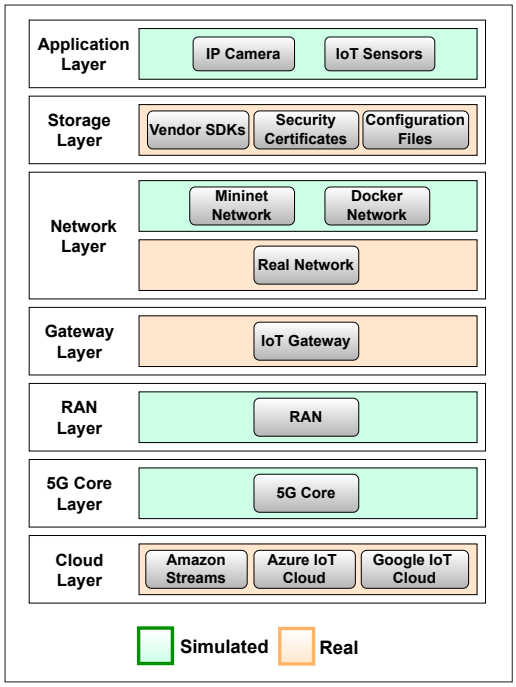

FIGURE 3. Kaala 2.0 layered architecture.

- We present Kaala 2.0: An IoT modeling and simulation platform that can specify IoT devices of various types to communicate with real-world cloud IoT systems, like AWS, Amazon, Google, and many other IoT Cloud platforms as case studies through simulated or real-world 5G networks.
- Kaala 2.0 is a scenario based IoT simulator capable of mimicking various IoT scenarios such as "fire in a room or building" to evaluate URLLC service of 5G and "5G network capable data generation (including 4K/8K video IP cameras)" scenarios.
- Kaala 2.0 can simulate massive number of IoT devices to evaluate mMTC service of 5G.
- Kaala 2.0 can generate massive amounts of IoT data for prototyping data intensive IoT applications to evaluate eMBB service of 5G.

The rest of the article is organized as follows: We motivate the need for new simulation framework using case studies. And define the multi-layered paradigm of Kaala 2.0. The design and implementation of Kaala 2.0 are presented below, respectively. Kaala 2.0 is simulated and evaluated with a conclusion.

Case Studies: Need for a Better Simulation Framework

We use three case studies to argue the need for a better simulation framework, while discussing their challenges.

Ability to Interact with Real Systems using 5G Networks

Emergent IoT applications are extremely complex and operate in a very diverse IoT world. These are not just the smart speakers, smart thermostats, or smart door locks in our homes, but also sensors used in domains like in the oil, gas, noise, and automobile industries. These applications support different systems connected through IoT using different net- works. There are no differences in data flow between real and simulated IoT devices. And there are less differences when using real and simulated networks. Researchers use simulators to study, prototype, test, and evaluate new IoT and 5G concepts. However, these simulators fail to reflect the complexity that exists in real IoT environments. For example, current IoT simulators do not simulate 5G networks. The IoT simulator provided by AWS [10] can only simulate one type of IoT device. It simulates hard-coded IoT messages and does not simulate the network characteristics of 5G networks along with massive IoT data. To the best of our knowledge, current IoT simulators operate in isolation and do not interact with real cloud IoT system or support 5G networks, failing to reflect the complexity present in the IoT world. Kaala 2.0 is intended to remedy these deficiencies. Kaala 2.0 simulates several vendors specific IoT devices, (including those from AWS, Google, and Azure), and can connect to and communicate with real-world cloud IoT systems provided by these vendors using 5G networks. Kaala 2.0 connects simulated IoT devices with real servers (within the complex IoT world) using 5G networks, so that services provided by 5G can be used, validated, and verified. For instance, Kaala 2.0 connects simulated IP cameras, temperature sensors, humidity sensors, and flame sensors to the Amazon's Kinetic video streams using simulated or real 5G networks and builds logic around these sensors to simulate a fire event.

Scenario-Based Data/Event Simulation

We use a *fire-in-a-building* event to make the case for the need of a scenario-based data generation and 5G service. IoT data generated by IoT simulators are hard-coded [11]. A more realistic approach might be to generate sensor data based on a distribution or based on the actual behavior of the sensor. Consider a *fire-in-a-room* scenario. When there is a fire in the room, the temperature in the room increases and the temperature sensor will report a higher value than usual. The smoke sensor will detect the smoke in the room and send the smoke alarm. The humidity in the room increases and the humidity sensor will be sending the updated humidity value. Some sensors might malfunction, burn, or lose connection because of the fire. Thus, relying on a few sensors' data values will be problematic. And the critical data need to be prioritized for cloud analysis within the expected latency of each application to realize the URLLC service offered by 5G even when massive IoT devices are connected to the 5G network.

