IGaaS: An IoT Gateway-as-a-Service for On-demand Provisioning of IoT Gateways

Mohammad Aminul Hoque
Dept. of Computer Science
University of Alabama at Birmingham
Birmingham, AL 35294, USA
mahoque@uab.edu

Mahmud Hossain
Dept. of Computer Science
University of Alabama at Birmingham
Birmingham, AL 35294, USA
mahmud@uab.edu

Ragib Hasan
Dept. of Computer Science
University of Alabama at Birmingham
Birmingham, AL 35294, USA
ragib@uab.edu

***Abstract*—The widespread adoption of the Internet of Things (IoT) devices has increased its popularity and usage in diverse dimensions, including smart city, home, healthcare, and vehicles. The pervasiveness of the number of IoT devices that operate in low power and lossy network leads to performance issues. An excessive amount of IoT devices that operate with a fixed number of gateways reduce the quality of service (QoS) due to the increased latency of routing messages between the source and destination sensors. In this paper, we propose an IoT Gateway as a Service (IGaaS) that enables on-demand provisioning of IoT Gateways to maintain and improve QoS in an IoT system with a significant number of sensors. The IGaaS allows both the stationary and mobile gateways to be provisioned on-demand. The mobile devices, such as smartphones and drones, provide gateway services in exchange for incentives. The IGaaS supports both the upscale and downscale of IoT gateways depending on various metrics and requirements. The experimental results show that the IGaaS improves the QoS in terms of latency and power consumption.**

***Index Terms*—internet of things, gateways, provisioning, downscaling, 6LoWPAN**

I. Introduction

The Internet of Things (IoT) is a technology paradigm that is envisioned as a global network of machines and devices capable of interacting with each other. It is anticipated that there will be more than 50 billion IoT devices by 2020 [1]. In the concept of IoT, billions of physical devices are connected through the internet, and they are capable of collecting and sharing data. IoT devices have gained huge popularity in numerous systems, such as smart homes, smart healthcare [2] [3], and connected vehicles [4] where these devices can communicate and interact among themselves as a cyber-physical system. In the IoT network, the IoT devices form a Directed Acyclic Graph (DAG) to route the packets. A gateway acts as the root of the DAG which is responsible for sending the sensor data from IoT devices to the cloud and vice verca. The number of IoT devices can vary from time to time in the IoT environment as the mobile IoT devices can join or leave the network anytime. With the increase of IoT nodes, the size of the DAG will increase. A larger graph may increase communication overhead and security vulnerabilities in the IoT network. Breaking down a larger graph into several smaller ones by dynamically provisioning gateways will help to operate the IoT network with optimized performance. Hence, we need a framework that provisions gateway devices on-demand by considering various properties of the IoT network.

There are several challenges with provisioning IoT gateways on-demand. First, we need to identify when such a kind of IoT gateway provisioning is required. Moreover, we also need to determine when the service is no longer required after identifying an over-provisioned scenario. Second, for static network architecture, we need to decide which gateways should be turned off to downgrade the network and merge multiple smaller DAGs into a single one. Third, the IoT gateways are required to be provided on demand. Fourth, there will have to be an incentive model for service providers (owners of the smartphones and drones) who will rent out their devices to serve as IoT gateways. All these unique issues make this problem interesting and hard to solve.

In this paper, we propose an **I**oT **G**ateway **a**s **a** **S**ervice (IGaaS) that provides on-demand gateway service using smart mobile devices, such as smartphones or drones. The mobile gateways (M-Gateway) have a dual network interface: WLAN (IEEE 802.11) and IoT (IEEE 802.15.4). An M-Gateway communicates with IoT nodes and clouds using its IoT and WLAN interfaces, respectively. The owners of smartphones and drones can rent out their devices as an M-Gateway in exchange for incentives. We propose a rating based incentive mechanism to encourage smartphone and drone owners to rent out their devices through the IGaaS. We demonstrate the feasibility of our proposed framework through a proof-of-concept implementation of the IGaaS on Contiki powered IoT devices. We provide an experimental evaluation of IGaaS that shows IGaaS can reduce communication latency and power consumption in IoT devices when an IoT network inundates with numerous IoT nodes.

