Experiments with a domain specific language for simulation based testing of IoT edge-to-cloud solutions

Pandravada Abhiram

August 2022

1 Introduction

IoT is used in several domains, which require a consistent performance even in extreme scenarios. Testing by replicating these scenarios in real life would be impossible to conduct. Hence, simulation is an important tool which we can use to test the system under extreme scenarios. Not only does simulation give us data about the performance of an edge device in extreme scenarios, it provides us with further information on how to properly structure our development and deployment of devices.

The Internet of Things (IoT) envisions sophisticated systems that connect vast numbers of smart objects equipped with sensors and actuators over the Internet or other network technologies. IoT is already widely used in a variety of applications, including emergency response, smart cities, smart agriculture, and autonomous cars, to mention a few.IoT simulation has several features, including IoT sensor and actuator simulation [Ahl+21] [Pen+18] [Bor+21], IoT edge device simulation [Nan+19] [SOE18], and IoT network simulation [Bou+11] [KBK12] [Ost+06] .

2 Problem description

In this research, I plan to simulate edge devices and their interactions with cloud applications. An edge device is a broad phrase that refers to any device that acts as a network entry point. Edge devices in IoT often runs programs and protocols that manage groups of sensors and actuators, conduct edge computing, and send and receive data to and from cloud applications.

An edge-to-cloud simulator performs the functionality of a large number of edge devices that operate in parallel and connect synchronously or asynchronously with cloud applications [Ela19]. Edge-to-cloud simulators may be

used for testing and making decisions, such as how frequently edge devices should connect with the cloud and how to plan data flow from IoT devices so that total network traffic stays constant and burst-free.

In this project, scale testing of cloud applications is conducted which will yield how well the performance of the system is under an extreme work-load and at peak limits of its capacity. The workload is generated by edge devices and is sent to the cloud for processing and storage. Existing feature-rich IoT simulators [Nan+19] [SOE18] are not suitable for cloud scale testing, since they may demand significant computing resources (CPU, memory, and storage) to mimic thousands of devices. For simulation, many IoT vendors employ hosted services. The more resource-intensive the simulator, the more difficult it is to set up and the more expensive it is to run for scale testing.

Replicating extreme scenarios involves proper use of communication protocols, managing massive concurrency of simulated edge devices, and efficient use of computing resources. Hence, instead of building an Edge-to-cloud device from scratch, I will utilise the existing work done using a domain-specific-language (DSL), named IoTECS(the IoT Edge-to-Cloud simulation language), which provides resources for constructing such a simulator. My role in this project would be to try to replicate the installation steps of the tool and successfully run the tool which would hopefully deliver positive results of the work done. IoTECS offers a hierarchical design for grouping huge arrays of simulated edge devices, based on ideas and standards from the multiagent simulation literature [Sch+95]. The resultant groups can then be simulated using containerization, virtualization, or a mix of the two. IoTECS also provides variable time intervals for coordinating the execution of parallel edge devices, ensuring that communications between edge devices and cloud applications are sent at a rate that is consistent with real-world expectations in a given IoT system.

3 Project scope

The scope of this project can be explained with a use case scenario.

Sensors in a smart city monitor a variety of aspects such as weather, illumination, motion, traffic conditions, utility use, water chemistry, and air pollution. The data collected by these sensors is utilised to make decisions and send orders to actuators that regulate things like traffic and weather. lights on the roadway, alarms, barrier gates, pumps, and heating and cooling systems. The sensors and actuators that are near to each other include clustered together and wired or wirelessly connected to a gateway, also known as an edge device Typically, gateways have little processing capacity, capable only of rudimentary data analysis A gateway transmits sensor data, maybe after some processing. - to cloud-based services The interaction between a gateway and The cloud is connected to a core network. When the cloud services receive unprocessed or semi-processed sensor

data, they undertake a more thorough analysis of the data, create analytics, identify any necessary course of action, and provide instructions back to the gateway, which then communicates the instructions to actuators.