High-Bandwidth Data Generation

IoT simulators ought to support next-generation network technologies such as 5G. With higher 5G throughput, next generation IoT applications should seamlessly adapt to 5G. However, this is not the case due to lack of tools which foster the design, development, and deployment of 5G-capable applications both on the client and server side. For instance, a video streaming service provider like YouTube or Netflix have millions of users watching videos. The throughput will depend on the quality of the video. Recently 8K videos require significantly higher bandwidth to play a single frame compared to a 480p video

quality. For example, a video of 'X' minutes requires 119 MB for a 240p video, 1038 MB for a 1080p video, and 7284 MB for an 8K video. Thus, prototyping 5G next generation IoT applications using an IoT simulator that can support the massive data workloads seen in production environment should be addressed. Current IoT simulators do not support modeling thousands of IoT devices with large amount of data. Kaala 2.0 realizes this challenge by simulating IP cameras that supports 8K video streaming and can scale to hundreds (and even thousands) as shown later. In addition, each IoT edge device can be simulated as a UE to communicate to the NextG network.

Defining Kaala 2.0: A Multi-Layered Paradigm

Kaala 2.0 is able to integrate real and simulated devices while leveraging vendor specific SDKs to connect them to real systems in the cloud. Next, we motivate and detail each layer in Kaala 2.0's architecture as shown in Fig. 3.

Key Objectives: There are three main objectives of Kaala 2.0:

- Simulate various IoT devices, including vendor specific IoT devices and connect them to vendor's cloud IoT services.
- Simulate realistic data across all applicable devices to mimic real IoT service scenarios.
- Generate high-throughput data.

Cloud Layer: This layer is the real cloud IoT systems that Kaala 2.0 connects to. It authenticates, validates, and accepts incoming connections and IoT data from the IoT gateway. The cloud layer provides core entities for massive data transformation, data analysis and interpretation. It provides data stream processing resources and cloud services for machine learning related tasks, business integration and user management.

5G Core Layer: This layer is the simulated 5G core to which each of the simulated RAN connects to. On the other end, the 5G Core connects to the internet. This can be replaced with real-world 5G core as well in which the RAN also should be real-world 5G RAN and the UE need to have proper service to the real-world 5G provider.

RAN Layer: This layer is the simulated RAN to which the simulated UEs connect to. There can be more than one RAN and each RAN connect to one 5G core. This can be replaced with real-world 5G RAN which connects to real-world 5G core as well in which the UE need to have proper service to the real-world 5G provider.

Gateway Layer: The IoT gateway connects Kaala 2.0 to real cloud IoT systems. It runs vendor-provided SDKs, which contain the RESTFul APIs needed to connect to the cloud layer. It serves as an MQTT broker (server) to IoT (Edge) devices and is a client that connects to the cloud. The MQTT broker receives IoT data from the network layer and translates the data into vendor specific data formats via their SDKs.

Network Layer: The network layer connects all the other layers. It is a virtual network that models a real network. This layer provides the resources needed for the gateway to connect to real systems. Every instance of this layer creates a separate, isolated virtual network that can connect to real systems.

Storage Layer: The storage layer is composed of different vendor SDKs, security contexts, data storage, and configuration files. Cloud service providers support different SDKs, RESTFul APIs, data formats, security mechanisms, certificates, and keys. This layer is responsible of contacting the service providers and obtaining the updated certificates, keys, and SDKs used by the gateway layer or the application layer for authentication and validation.

Kaala 2.0 is able to integrate real and simulated devices while leveraging vendor specific SDKs to connect them to real systems in the cloud.

Application Layer: This is the layer responsible for simulation configurations, parameters tuning, IoT device configuration, network configuration, and experiment scenarios. The purpose of the application layer is to run application specific logic.

Designing Kaala 2.0: Enhanced Capabilities With NextG Support

We discuss Kaala 2.0's design goals followed by a detailed design to guide our implementation.