Contribution: The contribution of this paper is summarized as follows:

1) We have presented IoT Gateway as a Service (IGaaS) to provide on-demand gateway provisioning for different crowd-sourced environment.
2) We have proposed an incentive model for the devices that serve as the gateways.

Organization: The rest of the paper is organized as follows: Section II and III provide details on the motivation and proposed framework. The operational model and experimental results are presented in Section IV and V. Finally, we conclude in section VI.

978-1-7281-5503-6/20/$31.00 ©2020 IEEE

II. Background And Motivation

The operation of IoT devices are conducted in a constrained low powered and lossy network. The devices can use multiple protocols for communications such as Zigbee [5], 6LoWPAN [6], ZWave [7], and BLE [8]. An IPv6 routing protocol named RPL [9] is used to route the packets over the network. The 6LoWPAN nodes communicate over a wireless network defined by IEEE 802.15.4 standard [10]. 6LoWPAN network has three type of nodes: root node, intermediate node, and leaf node. The root node is responsible for maintaining the communication with other networks. The intermediate nodes forward packets to the root node and leaf nodes. Construction of the network topology is based on the DAG concept where every node selects a neighbour as its parent based on an objective function. RPL organises the 6LoWPAN nodes as Destination Oriented DAGs (DODAG) [11] for point-to-point communication. Here, a root node is responsible for starting the construction of the network and every node chooses its parent through the RPL. A set of control messages are defined by RPL for DODAG formation. A RPL node broadcasts a DODAG Information Solicitation (DIS) message to request DODAG Information Object (DIO) messages from nearby RPL nodes. Upon receiving reply from RPL nodes, the new node uses the DIOs for selecting its parent(s). Finally, the node sends a Destination Advertisement Object (DAO) to the root so that the intermediate nodes can update their routing table.

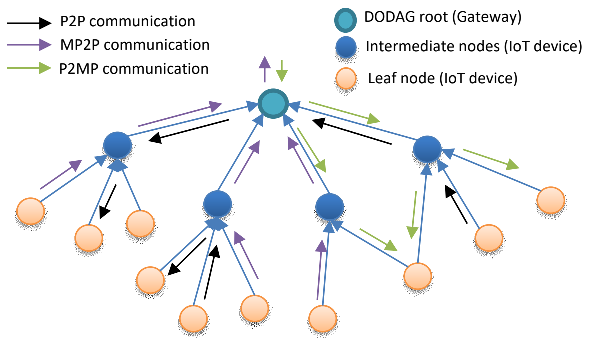

Fig. 1: 6LoWPAN traffic

In DODAG, a gateway works as the root node of the DAG and the IoT nodes work as the intermediate and leaf nodes. There are three type of routing in 6LoWPAN: 1) point to point (P2P), 2) point to multipoint (P2MP), and 3) multipoint to point (MP2P). In P2MP communications, traffic flows from a DODAG root to a subset of 6LoWPAN devices. Traffic flows from 6LoWPAN devices to a DODAG root in MP2P communications. In P2P communications, traffic flows between two 6LoWPAN devices. Figure 1 shows different types of 6LoWPAN traffic. The routing scheme in 6LoWPAN can be divided into two categories: mesh-under and route-over [12]. In mesh-under routing, the node does not contain any routing table to forward packets, rather packets are forwarded to a neighbor node to deliver the packet to the destination over multiple radio hops. On the other hand, in route-over routing, the node holds a routing table and acts as an IP router. In this paper, we are mainly focusing on mesh-under routing.

Problem statement: IoT devices form DAG and use RPL routing to forward packets. An IoT gateway is the root of the DAG and it works as the communication bridge between IoT network and internet. The size (height and width) of the DAG increases as more number of nodes join the IoT network. As a result, the communication latency for exchanging messages and the CPU and memory utilization for routing packets are increased. Hence, the quality of service in an IoT network decreases with the increase of smart nodes in the network.