This smart city is one of the many use cases that are present in the applications of IoTECS. Ofcourse, there are multiple factors to be considered here like, data storage, data analysis and decision making services which are deployed to the cloud. The producers of these services must address a crucial question: how far their services will scale in the face of high demands from a large number of IoT sensors, actuators, and gateways. Simulating extreme scenarios thus opens up multiple avenues for a company that can now work on development and deployment tasks in a cost-efficient manner. In order to serve cost-efficiency, IoTECS is developed. In the development scenario, we can outline the details of how IoTECS can help a team of engineers determine in a systematic way the number of IoT and edge devices that the cloud applications under extreme scenarios can handle.

The scope of this project further expands by creating an FAQ report followed by a set of experiments that will be conducted on IoT devices from Stratosfy. These steps will give us development scope into not only the IoT devices but also the current IoTECS product.

4 Objectives

To summarize the objectives of this project, we have the following:

- Understand the requirements and tools necessary to replicate IoTECS.
- Simulation of edge-to-cloud functionalities under extreme scenarios using the Domain Specific Language - Internet of Things Edge to Cloud Simulator.
- Create an FAQ report that will outline the basic queries to some edge case scenarios to be considered while using the application.
- Evaluate the results obtained from replicating the work and outline the optimal number of IoT and edge devices required to offer the best and most reliable connections to the cloud.
- Layout future work and improvements that can then be made out of the evaluations of the project.

5 Work plan

S.No	Deliverable	Description	Deadline
1	Initial Project Proposal	A detailed description of an initial project that will highlight the motivation, necessity and outcomes expected from the project.	September 15th, 2022
2	Replication of IoTECS application	A step-by-step tutorial explaining how IoTECS can be used and executed. This will give me a better understanding of the product as well as describing to any new user the steps to use the product.	September 30th, 2022
3	FAQ report	An faq report describing the most common problems/errors one may have using IoTECS and how they can be solved. This will answer not only some underlying issues (if found) withing existing IoTECS product as well as a general report that will help the developers in the future with development of the product.	October 31st, 2022
4	Applications	The current IoTECS products has only been used in Cheetah networks - a company which develops AI-based solutions for real-time monitoring of quality of experience in IoT networks. To expand the scope and horizon of this project, a set of experiments that you perform by applying the IoTECS simulator to some products at Stratosfy.	December 10th, 2022

6 Step by Step process to setup IoTECS

IoTECS has esclusive scripts written out for you to run on either a cloud machine, a linux based VM or your personal local machine. In this section, a step-by-step guide is given for you to make an initial setup of IoTECS on your personal machine. The prerequisites for this project are:

- Eclipse IDE 2022-09
- Java Development Kit (version 11 or above)
- Virtual Box (version 6.1 or above)

- Docker (version 20.10.11 or above)
- Xtext (version $\geq$ 2.25.0. Can download from Eclpise, or from the original distribution site)
- Xtend (version $\geq$ 2.25.0. Can download from Eclpise, or from the original distribution site)
- Ubuntu 20.04 disc image
- Wireshark (version *geq* 3.4.9)
- Python (version 3 or greater)
- ssh for shell scripting.

The installation instructions are as follows:

1. Visit the GitHub https://github.com/JiaLi123456/IoTECS.
2. Download the zip file and extract the file.
3. Open all the five projects in IoTECS.zip in Eclipse.
4. Go to iotecs > src > iotecs > IoTECS.xtext, right click, run as generate Xtext artifacts.
5. Go to iotecs > src > iotecs > GenerateIoTECS.mwe2, run as MWE2 workflow.
6. Run the DLS project (iotecs) as Eclipse Applications and choose launch runtime Eclipse.
7. Create linux-based (Ubuntu 20.04) virtual machines for each platform whose type is "VM".
8. Install python, jdk and wireshark on the local machine and all remote platforms.
9. Create a java project on the runtime Eclipse Platform and new a file with a suffix of "iot" (e.g., test.iot). Choose "yes" when a window called "Congifure Xtext" popped out asking "Do you want to convert 'test' to an Xtext project?". Input the specific instance of conceptual model for IoTECS according to the grammar. Or use the demonstration IoTECS project provided in Demonstration and modify it according to the instance of the conceptual model.
10. Save the file in step 9 and the code and scripts are generated in a directory called "src-gen" of the project in step 9 (e.g., test>src-gen).
11. install openssh and make sure localhost, Platforms, SimulationNodes can communicate with each other using ssh and scp.
12. Go to the generated directory of the IoTECS project (src-gen/) and run main.sh.

