IoTbed: A Generic Architecture for Testbed as a Service for Internet of Things-based Systems

Mahmud Hossain, Shahid Noor, Yasser Karim, and Ragib Hasan
{mahmud, shaahid, yasser, ragib}@uab.edu
Department of Computer and Information Sciences
University of Alabama at Birmingham, AL 35294, USA

***Abstract*—In recent years, the concept of the Internet of Things (IoT) has already been implemented in numerous application domains, such as smart city, intelligent healthcare assistant, and smart transportation management. Researchers or developers associated with these domains frequently require diverse types of IoT devices to implement or test their IoT related innovations. However, building an IoT-testbed by purchasing these IoT devices in every few months is infeasible considering that most of them will be used only for a small amount of time and will become obsolete with the arrival of an upgraded version of those devices. Instead, a better approach would be rented out the IoT devices in on-demand, just-in-time, and pay for only the time the devices are used basis. Therefore, in this paper, we propose IoTbed – an architecture to deliver a large scale Testbed as a service using the IoT devices located in the users' proximal networks. IoTbed enables testbed providers to build their testbeds in the edge of the network and allows easy and secure access for IoT service providers to run experiments on those testbeds. IoTbed presents an economic model that offers monetary incentives to the device owners for using their devices in the testbed experiments. We posit that such an incentive model will encourage more testbed providers to rent out their smart devices through IoTbed. To demonstrate the feasibility of IoTbed, we implemented a prototype of IoTbed to run on Contiki powered IoT devices and evaluated the performance.**

***Index Terms*—Internet of Things, Testbed, Incentive, Architecture, Contiki, Cooja.**

I. Introduction

With the recent revolution of the computing devices along with technological advancement in communication, the Internet-of-Things (IoT) concept is utilized by several application domains [1–6], such as smart city, smart home, intelligent healthcare assistance, smart transportation management, and so on. Because of the light weight, decent computing and communication capabilities, and low cost structure people start reshaping their manually controlled environment with an automated one using the smarter IoT devices. Nonetheless, IoT is still a new concept and researchers are continuously improving it. There are several ongoing academic and industrial researches for enhancing the computing and storage capabilities, improving communication power, and providing a secure framework for the IoT based systems. IoT researchers often require IoT devices to test their proposed approaches in those devices and evaluate the performance of their proposed schemes. However, due to the heterogeneity of IoT devices, it is difficult and expensive for a researcher to build an experimental prototype. While some of them can afford it, they rarely venture on buying IoT devices because of the uncertainty of the outcome of their research. Therefore, it now becomes very important to provide an IoT testbed as a service where an individual can afford to perform their IoT related experiments.

A recent research anticipates that on an average around one million new IoT devices in each year will be deployed to different application domains for the next few years [7–10]. The new IoT devices would be deployed for the development of several personal and commercial smart applications and services [8]. However, the majority of those IoT devices will be remain unused either within a specified time interval in a day, month, or year. Additionally, some organizations upgrade their devices in each year. Therefore, the old devices remain unused unless they sell them to an individual or institute. Therefore, we can convince those IoT device owners to rent out their devices virtually while the devices are either idle or unused.

There are several challenges associated with providing an IoT testbed as a service. First, we need to have a proper incentive scheme for the IoT device owners in order to convince them for using their devices. Second, authenticating the valid users and device providers is challenging. Third, we need to provide an efficient scheduling approach for users requesting for some IoT devices. Fourth, monitoring the activities of the users using the IoT devices so that they can not do any harmful activities using the IoT devices is non trivial. Finally, we must have an efficient protocol for investigating any past malicious activities done by either any evil user or device owner.

In this paper, we present IoTbed – an infrastructure for delivering Testbed as a Service using IoT devices distributed in edge networks. Smart device owners develop testbeds in their proximal networks. IoTbed provides a mechanism to rent out these testbeds to users. Users run IoT-based experiments on a large number of IoT devices with various specifications and network settings through IoTbed. IoTbed also provides an incentive model to offer monetary incentives to the device owners that participate in experiments. In addition to provide the client specified testbed, IoTbed also monitors the users' activities and determine any incongruity in the system.

Contribution: The contributions of this work are as follows:

1) We proposed IoTbed, a generic architecture for Testbed as a Service using smart devices located in the edge networks.
2) We provided an incentive and contract mechanism which allows building and operating such a system in an economically feasible manner.
3) We implemented a prototype of IoTbed on the popular Contiki [11] operating system and evaluated performance to demonstrate the feasibility of our proposed model.

Organization: We present motivation and background in Section II and Section III respectively. The details of the architecture of IoTbed are presented in Section IV. Section V presents the operational model of IoTbed. We provide experimental results in Section VI. A comparison of IoTbed with contemporary research is presented in Section VII. Finally, we conclude in Section VIII.