Design Goals

The key design goals for Kaala 2.0 are four-fold:

- Connect simulated IoT devices to real cloud IoT services using 5G networks.
- Provide extensibility of IoT device characteristics for IoT devices.
- Simulate real-time events coordinated across multiple IoT devices.
- Generate realistic data to support current and future technologies like 5G.

Below, we will discuss how Kaala 2.0 achieves these design goals.

Scenario-Based Event Simulation

The design of Kaala 2.0 supports most of the real-time scenarios. The basis of scenario-based simulation is to coordinate one or more simulated IoT devices to match values based on the scenarios at the same duration range. Kaala 2.0 supports simulation of one or more scenarios either at the same time or in sequence. The fire-in-the-room scenario is a built-in scenario in the Kaala 2.0 simulation framework. When there is a fire detected in the room, the smoke sensor detects the smoke, the door lock opens automatically, the temperature in the room increases, the humidity in the room increases as well and the IP camera in the room captures the video.

High-Bandwidth Data Simulation

Next, to simulate a high-bandwidth scenario, we design our simulated IP camera using a Real-Time Streaming Protocol (RTSP) server [12] and client. The server will listen in a port and the client will be listening in a different port. The producer of the video will be connecting to the server port and the consumer of the video will be connecting to the client port to play the video. Both the producer and consumer can be designed to run in any host, so that the traffic flows through the network. More about RTSP implementation and evaluation are discussed below.

NextG Network Support

To support NextG simulation, Kaala 2.0 is designed to support regular network connection

Features	IoTNetSim	Kaala 2.0
Vendor specific IoT devices	No	Yes
MQTT protocol broker	No	Yes
Cloud layer	Yes	No
Semi-real IoT devices	No	Yes
5G capable scenarios	No	Yes
5G RAN	No	Yes
5G core	No	Yes

TABLE 1. Comparison of Kaala 2.0 With IoTNetSim.

to the cloud as well as connecting to the cloud through RAN and 5G Core. In case of NextG, each IoT device will connect to the 5G network as a UE. The design supports connecting the IoT gateway to the RAN as a UE or each IoT device as a UE to the RAN. If each IoT device acts as a UE, then those devices won't be able to integrate with a IoT gateway, it needs to connect to the IoT cloud through the 5G Core. Once the UE is connected to the 5G Core, then it can start sending the IoT related data to the IoT cloud through the RAN and 5G Core. Each IoT device will have an option to be a regular IoT device or NextG capable device. Kaala 2.0 supports both simulated and real-world UE's. The real-world UE's can connect only to real-world RAN and the simulated UE's can connect only to simulated RAN due to the radio conditions. Because in simulations, the radio conditions are simulated as well. The real-world RAN can connect to both simulated and real-world 5G core based on the system setup.

Interacting with Real-World Systems Using 5G networks

To support connection with both simulated and real networks including 5G networks, we leverage Mininet [13] and docker [14]. Mininet has the capability to create various types of virtual networks using different types of switches and controllers, and each host will be running as a process. Docker has the capability to create virtual networks and each host will have its own container [14]. These Mininet and docker virtual devices have IP addresses assigned to them and therefore can connect to real physical networks. Both the Mininet and docker architecture supports running real application processes in the respective host instances. Using this, the simulated applications run the respective SDKs in the various hosts as processes. This architecture is highly scalable when compared (Table 1) to the architecture proposed by IoTNetSim in [11]. This is because IoTNetSim creates virtual machine for each simulated device. Moreover, in IoTNetSim the SDKs need to be installed individually in each virtual machine.

Realizing Kaala 2.0: Implementation of the Framework

Key Enabling Technologies

We used Mininet [13] and the Docker framework [14] to simulate the IoT devices. The simulator framework can simulate both generic IoT devices and vendor specific IoT device. Since we are leveraging the Mininet framework, each IoT device runs in its own process and gets dedicated network resources. We used the NodeJS version of the Mininet framework to simulate the IoT devices. And for the Docker version, each IoT device runs in its own container. The generic IoT device uses the basic MQTT client and the vendor specific IoT device uses the respective vendor specific client SDK to communicate to the IoT Gateway. The IoT Gateway information, including the authentication details required to connect to the vendor specific IoT Gateway, are passed as parameters when starting the respective vendor- specific simulated IoT devices, so each device knows which IoT Gateway they need to connect, authenticate, and communicate with. Each simulated IoT device also consists of a profile, which specifies the type of device it simulates, and the list of properties associated with the IoT device.