6.1 Example

In this section, a simple example is given to setup the simulator in your personal machine. In step 6, when you run as eclipse application, an instance of Eclipse application is opened for you. in the new instance, create a "new java project". Create a file with the extension ".iot" in the src folder. To create your simulator you need to define the following items:

1. Simulator
2. Simulation Node
3. Edge Device
4. Device
5. Cloud
6. Platform

Let us now build a simulation model that would run on your own personal device.

- Simulator: Captures the attributes requried for mnaging time. We execute each simulation for a given duration and divide this duration into equal time steps, which is denoted bu duration and step field in the object. The xtext file defines the following properties for a simulator.
  - duration: We have to specify the duration for which the simulator is going to run. Lets define this to be 40s.
  - step: This defines individual time steps to execute the edge devices. In this example let us set this to 2s.
  - The next parameter to be defined is the different of simulation nodes present and the number of each type. The number of devices can be defined using the array syntax. Eg: If we want to define a Simulation node SN1 which has 25 devices, we write SN1[25]. We can define multiple types of nodes using the comma separator.

  Our Simulator object now looks like this:
  Simulator: {
  duration: 40s
  step:2s
  simulationNodes:{SN1[25]}
  }

- Platform: Each simulation node is associated with a platform represented by Platform concept. Edge devices that belong to a given simulation node all execute in parallel and on the platform of the simulation node. Since we are using our local machine as the platform, username and password need not be defined. However, we must define the ip, username and password for a virtual machine or docker platform. Based on the xtext file, we have:

- Platform: We need to define the ID of our platform. This could be P1. If we did have another platform which is in a docker, we have to declare the type explicity as Docker. The same goes for VM.
- Network adapter: This defines which network adapter the packets are sent and received. We define this mainly for using tshark in the bash script. Using tshark, we observe the channel in which the packets are being sent and received. The script for this is generated automatically, but the network adapter is a parameter that is required. Please refer the FAQ section to know how to fnd the network adapter.
- locationIP: For Docker and VMs, we must define the location IP. This is the IP address at which your VM or docker is running on. Please refer the FAQ section on how to find these details.
- username: Since this is not your local machine and you cannot have the control to authorize a VM/Docker directly, we must define the username of your docker/VM on which you are running.
- : password: This is the password you created while setting up the docker/VM.
- cpu: This is a very important feature which you use to allocate the resources to docker/VM. While creating a VM for example, you will be prompted to allocate the CPUs for the VM. I have used VirtualBox for setting up my VM. I allocated 4 CPU cores out of available 8 for my VM. So in here, I allocated all the amount of cores available on VM (i.e 4).
- memory: This is again up to you, but please keep this number below the amount of memory allocated to your VM.

Summing all the above points, we define our platform 1 on local machine, platform 2 on docker and platform 3 on VM:
Platform: P1 {
type:Native
networkAdapter:"eth0"}
Platform: P2 {
type:Docker
networkAdapter:"eth0"
locationIP:192.168.2.48
username:abhi
password:simulator
cpu:4
memory:2G
}
Platform: P3 {
type:VM
networkAdapter:"eth0"
locationIP:192.168.

username:uservbox
pass:testing
cpu:6
memory:5G
}

- Simulation Nodes: Above we have defined the blue print of our simulator that would contain the simulation nodes. These contain the details of edge devices, the platform they run on. Now we define these simulation nodes and on which platform these would be running along with the edge devices contained inside it. Note that we can define the platform as P2 and P3 on our choice and create more Simulation node types.
  SimulationNode: SN1{
  platform:P1
  EdgeDevices:{ E1[8], E2[4]}
  }