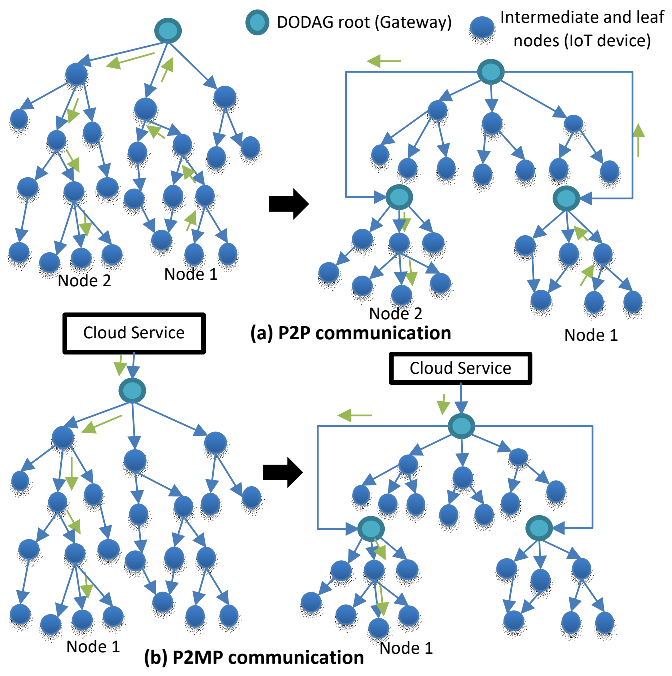

Fig. 2: Example scenarios for P2P and P2MP communication where on-demand gateway provisioning can improve QoS

However, in the existing IoT systems, IoT Gateways are not provided on-demand. The total number of IoT Gateways are fixed for an IoT system regardless of the number of nodes present in the IoT network at a given time. Few gateways with a huge number of IoT devices in the network may degrade the quality of service. Moreover, the over-provisioning of IoT Gateways increases power consumption, which contributes to the increase in the electricity cost or reduces the gateway devices' battery life. Previous research works implemented virtualization technology for IoT gateways. They set up an edge server nearby the IoT devices and implemented container-based virtualization of gateway functionality [13] [14] [15]. These research works require an edge server and proper hardware and software installation for gateway virtualization. To eliminate these requirements, we propose to provision IoT Gateways on-demand through drones for improving the quality of service of an IoT system. We also provide the solution to downscale the number of gateways to reduce unnecessary energy consumption when multiple gateways are not required to maintain the QoS.

Figure 2 shows two scenarios where additional gateways can provide better service. Figure 2(a) shows a P2P communication scenario where node 1 wants to send a packet to node 2. In this case, the packet will route through the gateway and finally delivered to node 2. Incorporation of additional gateway can reduce the height of the DAG to ensure faster delivery of the message. Figure 2(b) shows a P2MP scenario where the cloud service wants to send a message to node 1. Delivering the message may cause significant delay if the graph depth is large. Incorporating additional gateway can improve the service in this scenario. We can show similar scenario for MP2P communication where multiple IoT nodes may want to send data to the cloud. For all type of communication, our target is to send the traffic from source to destination as quickly as possible. Hence, we need to identify when to break down the DAG by incorporating new gateway(s). For this purpose, we need to consider several aspects, such as:

Number of nodes: Total number of nodes are an important metric to figure out the current condition of the IoT network. Huge number of nodes are likely to cause more latency and power consumption.

Depth of the DAG: With the increase of tree depth, a packet will have to travel more nodes to reach the destination. We need to optimize the tree depth and number of new gateways to incorporate for achieving the best advantage from on-demand gateway provisioning.

Breadth of the DAG: If the breadth of the DAG becomes too large, then the gateway will consume more power, CPU cycle, and memory. Moreover, it will have to route more packets due to presence of more IoT nodes which may incur latency issues.

CPU utilization: CPU utilization refers to different utilization metrics of IoT devices such as RAM, storage, bandwidth, CPU cycles etc.

Power consumption: Power consumption by the IoT nodes in a certain period of time is also a important metric to consider for deciding about gateway provisioning.

III. Proposed System Framework

Our proposed system framework has several modules. Each module has some components to perform specific tasks. Figure 3 shows the overview of the proposed system framework. The modules of our proposed framework are as follows:

A. Management portal

The internal components of the proposed framework communicate with users or service providers through the management portal. The users can use the management portal through a web interface for various purposes, such as creating and maintaining service provider profile, registering devices, bidding to a new service advertisement, receive instructions about service, etc.