II. Motivation

In recent years, a very large number of IoT devices have been developed for different applications. Such devices are continuously being improved and upgraded. Researchers are also developing new applications using IoT devices. However, this poses a new problem – how can a resource constrained IoT device researcher develop and test their IoT based applications without making a significant investment on the purchase and maintenance of the IoT device? This is especially critical for researchers from the developing countries. We argue that this is analogous to the development of the cloud computing data centers, where users with short term but large scale computations benefitted from the existence of clouds. In that case, the users who were required to make large investments in hardware purchases could just rent as many cloud machines as they needed, only when they needed, and paid only what they used. The same model can be applied in the IoT testing context – a researcher does not need a huge number of IoT devices just for the purpose of testing if such devices are made available for rent analogous to the cloud model. For example, consider the following scenario:

Example Scenario: *Dr. Smith has a research lab IoTSec in University of Alabama at Birmingham where he does several types of researches related to IoT security. Along with two more his research students he proposed HSC-IoT, which is a new lightweight device-to-device IoT authentication scheme. In order to test their proposed approach, they are currently working on building a prototype. Hence, they ordered some IoT devices for their lab through the university finance department. However, due to the administrative complications, they found that their order has not been processed yet. Dr. Smith would like to finish his work as soon as possible in order to work on his another idea that is dependent on the outcome of HSC-IoT. Dr. Smith contacts IoTbed for using their provided IoT testbed as a service and complete his research.*

Besides the educational or research institute, people in several other sectors continuously deploying the existing IoT research works and develop new IoT enabled smart products. Consequently, the IoT manufacturer is embracing some of the promising outcome of those research and release upgraded version of the existing IoT devices. In contrast, very few company can afford to buy those upgraded IoT devices and keep their research up-to-date. From the perspective of scalability, it is difficult to test IoT application properly without large scale deployment. Consider the following example scenario:

Example Scenario: *Mr. Brandon has a startup company TechCom where he designs several products related to healthcare. Their next venture is to build a smart gown for medical patient for monitoring the patients' condition efficiently. They have already designed some protocols as part of the development of the product. However, during testing, they figured out that their design is expensive for their currently available IoT devices. Currently, Mr. Brandonis somehow perplexed on taking decision of buying some new powerful IoT devices for TechCom. First, his company has a very limited budget, second, he is uncertain about the outcome of his new product. Finally, he is planning to work on other projects after this work and do not want his new devices to remain unused for the long time. Instead Mr. Brandon decided to contact IoTbed and use their IoT testbed for testing the performance of its new product.*

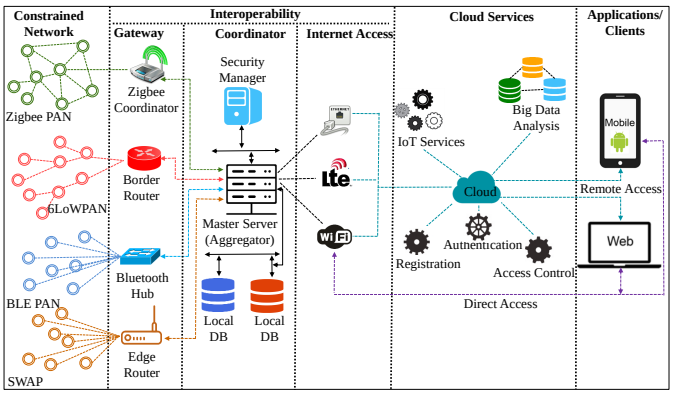


Fig. 1: An IoT-based system.

III. Background

In this section, we provide a brief overview of the interoperability of IoT components. Figure 1 presents the operational model of an IoT-based system.

IoT device: IoT devices are embedded with sensors, actuators, radio interfaces, operating systems, lightweight services. An IoT device uses sensors to collect contextual information, and performs actions using the actuators according to the sensed information. For example, a smart thermostat perceives room temperature and humidity, and adjusts the air conditioner's temperature accordingly.

Coordinator/Cluster head: A group of smart things operate under a single Coordinator, which monitors the health and activities of the smart things, and aggregates actions and events. For example, in a smart home a motion sensor camera detects human activities and sends an unlock command to a smart door lock. The door lock receives the command and opens the door. The Coordinator aggregates the sequence of actions and events to generate a report, and sends the report to the IoT service provider, such as network management service and security management service. The information can later be used for Big data analysis, network management, and auditing purposes.

IoT gateway: An IoT gateway is a multi-protocol device that bridges a constrained network (e.g. 6LowPAN) with an external network (cloud/Internet). The gateway device enables IoT devices with heterogeneous networking protocols to communicate each other. For instance, a gateway device enables communication between ZigBee IoT devices and Z-Wave IoT devices. The gateway translates a Zigbee packets

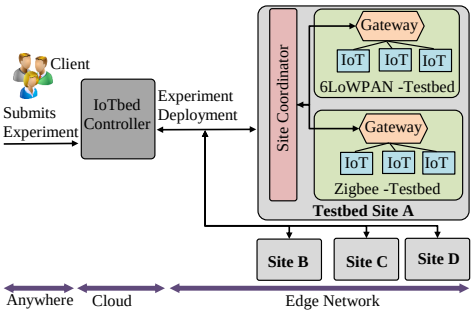


Fig. 2: System overview.

into the Z-Wave packet and forwards the translated packet to the Z-Wave network, and vice versa.

IoT service: There are two major categories of IoT services: Local IoT service and Cloud IoT service. Local IoT services are located in the constrained network and are offered by the smart devices, such as thermostat, door lock, and light. On the other hand, Cloud IoT services are hosted on the Internet and are offered by IoT service providers. A gateway device enables communications between the Local and cloud IoT services. Cloud IoT services enable users to access IoT devices remotely; i.e., a user can query a smart device from anywhere. Cloud IoT services are also responsible for IoT process automation, device management, and decision making task.

End users/clients: They are the consumers of the IoT services. For example, a user issues a command to an IoT device by using a smart phone, laptop, or web-browser.