- Edge Device: This is a generic term that serves as an entry point to a network. In IoT, these devices run applications and protocols for managing groups of sensors and acutators, performing edge computing. We have declared edge devices in the simulation nodes above. Now, we have to define these values in the edge devices object. Based on the parameters provided in our xtext file, we have :
  EdgeDevice: E1 {
  protocol:TCP
  speed:Max
  cloud:C1
  workload:10ms
  devices:{D1[40]}
  }

- Cloud: The workload is originated from the edge devices and is destined for processing and storage by the cloud. We declared cloud and devices for a particular edge device. Now, we define these values separately, using the parameters provided in the xtext file. Note that we have to separately define each device we declared above. If we declared a second device we must define it separately.
  Cloud:C1{
  ip:0.0.0.0
  port:50051
  }

- Device: To define the device, we can do it like this:
  Device: D1 {
  period:1
  payload:80b
  }

Now that we defined our models, once we save the file, we see an src-gen folder created in our project. To run this simulation, we run the main.sh file. This file contains all the scripts required for each component. Once we run this, a terminal opens which will run the script in it. Some messages to note are the tests we defined and the build status. We can evaluate our simulation if we have a successful build and the number of packets lost which we can see from wireshark. In order to evaluate these values, start a wireshark capture in the network adapter that you defined. In my case, I have an ethernet cable and all my requests were running on the Ethernet interface.

7 FAQ

1. Q. How do I find my ip address:
   A. If on a windows device, please search for Control Panel > View Network status and tasks > under internet, select your network > Details > Ipv4 address. You can also define your local machine IP address as 0.0.0.0 which is the default host address.

2. Q. How to check my IP address for docker?
   A. On your docker terminal, run: docker-machine ip default

3. Q. How to check IP address on my linux VM?
   A. Open the terminal of linux vm and type: ip addr. This will give you the details necessary to set the ip address.

4. Q. How to properly install Xtext and Xtend on Eclipse
   A. It is recommended to download xtext and xtend from the following links :
   https://www.eclipse.org/Xtext/download.html and
   https://www.eclipse.org/xtend/download.html. Right click and copy the link from these distribution URLs. In Eclipse, go to Help > Install New Software and paste the copied link and follow the instructions listed in the installer in eclipse.

5. Q.What is the maximum number of devices you were able to run the simulation for?
   A. The maximum number of simulation nodes I could run without a memory error was up to 75. So the way I have constructed is, there are 50 SN1 and 25 SN2. On top of this there are 75 edge devices for the two simulation nodes, divided into 50 and 25 respectively. All these edge devices were defined the same way. Finally, I have added a 100 devices to each edge device. The above program managed to build successfully without any memory constraint. Going above these numbers resulted in an insufficient memory error.

6. Q. How do I capture the packets by myself using wireshark?
   A. Create a .iot file and run main.sh. Parallelly, open wireshark and apply

the filer ¡protocol_you_chose¿.port == ¡port_you_chose¿. For example, if the total protocol defined for edge device is UDP and the port chosen in cloud is 1883, your filter is udp.port == 1883. Alternatively you can filter by ip address of the cloud and the protocol chosen. An example would be ip.addr == "192.168.0.2" when the cloud ip was set to 192.168.0.2.

7. Q. What are the changes to be made in the Windows platform for running the scripts?
   A. For windows, in main.sh, make the change mentioned in section 8. Along with this, some commands in windows are different to those of mac. For example, we dont have sudo in Windows. The places where sudo was used were to kill the processes in platform_¡platform name¿. Please replace these with a simple "kill $tsharkpid¡num¿". This ensures this runs in a windows OS. Furthermore, to run the python files, replace python3 with python. Finally, manually set the ip address for the ip variable. There is no direct way to set ip address in windows OS.