B. Controller module

The controller module is responsible for the core functionalities of IGaaS. The module sends the advertisement to the registered gateway providers about the requirements of dynamic gateways in a certain IoT network. Moreover, it receives the requirements of new gateways from the gateway of the network. Additionally, the controller also informs the service providers whether the service is no longer required or not. The controller module has the following components:

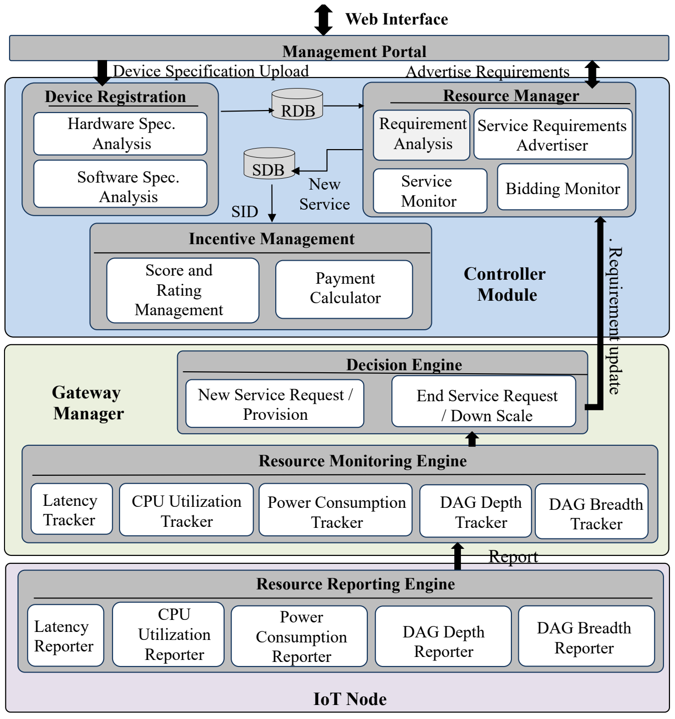

Fig. 3: System framework of IGaaS

Device registration component: The device registration component handles new device registration requests from the users. New devices are registered after exploring the software and hardware specifications to ensure its capabilities to serve as an IoT network gateway. Then details of the resources are stored in the Resource DataBase (RDB) with a unique device id. The device is related to the service provider's profile.

Resource manager: The resource manager receives a new requirement of gateways through requirement status monitor. Upon receiving a new gateway request, the controller broadcasts the advertisement of new gateway requirements. After a bidding procedure, the winner is instructed to provide the service to the intended IoT network. When the service is no longer required, the requirement status monitor informs the service providing party about the end of the service. Service related metrics are updated in the service database (SDB) and the service id (SID) is provided to the incentive management component.

Incentive management component: The incentive management component is responsible for calculating the performance point after the completion of service and thereby calculates the new rating point of the device. The payment calculator module receives the Service ID (SID) from service monitor and retrieves all the details of the service from SDB to calculate new rating. Later, the incentive is calculated using the service details by the payment calculator component.

C. Gateway manager

The gateway manager is responsible for keeping track of the current status of the network and inform the requirements about new mobile gateways to the controller. There are two engines in the gateway component:

Resource monitoring engine: Resource monitoring engine keeps track of the current status of the resources and scenario

of the network such as latency, CPU utilization, power consumption, graph depth, and breadth. These trackers report their data to the decision engine. After analyzing the reports provided by the trackers, the decision engine decides provision or downscale the gateways in the network. The latency monitor gets the latency by probing a packet to the IoT nodes and calculating the round trip time. Additionally, The IoT nodes may also inform the gateway about the required time to receive the response after sending a request. CPU utilization tracker receives reports from each IoT device about the usage of RAM, storage, network bandwidth, and CPU speed. The power consumption tracker tracks the power consumption by each node in the network. The depth of the graph can be tracked by finding the maximum rank reported by the nodes. Finally, the breadth can be tracked by tracking the count of similar ranks reported by the IoT nodes.
Decision engine: The decision engine is responsible for deciding whether new mobile gateways are required or not. It also decides the amount of provisioning or downscaling of IoT gateways for optimized performance in the IoT network.