Configuring the End-to-End System

The simulation framework loads a configuration file during startup. This configuration file includes the list of IoT devices which needs to be simulated. The support for new IoT devices can be easily added to Kaala 2.0 by just adding a profile for the newly added IoT device. The new IoT device can either use the generic application logic which sends data periodically based on the configuration which is discussed next or implement its own application logic. Each IoT device entry in that list contains a list of properties and these properties can be easily extended for new properties. Some of the key properties include the name of the IoT device, the profile (light, HVAC, smoke sensor, etc.) of the IoT device, the type of device (generic or vendor specific) and finally the list of properties and the respective time-interval to report to the IoT Gateway.

Scenario-based Event Setup

There is another configuration file called the scenario configuration file. The main purpose of this configuration file is to specify when the scenario needs to be executed, the list of IoT devices properties which need to be included in the scenario and the values of the properties of those IoT devices during the scenario. The default values need to be specified at the end of the scenario-based configuration file, to complete the scenario.

High-Bandwidth Data Generation Using Video Streaming

As discussed above, there are two main components in simulating an IP camera: the producer, and the consumer. The video that needs to be streamed in the IP camera needs to be configured in the simulation configuration file, along with the period of the stream. Based on the configuration, Kaala 2.0 will connect to the RTSP server to send video data. The producer keeps producing the video to the RTSP server by connecting to the local RTSP server port and the consumers can consume the video by connecting to the client port of the RTSP server of the respective IP camera IoT devices.

NextG Integration

We leveraged Open Air Interface's (OAI) [15] UE, RAN and 5G Core modules for Kaala 2.0. Each of the IoT device will run OAI's UE module. To simulate 5G networks, the 5G Core and RAN are started in sequence. Then the IoT devices are deployed and the UE module in the IoT device connects to the configured RAN accordingly. If a IoT device is configured to be a NextG capable, then it will act as a UE and try to connect to the 5G RAN else it will try to connect to the cloud using regular IP network.

Simulation and Performance Evaluations

We seek to understand the capabilities, scalability, and performance of Kaala 2.0. We connect simulated IoT devices to a real network (we use Amazon AWS as case study) using 5G networks and finally evaluate a scenario-based simulation using IP networks.

System Setup

For the experimental setup, we used a Linux Virtual Machine (VM) which was allocated 4GB memory, two processors and 20GB for storage. The VM was running Ubuntu.

Scenario-Based Event Simulation

We assess the scenario- based event simulation in Kaala 2.0. First, for fire-in-the-room scenario, the necessary IoT devices were configured via the Kaala 2.0 configuration file. This configuration file is also a pre-configured profile in Kaala 2.0. Initially, all the IoT devices will be publishing respective data to the IoT gateway or IoT cloud and the data will be related to normal operation in a room. In the scenario configuration file, the time to start the scenario-based simulation will be specified. At that time, each IoT device property configured in the scenario configuration file will be configured with the value specified in the scenario configuration file. And the values get changed synchronously across all these IoT devices. The flame sensor will report 'true' stating that a flame has been detected. The temperature sensor shows a significant increase in the current temperature. Also, a video in which fire is shown is played at the same time.

High-Bandwidth Data Generation

Kaala 2.0 have an RTSP server running when simulating an IP camera IoT device. The server and client port of the RTSP server are configurable in the IP camera IoT device profile. As of now, the simulated IP camera IoT device can run server and client on the same port number. This can be a potential future work to support different ports for different IP camera IoT devices. An 8K video is sent to the simulated IP camera by connecting to the server port of the RTSP server. And the consumer consumes the video by connecting to the client port of the RTSP server which is running in the simulated IP camera IoT device. Both the producer and consumer were run in a host other than the IP camera IoT device, so that the traffic can be flowed in the network. The video being played by the producer can be completely controller by the application using the simulator configuration file. Additionally, the timing of different scenarios can be controlled and configured by the scenario-based configuration file.

The simulation framework loads a configuration file during startup. This configuration file includes the list of IoT devices, which needs to be simulated. The support for new IoT devices can be easily added to Kaala 2.0 by just adding a profile for the newly added IoT device.

NextG Simulation

During the startup, an instance of 5G Core and RAN are started and the RAN is connected to the 5G Core. In the configuration file, if the IoT device is configured as NextG capable, then the IoT device acts as a UE and successfully connected to the configured RAN. Once when the UE is successfully attached to the RAN, the IoT data is sent through the RAN and 5G Core to the IoT cloud. Support for multiple UE's connecting to the 5G core was evaluated and the data flow from the UE to the IoT cloud through the RAN and 5G Core was evaluated as well.