8. Q.How are the packet losses in different run configurations?
   A.

   (a) RQ2 configuration used(payload attached to devices); Simulation nodes: SN1[5]; Platform:Native; Edge Devices: E1[10]; Devices+Payload:D1[20]+60b

Total packets sent	Delay	Packet Loss
45015	0.24574299999999996	0

   (b) RQ2 configuration used(payload attached to devices); Simulation Nodes: SN1[10], SN2[20]; Platform:Native+VM; Edge Devices: E1[10] given to SN1 and E2[25] given to SN2; Platform: P1 native, P2 VM (ip address from the VM is attached here). Username and pass are the ones from the one we setup the VM on. Cloud: ip address of local machine is given. port set to 1883. Devices D1[100]+60B and D2[50]+30b Local machine stats:

Total packets sent	UDP Delay	TCP Delay	Packet Loss
216+317	1.17107	0.98045	9.005%

   (c) RQ2 configuration used(payload attached to devices); Simulation Nodes: SN1[10], SN2[20]; Platform:VM ; Edge Devices: E1[100] given to SN1 a; Platform: P1 VM (ip address of vm given). Username and pass are the ones from the one we setup the VM on. Cloud: ip address of VM given. port set to 1883. Devices D1[100]+60B and D2[50]+30b.

UDP packets sent	Delay	Packet Loss
67012	1.0122229	4.9%

8 Errors

1. When you try to run the simulator using the main.sh file, you might get the error "No such file or directory". In this case, Windows users need to edit their derived script files. In windows, automatic setting of Current working directory is not supported. Please paste the following command at the start of main.sh file and platform.sh:
   parentpath=$(cd "$(dirname "$BASH_SOURCE[0]")" ; pwd -P)
   cd "$parentpath"

2. You might get the error "run_SN2" not found, although you have not created an SN2 node in my .iot file either. Why is this created in bash?
   A. Please remove the bash run_SN2.sh, if you have not created an SN2 node in your .iot file. Please double check the main.sh file that only the nodes that you mentioned in the .iot file are present in the main.sh file.

3. The iotecs.tests and iotecs.ui.tests files show errors, specifically "org.junit.jupiter.api cannot be resolved". This is an important error that you need to fix, otherwise the packets will not be shown in the VM/Local machine's wireshark. To fix this, go to window > show view > problems. Then click on the problem in the terminal. Remove the org.junit.jupiter.api line where the error is shown, and replace this with "Import-package: org.junit5" outside the Require-Bundle statement. Now the error should be fixed.

9 Experiments on Stratosfy's setups

In this section, we will setup our simulator for Stratosfy - An IoT based company. In order to perform this experiment, we require the details of how Stratosfy deploys their product in a region.
In a single location, Stratosfy deploys a maximum of 8 edge devices that all link up with a single gateway. So in our case, the cloud can be considered as a gateway since that is where the computation of data gathered by edge devices is going to happen. These edge devices are basically sensors that record the temperatures and send this data to the gateway or cloud via Bluetooth. Unlike TCP or UDP, bluetooth communication is primarily taken place using MQTT. In IoTECS, we do not have support for this feature. Based on how the packets are sent in MQTT, we will use TCP. Under the hood, the structure of MQTT is similar to TCP except the data transfer is done over binary. Furthermore, MQTT and UDP are not similar in the way they operate. Keeping these factors in mind, we use TCP communication as our means for communication over the network.
The gateway or the cloud will listen to these recordings sent over by the edge devices. It is connected to a Wi-Fi directly. Why? Bluetooth devices require lesser energy than Wi-Fi operated devices. Since gateway has a constant electric supply, it can talk directly with Wi-Fi without any problem. In a scenario where there is a power loss, the gateway has a battery that will operate for atleast 4