IV. IoTbed Architecture

We present a high level architecture of IoTbed in Figure 2. Testbed providers set up experimental environments in the edge of their respective networks. A provider can have experimental setup for various constrained networks, such as 6LoWPAN, Zigbee, Z-wave, KNX and so on. These testbeds operate under a Site Coordinator who is responsible for the deployment, monitoring, and controlling experiments in the edge networks. The IoTbed Controller aggregates resources from various sites,that are located in different regions of the globe and provides an unified view of the testbeds. Testbed clients submit their experiments to be run on the edge devices through the interfaces provided by the IoTbed Controller. Figure 3 illustrates the detailed design of IoTbed. The components of IoTbed are as follows:

Management and Operation Control (MOC): The MOC enables a testbed client to explore resources provided by the IoTbed Controller. The client can query for the details of resources, such as hardware and software specifications of sensors, actuators, and Gateways along with network topologies (mesh, start, and tree), devices' mobility properties, testbeds' geo-locations, and so on. The MOC provides interfaces to submit experiment specifications and to upload application binaries. The MOC also allows users to interact with on-going experiments and to receive and analyze experiment data, such as energy consumption, communication latency, computation cost, storage requirement, memory overhead, and so on.

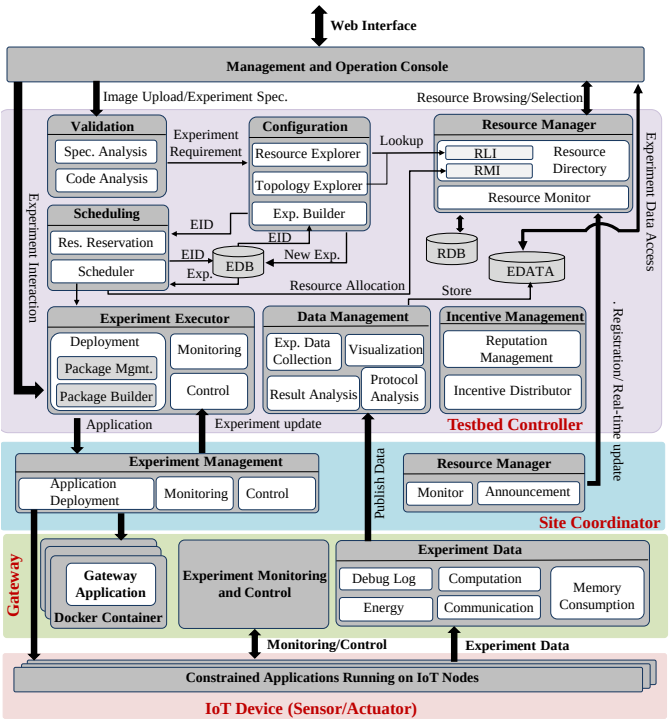


Fig. 3: Component details.

Resource Manager: The Resource Manager monitors and collects available resources in the Testbeds. The Resource Manager consists of Resource Directory Service (RDS) and Resource Monitoring Service (RMS). The RDS implements two interfaces to provide access to resources: Resource Lookup Interface (RLI) and Resource Managing Interface (RMI). The RLI provides information about available and allocated resources in the system. The RMI is used to allocate resources for experiments from the available resource pool and to return resources back to the resource pool when experiments are completed. The RMS manages resources in the resource database (RDB) by keeping the real time information on the availability of resources in the system. A Site Coordinator implements a resource discovery mechanism that announces available resources in the Testbeds that are located in edge networks. The RMS receives announcements and makes changes in the RDB accordingly – registers or removes resources from the RDB.

Experiment Manager: The Experiment Manager implements five components: Validation Service (VS), Configuration Service (CS), Scheduling Service (SS), Experiment Executor (EE), and Data Management Service (DMS).

The VS receives an experiment definition (ED) for a client. The ED contains an experiment specification and a description of use cases of the experiment. The use cases could be represented using standardized syntaxes such as UML. The VS validates the Exp-Spec against a set of experiment policies that are defined by the IoTbed Controller and Site coordinators. For example, the Testbed Controller does not accept experiments that scan ports in a constrained network or broadcast messages in the network at an interval that is lower than the value defined by a Site Coordinator. Policies are defined to protect resources

from malicious activities as well as to discard experiments that are not feasible to run on IoT devices. The VS also revives application image/binary of the experiment. The VS analyzes application images to ensure that they do not include malicious code segments such as malware. All the segments of an experiment are verified and any detected irregularities are reported before any further processing.

The CS explores the RDB by using RLI and selects resources and topologies that match the experiment specification. The CS creates a standardized representation of the validated experiment that includes the descriptions of testbeds, which will be used to execute the experiment, and stores it in an experiment database (EDB). A Testbed's details comprises of devices' description, Gateways' configurations, communication protocols, routing algorithms, network topologies, and Site Coordinators' locations.

The SS reserves resources for an experiment, and then schedules the experiment to be run on the resources using the Provision component for appropriate testbeds. If the SS finds that the selected resources are busy in participating in other experiments then the experiment is queued for the resources until it becomes available.

The EE comprises of three services: Deployment, Monitoring, and Control. The Package Management component of the Deployment service collects all the necessary information that is required to run an application on the Gateway and IoT nodes, which includes required libraries, source files, configurations, policies, and dependencies on other artifacts. The Package Builder component creates the actual applications to be deployed to Gateways and IoT nodes. The Package Builder receives application and device specific information and builds two types of application packages – one for Gateway and another for IoT devices.