D. IoT node module

The IoT nodes have a resource reporting engine. The resource reporting engine includes a latency reporter, CPU utilization reporter, power consumption reporter, DAG depth reporter, and DAG breadth reporter. The nodes can report the gateway manager about the delay of receiving a response after sending a request. For all other metrics, the IoT devices can send a periodic update about their CPU utilization, power consumption, rank, etc. to the resource tracking engine.

IV. Operation model

A. Profile creation and device registration

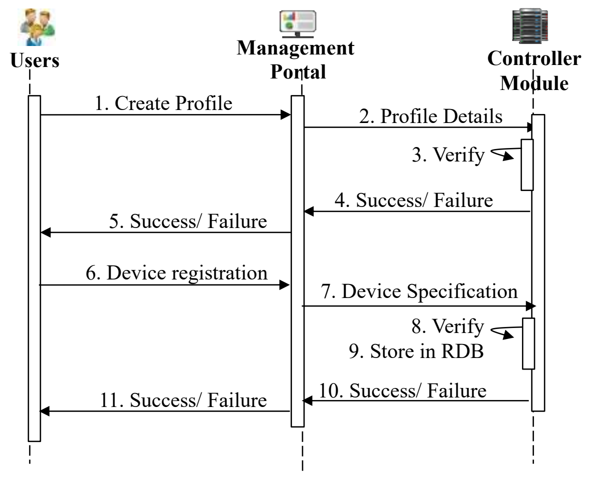

Fig. 4: Profile creation and device registration

Before providing a service, the service provider needs to create a profile and register the devices. A user registers her devices through the management portal by providing hardware and software specifications. We are assuming that the service will be provided by drones or smartphones. All the devices have both the network interface of WLAN (IEEE 802.11) and IoT (IEEE 802.15.4). Figure 4 shows the steps of the profile creation and device registration phase.

Step 1: The service provider creates a profile through the management portal. **Step 2:** The management portal sends the details of the profile to the controller module. **Step 3-5:** The controller module verifies the profile and sends back success/failure message. **Step 6-7:** If the profile is successfully created, the service provider can add the device specifications and the controller module receives the software and hardware specifications of the devices through the management portal. **Step 8-11:** The controller module verifies the devices and decides whether to add them into the RDB using a unique device id and user id. Finally, the controller module sends the success/failure response to the service provider.

B. Dynamic and stationary gateway provisioning

We have considered dynamic gateway provisioning and downscaling for both stationary and mobile IoT network architecture. Figure 5 shows the operation model of the procedure.

Step 1: The IoT nodes send the report about themselves collected by resource reporting engine to gateway manager. **Step 2:** The resource monitoring engine collects reports from all the IoT nodes and sends it to the decision engine. **Step 3:** The decision engine examines the reports and decides whether a new gateway provisioning service is required or the current services are no longer required. **Step 4:** The controller module receives the resource request update from the gateway manager and initializes a new resource allocation procedure or service termination procedure. For stationary gateways, the controller module informs about up-scaling or down-scaling the gateways. **Step 5:** For new resource request, the controller module retrieves the available service providers. **Step 6:** The controller node advertises the resource requirements to the available service providers. **Step 7:** Interested service providers responses by bidding a price considering the cost and service requirements. **Step 8:** The controller node selects the service providers. Algorithm 1 presents the provider selection procedure. **Step 9:** The controller module informs the selected service providers and provides details about the requested service. **Step 10:** The service provider provides service by provisioning gateways through drones or smartphones. **Step 11-13:** When the service is no longer required, the gateway manager informs the controller module, and it is propagated to the service provider. For a stationary network, proper scaling instructions are sent to static gateways. **Step 14:** The service details are provided to the controller module and stored in the service database (SDB). **Step 15:** Rating and incentive are calculated from the quality of the provided service. **Step 16:** Finally, the incentive is paid to the service provider.