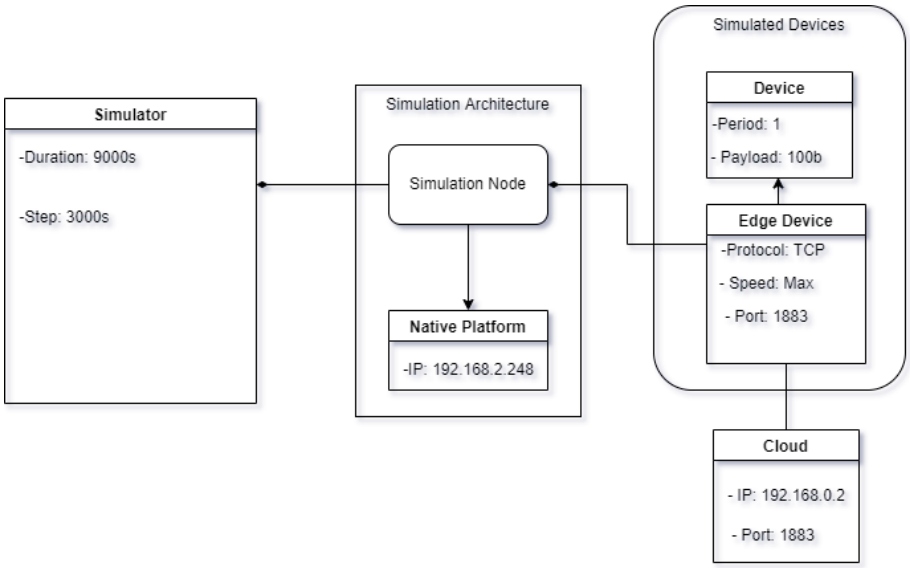

Figure 1: Simulation on Native Machine Workflow

hours. Based on this data, the connection we use is Wi-Fi.
How long must the simulator run for? The gateway collects information from the edge devices every 15 minutes. So, we will be running the simulator for for 15 minutes i.e. 900 seconds. We will then report the total packets lost during this time.

For Stratosfy, we have at most 8 edge devices that read data from the sensors. In figure 1 and 2, the edge devices read temperature from the devices. Further computation is performed at the cloud level aka the gateway.
Regarding the way we run this simulation in oursystems, the topology is explained in figures 1 and 2. Figure 1 shows us how the simulation is run on our local setup. Suppose we have a total of 4 devices, the step value makes sure each of these devices runs in the following time period. If a device has a period of 1, that means it will run of 1 second in its allocated time by the simulator.
A similar approach is taken with Docker, except here, we are looking at running our simulator on the docker machine. For this scenario, we take into consideration the username password and provide the ip address in which or docker machine is running on.

9.1 Experiment setup

Let us use the same concepts we did in section 6.1 where we setup a single example setup.

- Simulator: The duration of the simulator will be for 9000s i.e 15 mins.