The Experiment Monitoring service collects information about the an on going experiment status, such as progress of the experiment, and makes them available to clients. The Experiment Control service implements a set of actions, such as Start, Stop, Pause, Resume, and Restart, that are invoked to issue command to Gateways and IoT nodes.

The DMS aggregates real time experiment data from the testbeds and stores them in the experiment database (EDATA). The DMS provides tools to visualize and analyze experiment results. The DMS can also provide analytics based on IoT data.

Reputation and Incentive Framework (RIF): The RIF manages Reputation Points of IoT nodes and Gateways. The RIF updates the Reputation Point (RP) of a device once the experiment that the device is currently participating in is completed. A new RP of a device is calculated based on it performance in the experiment. The RIF also offers monetary incentives to the devices when an experiment is completed.

V. Operational Model

A. Reputation Point Calculation

Each device is denoted by D_j, $1 \leq j \leq |D|$. The IoTbed Controller maintains for each device D_j a corresponding reputation point r_j such that $r_j \in [0.1,\ 1.0]$. Initially, each device is assigned an equal reputation-point $r_j = 0.5$. Then, depending on how efficient each device to complete her experiment the IoTbed Controller adjusts her reputation point. After each experiment, the IoTbed Controller computes a new reputation point as:

$$r_j^{t+1} = \begin{cases} r_j^t & \text{if the device does not participate in an experiment} \\ r_j^t + \mu & \text{if the device participates in an experiment and performs well} \\ r_j^t - \mu & \text{if the device participates in an experiment but does not perform well} \end{cases}$$

where r_j^{t+1} and r_j^t denote the value of the reputation point for two consecutive experiments, and $\mu \in [0.1,\ 1.0]$ is a constant and denotes a performance index. To this end, the IoTbed Controller updates the reputation point as follows:

$$r_j = \min\ \{\frac{r_j^{t+1} + r_j^t}{2},\ 1\}$$

The IoTbed Controller assigns a device μ according to its performance in an experiment. For example, a device completes an experiment smoothly and receives μ. However, the same device might receives a negative performance index ($-\mu$) in another experiment for performing poorly. For example, the Gateway has to reset the device multiple times since it stops responding in the middle of the experiment. As a result, the experiment requires multiple attempts to be completed; hence, the total time to complete the experiment is increased. Every time a device stops working in the middle of an on going experiment, either the Site Coordinator or the Gateway has to restart the experiment from the very beginning. That means testbeds will process fewer number of experiments in a day and earnings of the IoTbed Controller will be affected negatively. It is also possible that resources are allocated for an experiment from multiple testbeds. Therefore, poor performing devices are offered negative μ.

B. Incentive Calculation

The IoTbed Controller estimates a budget B for an experiment and offers incentive I_j to D_j when the experiment is completed as:

$$I_j = \frac{B}{|D|} \times r_j$$

B is calculated based on the requirements and behaviors of an experiment, such as experiment time, number of time to repeat the experiment, number of nodes to participate in the experiment, the types of the participating nodes (sensor or actuators), bandwidth usages, power consumption, user interfacing, and so on.

C. Testbed Registration

A testbed provider registers her devices to the IoTbed Controller Testbed Controller so that devices can participate in

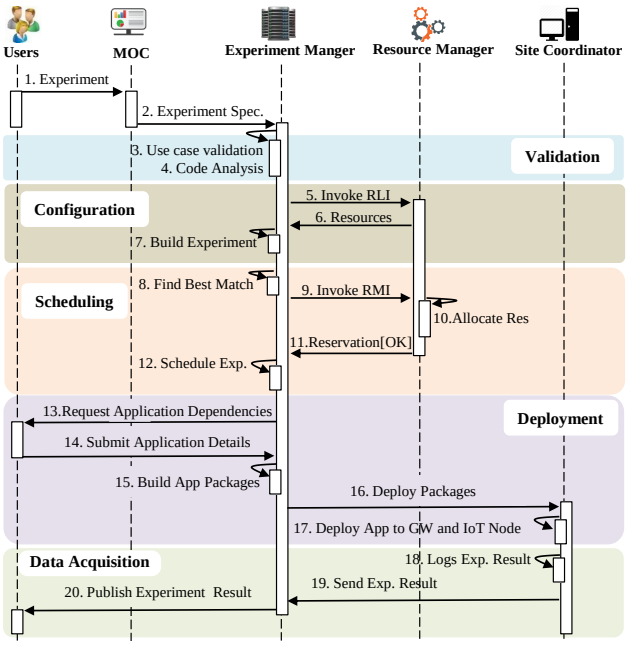


Fig. 4: Experiment submission process.

experiments. The provider installs the Site Coordinator service on a Computer in the edge network. The provider then connects Gateway devices to the Site Coordinator and IoT nodes to the Gateways. Next, the provider provides devices' software and hardware specifications and networking information, such as communication media, routing protocols, topological properties, etc., to the Site Coordinator. The Site Coordinator sends these information to the Resource Manager. The Resource Manager updates the RDB accordingly.

D. Experiment Submission Process

The process to run an experiment of on testbeds is described below (see Figure 4).