C. Score and rating calculation

We denote each device as D_j, $1 \leq j \leq |D|$. The framework maintains a corresponding score s_j and rating point r_j for each device D_j. The maximum and minimum rating point can be 100 and 0 which are denoted by r_{max} and r_{min} respectively. Initially, each device is assigned score $s_j = 0$ and rating

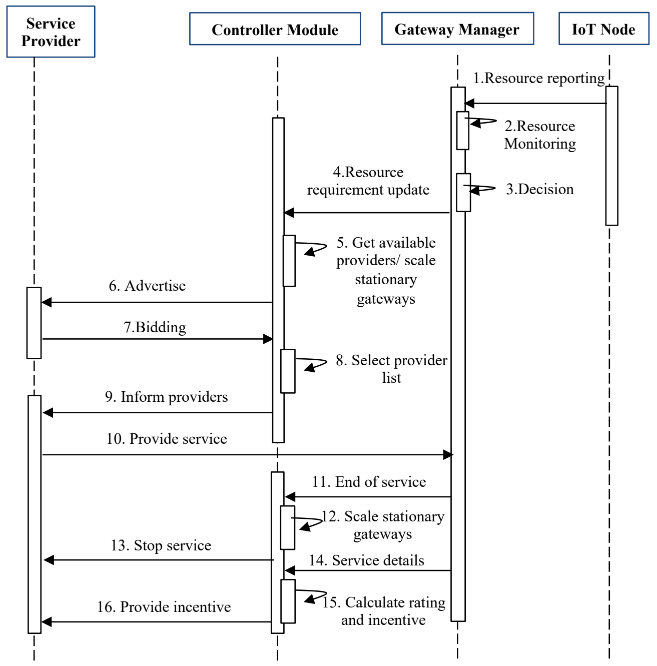

Fig. 5: Resource requirement analysis and gateway provisioning

$r_j = 0$. After completing each service, each participating device receives a performance point p_j after completing a service. Depending on the performance point, the framework updates the score of each participating device and calculates the corresponding rating points. The score and rating point calculation depend on the number of services χ each device has provided. The score is also dependent on rating coefficient μ which is calculated as: $\mu = \frac{r_{max} - r_j^t}{r_{max}}$ where the rating coefficient μ is actually a weight function which makes it harder to increase the score and rating if those are already high. The purpose behind this weight function is to put more responsibility to a well functioning device which has already gained trust of the users. The framework computes the new score of a device as follows:

$$s_j^{t+1} = \begin{cases} s_j^t & \text{if the device does not provide service} \\ s_j^t + ((1+\mu) \times \chi \times p_j) & \text{if the device provides service and } r_j^{t+1} < r_{max} \\ r_j^t \times \chi & \text{if the device provides service and } r_j^{t+1} >= r_{max} \end{cases}$$

where s_j^{t+1} and s_j^t denote the scores for two consecutive experiments, χ is the number of services the device has participated, and μ is the rating coefficient of that device. Initially the score and rating are calculated assuming $r_j^{t+1} < r_{max}$. If that is not the case, then the rating point of a device gets updated if the corresponding score of that device is updated. The framework computes the new rating point as $r_j^{t+1} = \frac{s_j^{t+1}}{\chi}$ where r_j^{t+1} and r_j^t denote the value of the rating point of a participating device for two consecutive services.

The rating of a service provider depends on the average of rating points of their devices. If a service provider has n devices and rating point of each device is denoted as r_i, then the rating point of that service provider will be $r_{sp} = \frac{\sum_{i=1}^{n} r_i}{n}$.

Algorithm 1: Dynamic Gateway Provider Selection

```
Input: gateway requirement $G_{req}$, rating requirement $P_{avg}$,
       cost requirement $costc$
Output: $List < selectedprovider >$
 1: $resources$ = getResources()
 2: var $compatibleResourceProviders$ = new
    $List < ResourceProvider > ()$
 3: $interestedResourceProviders = advertise(resources)$
 4: for $T$ in $interestedResourceProviders$ do
 5:     if $T.proposedCost <= costc$ and $T.rating >= P_{avg}$
        then
 6:         $compatibleResourceProviders$.push($T$)
 7:     end if
 8: end for
 9: var $targetedResourceProviderList$ = new
    $List < ResourceProvider > ()$
10: var $totalGateways$ = 0
11: while $compatibleResourceProviders$ not EMPTY do
12:     /*One or multiple available provider should be returned*/
13:     find $l_m$ from the $compatibleResourceProviders$ list such
        that $l_k.cost$ =
        $min(compatibleResourceProviders.getElement.cost)$
14:     if $totalGateways$ + $l_m.resources >= G_{req}$ then
15:         $targetedResourceProviderList.Push(l_m)$
16:         return $targetedResourceProviderList$
17:     else
18:         $targetedResourceProviderList.Push(l_m)$
19:         $totalGateways$ += $l_m.resources$
20:         $compatibleResourceProviders.Remove(l_m)$
21:     end if
22: end while
23: /*No provider is available to provide the service*/
24: return noCompatibleProviderFoundError
```