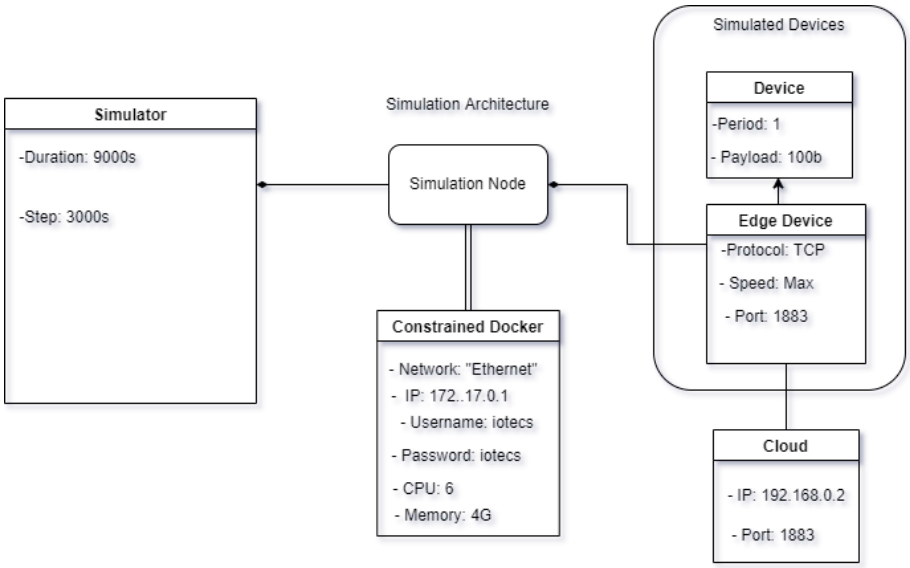

Figure 2: Simulation on Docker Machine Workflow

The step factor is not implemented in Stratosfy, except that each edge device reports data to cloud every 15 minutes. But, edge device senses the temperature every minute without any interruption. Hence for the sake of the project, we set step to 3000s i.e 5 mins. The simulation nodes here is 1m since for every case of Stratosfy, they set a maximum of 8 devices. So the worst case scenario for a system to fail is 8, and we are testing for 8. So any other setup will definitely do better than the numbers published here. Now, our simluator looks like this:
Simulator: {
duration: 9000s
step:3000s
SimulationNodes:{SN1[1]}
}

- Platform: The platform the computation and communication runs on is done on a server that runs a Node JS application with a Google Firebase backend. So it is unclear on which platform to setup for this case, hence we will go with Native.

- Each simulation node consists of a maximum of 8 edge devices. So, we have
  EdgeDevice: The total number of sensors present in the system is 8, but these sensors collect data from numerous devices. Using our knowledge from previous sections, we can set the total devices to 50 as an edge case
  E1 {

protocol:TCP
speed:Max
cloud:C1
workload:20ms
devices:{D1[50]}
}

- Cloud: Since there is a single gateway that has constant power supply and is connected to a power source, we have:
  Cloud:C1{
  ip:192.168.0.2
  port:1883
  }
- Device:The information for period and payload is not obtained, hence we will use the knowledge from previous sections and push these values to the limits.
  Device: D1 {
  period:1
  payload:100b
  }

9.2 Results

Upon running this setup on native platform, we have the following results:

Total packets sent	Delay	Packet Loss
59517	1.42	1.1%

Running the setup on Docker gives us:

Total packets sent	Delay	Packet Loss
59496	2.33	1.9%

Running the setup on VM was causing persistent connection errors. Hence this was skipped.

10 Related Work

In this section, we shall look at similar work done by fellow researchers in IoT device simulations, IoT testing and domain-specific languages for IoT.

1. IoT Simulation: In an IoT system, we have many components that we might want to simulate. Some of them are actuators and sensors, networks, cloud infrastructure. Unlike this project we we have implemented every component, some papers tended to focus on specific domains. For example [Pfl+16] focused on simulating actuators and sensors; [Kum+19] - worked on security of network in an IoT system simulation and [SKM21]

- implemented a simulatoin of the IoT system by focussing on 5G systems; When it comes to cloud infrastructure [Cal+09] propose a novel system for cloud infrastructure to provide support for large scale computing, self contained platforms, virtualization of simulations, and flexibility to switch between space-shared and time-shared allocation of processing cores to virtualized services.

2. IoT testing: Once we build a simultor, there are several tests that can be run on it to verify the performance.testing not only gives us a good estimate of our model, but als provides us with areas of improvement, if any. In this field, [Bei+22] provide an overview of the state of the art approaches present in testing an IoT system. One of the prominent papers in this field is [Moa+15a], where they provide discrete object-oriented modeling APIs, and enable models to represent very large sequences of sensor values by using mathematical polynomials. We show on various IoT datasets that this significantly improves storage and reasoning efficiency. In this paper [NMB21], the authors present a framework for transforming a cluster of lab machines into a Virtual Testbed that provides views of how systems will perform in a tiered IoT environment.

3. Domain Specific languages for IoT: In this paper we have created IoTECS which is a custom developed language to define an IoT system. This language as we have seen in section 9 has support for TCP and UDP. However, when developing a domain specific language, one can have a model in mind for which they want a language to adapt. For example, Stratosfy's goal would be to have a simulation that works with MQTT technology. In this paper [Moa+15b], the authors propose a concept of continuous models that can handle high-volatile IoT data by defining a new type of meta attribute, which represents the continuous nature of IoT data. All the languages we have seen thus far run on a terminal. [SN15] the authors propose a web-based domain-specific language for Internet of Things applications.