Step 1–2: A client submits the details of an experiment to the Experiment Manager through the MOC interface. Table I shows the message format for defining an experiment specification. **Step 3–4:** The VS verifies the use cases of the experiment against a predefined use case policies. Next, the VS ensures that the experiment images or binaries do not contain malicious code segments that can be harmful for the resources running the experiment. **Step 5–7:** The CS invokes the RLI interface to retrieve resources that are available in the system. The CS creates a resource pool by selecting the resources that meet requirements as mentioned in the experiment specification. The CS creates a new experiment description of the submitted experiment that includes the details of the selected resources and stores it to the EDB. The SS and EE use this description to schedule and deploy the experiment respectively. **Step 8–12:** The SS makes a reservation for the resources for which the queue waiting time of the experiment is minimum. It is possible that resources that are the best match for the experiment are occupied by another experiment. Resources are reserved from multiple testbeds in case a single testbed cannot provide all the required resources. Algorithm 1 presents the resource allocation procedure. The EE processes the experiment when the waiting time expires. **Step 13–17:** The ES receives all the necessary items (see Table I), such as source files, libraries, runtime, platform information and so on, that are required to run the experiment on IoT nodes and Gateways from the user. The ES builds application packages and sends them to the Site Coordinator. The Site Coordinator deploys the application to the IoT Node as is. However, the Site Coordinator spins up a container, such as Docker [12], in the the Gateway and installs the application on the container. Thus, a Gateway can run mutilate experiments simultaneously. **Step 18–20:** The Site Coordinator collects experiment data from the testbeds and sends to the DMS. The DMS publishes the result to the client.

TABLE I: IoTbed Message Formats

(a) Experiment Specification

EXPERIMENT SPEC → [USECASE, DEVSPEC, APPSPEC]
USECASE → [List[EVENT]] **EVENT** → [List[ACTOR], List [ACTION]] **ACTOR** → [GATEWAY, IOTNODE] **ACTION** → [GET, POST]
DEVSPEC → [DEVTYPE, DEVINFO, DEVCAPABILITY, DEVDETAIL, NETWORKSPEC] **DEVTYPE** → [GATEWAY, IOTNODE] **DEVINFO** → [MAKER, MODEL] **MAKER** → [ARDUIONO, RASPBERRY PI, TMOTE SKY, ⋯] **DEVCAPABILITY** → [MOBILE, STATIONARY] **DEVDETAIL** → [HARDWARESPEC, SOFTWARESPEC] **HARDWARESPEC** → [CPU, RAM, ROM, RADIO] **SOFTWARESPEC** → [OS, PLATFORM, CONFIGURATION] **NETWORKSPEC** → [TOPOLOGY, ROUTING, INTERFERENCE] **TOPOLOGY** → [STAR, MESH, TREE]
APPSPEC → [LIBRARY, SOURCE FILES, POLICIES, CONFIGURATIONS] **LIBRARY** → [List[APPLICATION DEPENDENCY]]

(b) Example Use Case Scenarios

EVENT TYPE: Room Temperature Adjustment **DESCRIPTION:** Adjust room temperature by using a smartphone
ACTOR$_1$: Thermostat, **ACTOR$_2$:** Smartphone **ACTION:** Smartphone → Thermostat: POST /thermostat/set-temperature/value
EVENT TYPE: Locking Main Door **DESCRIPTION:** A smart door-lock locks the main door when no one is present in the home
ACTOR$_1$: Smart Door Lock, **ACTOR$_2$:** Motion Sensor **ACTION:** Door Lock → Motion Sensor: GET /motion-sensor/room-no/residence-presence-status
EVENT TYPE: TV Screen Brightness Adjustment **DESCRIPTION:** A smart TV adjusts the brightness of its screen for better visibility according to the amount of ambient light presents in a room
ACTOR$_1$: Smart TV, **ACTOR$_2$:** Ambient Light Sensor **ACTION:** Smart TV → Light Sensor: GET /light-sensor/ambient-value

VI. EXPERIMENTAL EVALUATION

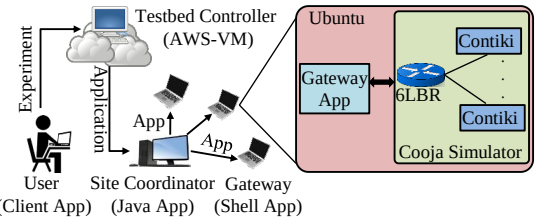
A. Experimental Setup

Prototype implementation: We implemented a prototype of IoTbed that ran on various types of IoT devices, such as Wismote, Open Mote, and TMote Sky, powered by Contiki

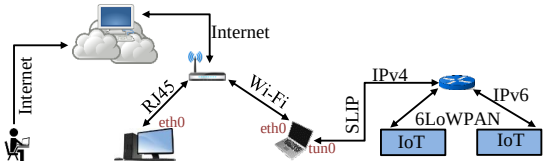
Algorithm 1: Testbed Reservation

Input: ExpSpec, DevCount
Output: Testbeds

```
testBeds = getTestbedsBy(ExpSpec.DevSpec, ExpSpec.AppSpec)
var intermediateTestBeds = NULL
for T in testBeds do
    if T.GWs == DevCount.GW and T.Nodes == DevCount.Node then
        intermediateTestBeds.push(T)
    end if
end for
if intermediateTestBeds not NULL then
    /*At least one Testbed can provide required resources*/
    var targetTestBeds = new TestBeds ()
    for T in intermediateTestBeds do
        if T.isBusy equals TRUE then
            targetTestBeds.push(T)
            targetTestBeds ⟵ sort targetTestBeds by time requires to
            become ready to participate in the experiment
        else
            return T
        end if
    end for
else
    /*Resources should be allocated from multiple testbeds*/
    var targetTestBeds = new TestBeds ()
    intermediateTestBeds ⟵ sort intermediateTestBeds by geolocation
    intermediateTestBeds ⟵ sort intermediateTestBeds by time
    requires to become ready to participate in the experiment
    targetTestBeds ⟵ select testbeds from intermediateTestBeds such
    that ∑targetTestBeds.GWs == DevCount.GW ∧
    ∑targetTestBeds.nodes == DevCount.nodes
end if
return targetTestBeds
```



(a) Experimental Setup. 6LBR = 6LoWPAN Border Router.