D. Payment calculation:

Let us consider that a provider provides d number of devices for providing a service. If each device uses r_j amount of resources per unite time and total service time is t, then the total resource used by all the devices from that service provider is $M = \sum_{j=1}^{d} r_j * t$, such that r_j = CPU speed, RAM, storage, bandwidth. If the service provider bids c as the cost for per unit resource usage, then the total payment P for the service is: $P = M * c$.

V. Experiment and Evaluation

A. Experimental setup

We implemented a prototype of IGaaS for the Contiki operating system [16]. The graph shown in Figure 2 was used for simulation on Cooja simulator [17]. We considered that the IoT network performs mesh-under routing, and hence the intermediate nodes hold no routing information. For P2P communication, we calculated the time and energy consumption for different size of packets to reach from node 1 to node 2. Similarly, for P2MP communication, we considered that the packet in sent from cloud to node 1.

B. Evaluation

We conducted our experiments for different message sizes varying from 64 bytes to 1024 bytes. For both P2P and P2MP

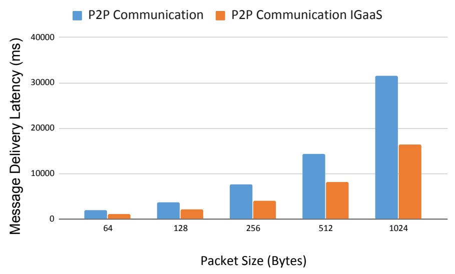

Fig. 6: Comparison of delay in P2P communication

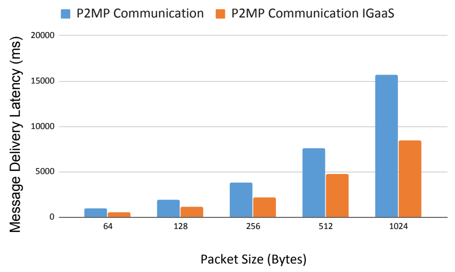

Fig. 7: Comparison of delay in P2MP communication

communication, we calculated end to end communication delay and energy consumption. Figure 6 and figure 7 shows the communication time comparison for P2P and P2MP communication respectively. We observe that the IGaaS framework reduces the communication delay. The communication delay was reduced by 43.2% for 32 bytes in P2P communication. For packet size 1024 bytes, we have found the reduction was 48%. For P2MP communication, we have seen a similar amount of improvement regarding communication latency with an average improvement of 41.82%. We have also performed a comparison of energy consumption. Figure 8 and figure 9 present the comparison of energy consumption for P2P and P2MP communications. We observed an average of 50.45% and 48.78% less energy consumption in P2P and P2MP communication, respectively. It is also possible to show similar kinds of improvements in MP2P communication. We can conclude, IGaaS framework can reduce the communication time and energy consumption.

VI. Conclusion

IoT devices are becoming popular in many different critical applications. The presence of numerous IoT devices in the RPL network with a fixed number of gateways may reduce the quality of service. Moreover, a relatively low number of IoT devices with an excessive number of gateways may lead to unnecessary power consumption. In this paper, we presented IGaaS - a service that achieves optimization in quality of service by on-demand gateway provisioning through drones or smartphones as well as reducing the number of gateways for both stationary and mobile scenarios. Our proof of concept implementation on the Contiki platform and simulation on Cooja shows the feasibility of the framework.

Acknowledgement

This research was supported by the National Science Foundation through awards DGE-1723768, ACI-1642078, and CNS-1351038, and by the National Institute of Health grant 1R21HD095270-01.