11 Conclusion

IoTECS is used for scale testing a large number of edge devices. The major factor for pursuing this project is gain information by scale testing the edge devices and evaluating the performance of these devices. By doing so, we can get the performance of edge devices and areas of improvement.

References

[Sch+95] M Scheffer et al. "Super-individuals a simple solution for modelling large populations on an individual basis". In: *Ecological modelling* 80.2-3 (1995), pp. 161–170.

[Ost+06] Fredrik Osterlind et al. "Cross-level sensor network simulation with cooja". In: *Proceedings. 2006 31st IEEE conference on local computer networks*. IEEE. 2006, pp. 641–648.

[Cal+09] Rodrigo N Calheiros et al. "Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services". In: *arXiv preprint arXiv:0903.2525* (2009).

[Bou+11] Athanassios Boulis et al. "Castalia: A simulator for wireless sensor networks and body area networks". In: *NICTA: National ICT Australia* 83 (2011), p. 7.

[KBK12] Dzmitry Kliazovich, Pascal Bouvry, and Samee Ullah Khan. "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers". In: *The Journal of Supercomputing* 62.3 (2012), pp. 1263–1283.

[Moa+15a] Assaad Moawad et al. "Beyond discrete modeling: A continuous and efficient model for iot". In: *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE. 2015, pp. 90–99.

[Moa+15b] Assaad Moawad et al. "Beyond discrete modeling: A continuous and efficient model for iot". In: *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE. 2015, pp. 90–99.

[SN15] Manfred Sneps-Sneppe and Dmitry Namiot. "On web-based domain-specific language for internet of things". In: *2015 7th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. IEEE. 2015, pp. 287–292.

[Pfl+16] Tamas Pflanzner et al. "MobIoTSim: Towards a mobile IoT device simulator". In: *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. IEEE. 2016, pp. 21–27.

[Pen+18] Sancheng Peng et al. "Influence analysis in social networks: A survey". In: *Journal of Network and Computer Applications* 106 (2018), pp. 17–32.

[SOE18] Cagatay Sonmez, Atay Ozgovde, and Cem Ersoy. "Edgecloudsim: An environment for performance evaluation of edge computing systems". In: *Transactions on Emerging Telecommunications Technologies* 29.11 (2018), e3493.

[Ela19] Hanan Elazhary. "Internet of Things (IoT), mobile cloud, cloudlet, mobile IoT, IoT cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions". In: *Journal of Network and Computer Applications* 128 (2019), pp. 105–140.

[Kum+19] Vikash Kumar et al. “Simulation Analysis of DDoS Attack in IoT Environment”. In: *International Conference on Internet of Things and Connected Technologies*. Springer. 2019, pp. 77–87.

[Nan+19] Devki Nandan Jha et al. “IoTSim-Edge: A simulation framework for modeling the behaviour of IoT and edge computing environments”. In: *arXiv e-prints* (2019), arXiv–1910.

[Ahl+21] John Ahlgren et al. “Facebook’s cyber–cyber and cyber–physical digital twins”. In: *Evaluation and Assessment in Software Engineering*. 2021, pp. 1–9.

[Bor+21] Markus Borg et al. “Digital twins are not monozygotic–cross-replicating adas testing in two industry-grade automotive simulators”. In: *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE. 2021, pp. 383–393.

[NMB21] Fotis Nikolaidis, Manolis Marazakis, and Angelos Bilas. “IOTier: A Virtual Testbed to evaluate systems for IoT environments”. In: *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE. 2021, pp. 676–683.

[SKM21] Donát Scharnitzky, Zsolt Krämer, and Sándor Molnár. “Comparison of TCP SIAD and TCP BBR Congestion Control in Simulated 5G Networks”. In: *Acta Universitatis Sapientiae, Electrical and Mechanical Engineering* 13.1 (2021), pp. 114–124.

[Bei+22] Jossekin Beilharz et al. “Continuously testing distributed iot systems: An overview of the state of the art”. In: *International Conference on Service-Oriented Computing*. Springer. 2022, pp. 336–350.