(b) Network Setup

Fig. 5: Experimental Environment

Operating System [11]. The IoT devices were simulated on Cooja simulator. The experiment setup for the proof-of-concept is shown in Figure 5a. We implemented the Testbed Controller using NodeJS. The IoTbed Controller was running as a Web Service on a Virtual Machine located in Amazon Cloud (AWS-VM). The Site Coordinator was a Java application running on a Desktop PC. The Gateway application was implemented using Shell Script and was running on a Laptop. Applications for the IoT nodes were implemented using C. The Cooja simulator was installed and running on the Laptop. Table II shows the hardware and software specifications of devices that were used for the prototype.

Network setup: Figure 5b illustrates the network architecture

TABLE II: Experimental Devices' Specifications

TESTBED CONTROLLER SPECIFICATION
HARDWARE → Amazon m3.medium instance with 4 vCPUs, 2.8GHz processor, 3.75 GB memory, and 8GB hard drive with moderate network performance
SOFTWARE → **OPERATING SYSTEM:** Ubuntu 14.04 LTS, **RUNTIME:** NodeJS
SITE COORDINATOR SPECIFICATION
HARDWARE → Desktop computer with 2 Quad CPU, 2.83 GHz, 12 GB memory, and 500 GB hard drive
SOFTWARE → **OPERATING SYSTEM:** Windows 10, **RUNTIME:** JVM
GATEWAY SPECIFICATION
HARDWARE → Laptop with Core i3 CPU, 2.2 GHz, 8 GB memory, and 1 TB hard drive
SOFTWARE → **OPERATING SYSTEM:** Ubuntu 16.10, **RUNTIME:** Bash
IOT DEVICE SPECIFICATION
DEVICE → TMote Sky, Open Mote, Wismote, RE Mote
HARDWARE → 8 ∼ 64MHz CPU, 8 ∼ 128KB RAM, 10KB∼ 1MB ROM
SOFTWARE → **OPERATING SYSTEM:** Contiki 3.1, **RUNTIME:** C

of our prototype. The Site Coordinator was communication to the Testbed Controller over IPv4. The communication medium between the Site Coordinator and a Gateway was WiFi. However, the Gateway was connected to an IoT Border Router through a SLIP interface [13] and was communicating to the Router over IPv4. The IoT nodes formed a 6LoWPAN network to communicate among themselves.

Request generation: To simulate testbed clients, we implemented an application the generated different types of requests, that were submitted to the IoTbed Controller, by varying the parameters of the experiment specification (see Table I for specifications). Each client ran a task, which is assumed similar to performing actions in a use case or running an experiment in an IoT node, after getting access to its specified IoT device and released that device once the task was completed. We assumed that a client can run any of the following six tasks – word count, sorting, encryption, decryption, compress, and decompress. Each of these tasks received a file of maximum size 10MB as input. We created a runtime profile for each of the tasks for various types of IoT device, such as Wismote, RE-Mote, and TMote Sky, as follows; for each of those six problems, we varied the file size, and determined the runtime for executing the problem in one of our specific IoT device. We repeat the procedure for other types of IoT devices and noted the time. Initially, we assumed that our system can have only one types of IoT device and all clients requested for that types of device. We measured the average waiting time for a client to access its specified IoT device after sending the request. The corresponding graph is shown in Figure 6.

B. Simulation Results

Next, we increased the total different types of device from 1 to 5 and 5 to 10, and measured the average waiting time. The corresponding figure is shown in fig 7 and fig 8 respectively.

From Figure 6, 7, and 8, we see that when we increase the total number of tasks or requests, the average waiting time in increasing as well. This is because there are more clients' requests than the total available devices. Therefore, some clients

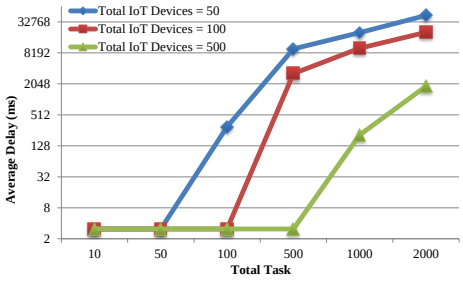


Fig. 6: Request for a single type of IoT Device

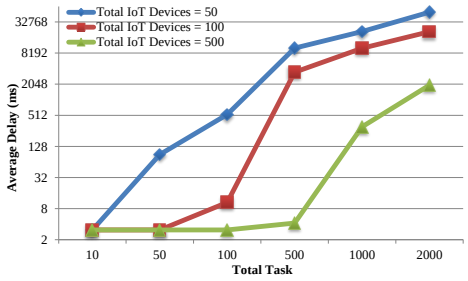


Fig. 7: Request for 5 different types of IoT Device

had to wait until a client finished its tasks and released the IoT device. Moreover, if we observe Figure 6, 7, and 8, we can find that increasing the total number of IoT devices reduces the average waiting time. The reason is that increasing total number of devices increases the chances that an IoT device of a client specified configuration will be available to run a task upon the time of the request.