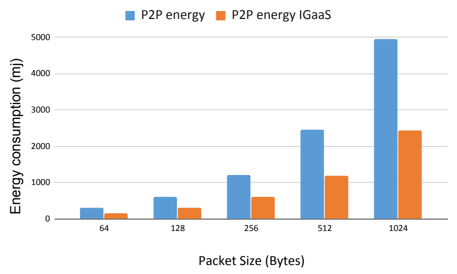

Fig. 8: Energy consumption comparison in P2P communication

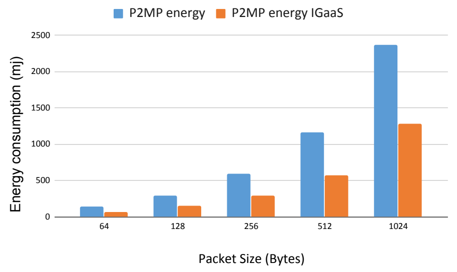

Fig. 9: Energy consumption comparison in P2MP communication

References

[1] H. R. Schindler, J. Cave, N. Robinson, V. Horvath, P. Hackett, S. Gunashekar, M. Botterman, S. Forge, and H. Graux, "Examining europe's policy options to foster development of the'internet of things'," 2012.

[2] M. S. Hossain, M. A. Rahman, and G. Muhammad, "Towards energy-aware cloud-oriented cyber-physical therapy system," *Future Generation Computer Systems*, 2017.

[3] M. Alhussein, "Monitoring parkinson's disease in smart cities," *IEEE Access*, vol. 5, pp. 19 835–19 841, 2017.

[4] S. Bitam and A. Mellouk, *Bio-inspired Routing Protocols for Vehicular Ad-hoc Networks*. Wiley Online Library, 2014.

[5] Zigbee, "An ieee 802.15.4-based high-level communication protocols for personal area networks," 2016, last accessed: 25-February-2020. [Online]. Available: https://zigbee.org/

[6] N. Kushalnagar, G. Montenegro, C. Schumacher *et al.*, "Ipv6 over low-power wireless personal area networks (6lowpans): overview, assumptions, problem statement, and goals," 2007.

[7] M. B. Yassein, W. Mardini, and A. Khalil, "Smart homes automation using z-wave protocol," in *2016 International Conference on Engineering & MIS (ICEMIS)*. IEEE, 2016, pp. 1–6.

[8] Z.-M. Lin, C.-H. Chang, N.-K. Chou, and Y.-H. Lin, "Bluetooth low energy (ble) based blood pressure monitoring system," in *2014 Int. Conf. on Intelligent Green Building and Smart Grid*. IEEE, 2014, pp. 1–4.

[9] O. Gaddour and A. Koubâa, "Rpl in a nutshell: A survey," *Computer Networks*, vol. 56, no. 14, pp. 3163–3178, 2012.

[10] J. T. Adams, "An introduction to ieee std 802.15. 4," in *2006 IEEE Aerospace Conference*. IEEE, 2006, pp. 8–pp.

[11] T. Winter, "Rpl: Ipv6 routing protocol for low-power and lossy networks," 2012.

[12] A. H. Chowdhury, M. Ikram, and H.-S. Cha, "Route-over vs mesh-under routing in 6lowpan," in *Proc. of the 2009 international conf. on wireless communications and mobile computing*. ACM, 2009, pp. 1208–1212.

[13] P. Karhula, J. Mäkelä, H. Rivas, and M. Valta, "Internet of things connectivity with gateway functionality virtualization," in *2017 Global Internet of Things Summit (GIoTS)*. IEEE, 2017, pp. 1–6.

[14] J. S. de Puga, C. E. P. Salvador, and A. B. Pellicer, "Architecture and use case for an iot deployment with sdn at the edge and dual physical and virtual gateway," in *ICCCN*. IEEE, 2019, pp. 1–6.

[15] R. Morabito, R. Petrolo, V. Loscri, and N. Mitton, "Legiot: A lightweight edge gateway for the internet of things," *Future Generation Computer Systems*, vol. 81, pp. 1–15, 2018.

[16] Contiki, "An open source operating system for the internet of things," 2019, last accessed: 25-February-2020. [Online]. Available: http://www.contiki-os.org/

[17] Cooja, "An introduction to cooja," 2019, last accessed: 25-February-2020. [Online]. Available: https://github.com/contiki-os/contiki/wiki/An-Introduction-to-Cooja