Interacting with Real Systems Using 5G Networks

For this experiment setup, we first created a user profile in the AWS IoT, and then configured an IoT device along with the necessary security certificates that are required to authenticate and connect to the AWS IoT Cloud. The security certificates are downloaded in the host machine in which Kaala 2.0 is running. Next, in the Kaala 2.0 configuration, we specify that a vendor specific IoT device needs to be simulated; the path to the downloaded security certificates is configured as well. These steps can be repeated for any number of vendor- specific devices. The same steps can also be followed for different vendors, like Microsoft Azure or Google IoT clouds, as well. Then, we start the simulation framework. Based on the loaded configuration, Kaala 2.0 knows that a vendor specific IoT device needs to be simulated and the application in the simulated devices will try to connect to the IoT gateway or IoT cloud using the provided security contexts using the above mentioned 5G network. Once connected, the simulated IoT device will start publishing the data. Particularly, in this experiment, all the data is published to AWS IoT cloud.

Scalability and Performance

To understand the memory consumption and scalability of Kaala 2.0, we conducted two experiments; In the first and second experiment, we simulated 10 and 100 IoT devices respectively. The simulated IoT devices were a combination of temperature sensor, smoke sensor, humidity sensor, flame sensor and motion sensor. Each of these sensors publish their data based on their IoT device profile every 15 seconds. The data in Figs. 4 and 5 shows that as the number of sensors increases, the processor and memory usage increase significantly. Kaala 2.0 is not just a simulator; it is an emulator as well. Each sensor runs its application logic in an individual process. So, the processor and memory usage are expected to increase as the number of devices scale up. This is because as the number of IoT devices increases, more processes are created. Since Kaala 2.0 uses Mininet and the Docker framework to simulate

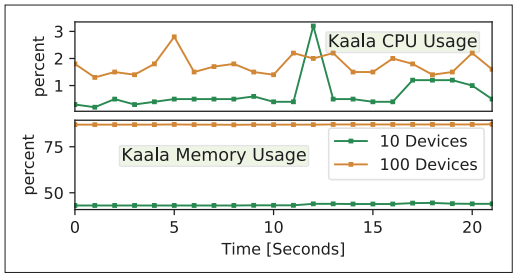

FIGURE 4. Kaala 2.0 Performance Evaluation — Mininet.

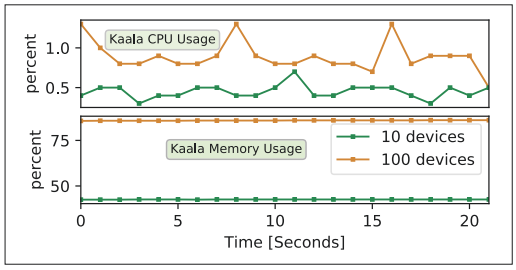

FIGURE 5. Kaala 2.0 performance evaluation—docker.

IoT devices, for each IoT device, Mininet, and the Docker framework create a network interface. For this article, we didn't use real-world or simulated 5G networks for the performance evaluation. The evaluation was done using IP networks.

Conclusions

We have presented *Kaala 2.0*—a modeling, simulation and emulation platform that can create IoT devices of various types. We were able to simulate devices which can generate large amount of data to verify and validate 5G technology. We were able to simulate an entire 5G system which includes UE, RAN, and 5G Core, and each IoT edge device was treated as a UE when it connects to the NextG network. As a future work, we are planning to do performance evaluation using 5G networks.

Acknowledgments

This research was in part supported by NSF under grants CNS-1814322, CNS-1831140, CNS-1836772, CNS-1901103, CNS-2106771, CNS-2128489 and CNS-2212318.