Finally, we can also notice from the three figures that the average waiting time actually increases when we considered more variety of IoT devices. For example, we can see that the average delay for 100 IoT device of a single type is much lower than the average time for 5 different types of IoT devices for 500 number of requests. Since we increase the type without increasing the total number of devices, some specific type of device might receive more request for accessing than some other. Therefore, although we might have a decent number of free IoT device, clients might have to wait more for getting their specified types of IoT devices.

VII. Comparison of IoTbed with Prior Research

With the recent growth of IoT, researcher have proposed solutions for IoT testbeds. FIT Iot-LAB [14], a very large scale open testbed, is composed of more than 2700 hundred low-power connected wireless nodes and 117 mobile robots available for experimenting with large-scale wireless IoT technologies. As a lot of commercial vendors and educational institutes are involved with IoT R&D, they have their own testbed platforms and facilities. FIESTA-IoT [15] is a concept architecture for interconnecting and sharing such testbeds among stakeholders.

IoT devices most often communicate with each other via low powered wireless sensors.There have been numerous testbed facilities using wireless sensor networks as the main component of their backbone infrastructure. SensLAB [16], a very large scale open wireless sensor network testbed, had been developed

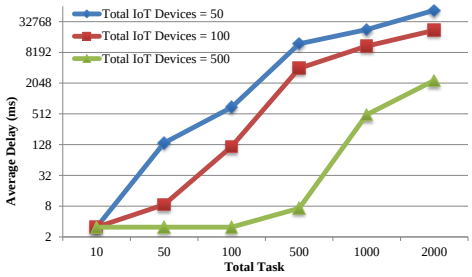


Fig. 8: Request for 10 different types of IoT Device

and deployed in order to allow the evaluation of scalable wireless sensor network protocols and applications. Syndesi [17] is a framework for unifying heterogeneous devices and services in the WSN domain, which can be used to develop testbeds. Vlado et al. presented TWIST [18], scalable and flexible testbed architecture for indoor deployment of wireless sensor networks. WISEBED [19] is a testbed architecture that allows operators of WSN to offer numerous users access to their testbeds in a standardized flexible way and also matches user and operator requirements. w-iLab.t [20] is a wireless testbed with 200 sensor nodes and equal number of Wi-Fi nodes supports wireless sensor experiments, Wi-Fi based mesh and ad hoc experiments, and mixed sensor/Wi-Fi experiments.

Several other testbed prototype are proposed for specific scenario related to IoT. Berhanu [21] et al. described the setup of a real life testbed to evaluate accurately adaptive security for the IoT using current commercial off-the-shelf products and open source software. Choosri [22] et al. proposed a prototype for IoT-RFID testbed. Pozza [23] et al. introduced SmartEye an energy efficient observer platform for IoT testbeds. Shachar [24] et al. proposed a set of requirements and an architecture for building a security testbed for IoT. Constantinos et al. [25] presented an IoT testbed architecture for Smart Buildings that enables the seamless and scalable integration of crowd-sourced resources such as smartphones and tablets.

To compare our approach IoTbed to existing works, FIT Iot-LAB, FIESTA-IoT and WISEBED are similar to ours. FIT Iot-LAB provides command line and web interface to interact with the platform. FIESTA-IoT provides tools, techniques, processes and best practices enabling IoT testbed/platforms operators to interconnect their facilities in an interoperable way based upon cutting edge semantics-based solutions. WISEBED is an architecture that is used by several WSN testbed which allows the operators to provide testbed service via web service to the users. We compared our approach with these based on following requirements:

- Heterogeneity: The testbed architecture should support the use of multiple sensing technologies and the integration of multiple communications schemes.
- Scalability: The testbed should guarantee that the algorithms and applications do not behave unstable in large scale.
- Federation: The support to repeat experiments in the same conditions is needed to improve the statistical accuracy of the results provided by a testbed.
- User Interfacing: Simple tools should be available to support configuration, programming and logging functionalities.

TABLE III: Comparison of IoT Testbeds

	Heterogeneity	Scalability	Federation	User Interfacing	Resource Allocation	Registration	Reputation Scheme	Incentive Framework	Security Policy
FIT Iot-Lab	✓	✓	✓	✓	✗	✗	✗	✗	✗
WISEBED	✓	✓	✓	✓	✓	✗	✗	✗	✗
FIESTA-IoT	✓	✓	✓	✓	✓	✓	✗	✗	✗
IoTbed	✓	✓	✓	✓	✓	✓	✓	✓	✓

- Registration: A well designed and easy process should be available for testbed service provider for adding their testbed to the network.
- Resource Allocation: An user should able to allocate resources from multiple testbed if one testbed is not sufficient enough.
- Reputation Scheme: A point based system for device rating based on its performance in the experimentation.
- Incentive Framework: The architecture should also provide an incentive framework so that a budget can be estimated for a particular experiment.
- Security Policy: The framework should have policies to protect resources from malicious activities.

Based on these requirements, we carried out a comparative analysis among these testbed architectures. The results of this analysis are in Table III. Here, we find that IoTbed satisfies all 9 criteria, while others satisfy at most 6 of them.