References

[1] Capgemini, "The Growing Need For Private 5G Networks in Manufacturing Plants," https://www.capgemini.com/insights/expert-perspectives/the-growing-need-for-private-5g-networks-in-manufacturing-plants; accessed Dec. 10, 2022.
[2] AWS, "Run Lambda Functions on the AWS IoT Greengrass Core," https://docs.aws.amazon.com/greengrass/v1/developerguide/lambda-functions.html; accessed Nov. 5, 2022.
[3] M. Azure, "Azure/azure-iot-sdks," https://github.com/Azure/azure-iot-sdks; accessed Oct. 19, 2022.
[4] G. Cloud, "Overview of Internet of Things | Solutions | Google Cloud," https://cloud.google.com/solutions/iot-overview; accessed May 23, 2022.
[5] U.-I. Project, "Deliverable d03.01, Report on IoT Platform Activities," https://docbox.etsi.org/SmartM2M/Open/AIOTI/IoTPlatformsAnalysisToImprove/D0301WP03H2020.
[6] U. K. Dayalan *et al.*, "ECIoT: Case for an Edge-Centric IoT Gateway," *Proc. 22nd Int'l. Workshop on Mobile Computing Systems and Applications*, 2021, p. 154–56.
[7] U. K. Dayalan *et al.*, "Veeredge: Toward an Edge-Centric IoT Gateway," *Proc. 2021 IEEE/ACM 21st Int'l. Symposium on Cluster, Cloud and Internet Computing*, 2021, pp. 690–95.
[8] M. Chernyshev *et al.*, "Internet of Things (IoT): Research, Simulators, and Testbeds," *IEEE Internet of Things J.*, vol. 5, no. 3, 2017, pp. 1637–47.
[9] U. K. Dayalan *et al.*, "Kaala: Scalable, End-to-End, IoT System Simulator," *Proc. ACM SIGCOMM Workshop on Networked Sensing Systems for a Sustainable Society*, 2022, p. 33–38.
[10] AWS, "IoT Device Simulator," https://aws.amazon.com/solutions/ implementations/iot-device-simulator; accessed June 10, 2022.
[11] M. Salama *et al.*, "IoTnetsim: A Modelling and Simulation Platform for End-to-end IoT Services and Networking," *Proc. 12th IEEE/ACM Int'l. Conf. Utility and Cloud Computing*, 2019, pp. 251–61.
[12] IETF, "Real Time Streaming Protocol (RTSP)," https://tools.ietf.org/html/rfc2326; accessed Nov. 12, 2022.
[13] Mininet, "Mininet — An Instant Virtual Network on your Laptop (or Other PC)," http://mininet.org/overview; accessed Oct. 24, 2022.
[14] Docker, "Get Started with Docker," https://www.docker.com/get-started; accessed Dec. 1, 2022.
[15] Open Air Interface, "Open Air Interface," https://openairinterface.org; accessed Oct. 12, 2022.

Biographies

Udhaya Kumar Dayalan [SM] (dayal007@umn.edu) is an Engineering Manager, Connectivity at Trane Technologies for the past 15 years. He is also a Ph.D. candidate in Computer Science at the University of Minnesota. He has received a B.S. degree in Computer Science and Engineering from the Anna University, and a M.S. in Software Engineering from the University of Minnesota. His research interests include 5G, 6G, wireless communications, Internet of Things, and communications security. He has more than 18 years experience designing and developing data communications and security and Internet technologies products and networks. He has filed more than 15 Patents.

Timothy J. Salo (salox049@umn.edu) is a Ph.D. candidate in computer science at the University of Minnesota. He has earned a B.S. and an M.S. in computer science, an M.B.A. from the University of Minnesota, and an M.S. in software engineering from the University of St. Thomas. He has also served as a part-time lecturer teaching undergraduate computer science courses in computer networks and operating systems. He is the founder and president of Salo IT Solutions, Inc. (SaloITS). He has over three decades of experience researching, designing, developing, marketing, deploying, and operating data communications and Internet technologies, products, and networks.

Rostand A. K. Fezeu (fezeu001@umn.edu) is a Ph.D. candidate in computer science at the University of Minnesota. Rostand earned a B.S. degree in computer science and mathematics with an emphasis in computation from Minnesota State University Moorhead, and a M.S. in computers science from the University of Minnesota. He has previously worked for big technology companies like Microsoft and Cisco Systems, and has a solid experience in theories, researching, developing, and implementing Internet technologies and networks systems.

Zhi-Li Zhang [F] (zhzhang@cs.umn.edu) received his Ph.D. degree in computer science from the University of Massachusetts. He joined the faculty of the Department of Computer Science and Engineering at the University of Minnesota in 1997, where he is currently the McKnight Distinguished University Professor and Qwest Chair Professor in Telecommunications. He currently also serves as the Associate Director for Research at the Digital Technology Center, University of Minnesota. His research interests lie broadly in computer and communication networks, multimedia systems and content distribution networks, cyber-physical systems and Internet of Things, machine learning, and data mining.