VIII. Conclusion

The proposed IoTbed model introduces a novel architecture for Testbed as a Service for distributed testbed providers. IoTbed enables providers to develop testbeds using smart devices in the edge networks, that are distributed across the globe, and to rent out these testbeds for Internet of Things (IoT)-based experiments. As a result, users can test IoT-enabled applications on a large number of devices with heterogeneous specifications and configurations prior to deploying them to the real word IoT-based systems. IoTbed presents an economic model in which users have to pay to use the testbed service and testbed providers are offered monetary incentives. This will encourage existing providers to maintain Quality of Services of their testbeds and will attract prospective provides to rent out their smart devices. Moreover, this model is cost effective for users since they might not afford, or it could be inappropriate for them, to develop and maintain such large scale testbeds. To this end, we provided a proof of concept implementation of IoTbed using Contiki powered IoT devices to demonstrate the feasibility of our proposed system.

Acknowledgments

This research was supported by the National Science Foundation CAREER Award CNS-1351038 and ACI-1642078.

References

[1] X. Li, R. Lu, X. Liang, X. Shen, J. Chen, and X. Lin, "Smart community: an internet of things application." *IEEE Communications Magazine*, 2011.

[2] P. Vlacheas, R. Giaffreda, V. Stavroulaki, D. Kelaidonis, V. Foteinos, G. Poulios, P. Demestichas, A. Somov, A. R. Biswas, and K. Moessner, "Enabling smart cities through a cognitive management framework for the internet of things," *IEEE communications magazine*, 2013.

[3] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, 2014.

[4] K. Su, J. Li, and H. Fu, "Smart city and the applications," in *Proc. of the ICECC*. IEEE, 2011.

[5] A. Zamanifar, E. Nazemi, and M. Vahidi-Asl, "Dmp-iot: A distributed movement prediction scheme for iot health-care applications," *Computers & Electrical Engineering*, 2016.

[6] K. M. Alam, M. Saini, and A. El Saddik, "Toward social internet of vehicles: Concept, architecture, and applications," *IEEE Access*, 2015.

[7] www.gartner.com, "The internet of things will transform the data center," 2014. [Online]. Available: http://www.gartner.com/newsroom/id/2684616

[8] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer networks*, 2010.

[9] www.idc.com, "Finding success in the new iot ecosystem: Market to reach $3.04 trillion and 30 billion connected things in 2020," 2014. [Online]. Available: http://www.idc.com/getdoc.jsp?containerId=prUS25237214

[10] Y. Huang and G. Li, "A semantic analysis for internet of things," in *ICICTA*. IEEE, 2010.

[11] "Contiki Operating System," http://www.contiki-os.org/.

[12] "Docker," https://www.docker.com/.

[13] "Serial Line IP (SLIP)," http://contiki.sourceforge.net/docs/2.6/a01693.html.

[14] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele *et al.*, "FIT IoT-LAB: A large scale open experimental IoT testbed," in *Proc of the WF-IoT)*. IEEE, 2015.

[15] A. Gyrard and M. Serrano, "Fiesta-iot: Federated interoperable semantic internet of things (iot) testbeds and applications," *ICT*, 2015.

[16] C. B. Des Rosiers, G. Chelius, E. Fleury, A. Fraboulet, A. Gallais, N. Mitton, and T. Noël, "Senslab," in *TridentCom*. Springer, 2011.

[17] O. Evangelatos, K. Samarasinghe, and J. Rolim, "Syndesi: A framework for creating personalized smart environments using wireless sensor networks," in *DDCOSS),*. IEEE, 2013.

[18] V. Handziski, A. Köpke, A. Willig, and A. Wolisz, "Twist: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks," in *International workshop on Multi-hop ad hoc networks: from theory to reality*. ACM, 2006.

[19] I. Chatzigiannakis, S. Fischer, C. Koninis, G. Mylonas, and D. Pfisterer, "Wisebed: an open large-scale wireless sensor network testbed," in *International Conference on Sensor Applications, Experimentation and Logistics*. Springer, 2009.

[20] S. Bouckaert, W. Vandenberghe, B. Jooris, I. Moerman, and P. Demeester, "The w-iLab. t testbed," in *TRIDENTCOM*. Springer, 2010, pp. 145–154.

[21] Y. Berhanu, H. Abie, and M. Hamdi, "A testbed for adaptive security for iot in ehealth," in *International Workshop on Adaptive Security*. ACM, 2013.

[22] N. Choosri, Y. Park, S. Grudpan, P. Chuarjedton, and A. Ongvisesphaiboon, "Iot-rfid testbed for supporting traffic light control," *International Journal of Information and Electronics Engineering*, 2015.

[23] R. Pozza, A. Gluhak, and M. Nati, "Smarteye: an energy-efficient observer platform for internet of things testbeds," in *WiNTECH*. ACM, 2012.

[24] S. Siboni, A. Shabtai, N. O. Tippenhauer, J. Lee, and Y. Elovici, "Advanced security testbed framework for wearable iot devices," *TOIT*, 2016.

[25] C. M. Angelopoulos, O. Evangelatos, S. Nikoletseas, T. P. Raptis, J. D. P. Rolim, and K. Veroutis, "A user-enabled testbed architecture with mobile crowdsensing support for smart, green buildings," in *ICC*, 2015.