International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 11, November-2020

Edge IoT Platform Supporting Multi-Tenancy for Smart Distributed Systems

Mrs. K. Mohanaleela
Department of ECE
Sridevi Women's Engineering College

Mrs. G. Bavitha
Department of ECE
Sridevi Women's Engineering College

***Abstract*-The rapid growth in technology and wide use of the internet has increased smart applications such as intelligent transportation control systems, and the Internet of Things, which heavily rely on an efficient and reliable connectivity network. To overcome high bandwidth workload on the network, as well as minimize latency for real-time applications, the computation can be moved from the central cloud to a distributed edge cloud, that shifts the function of centralized cloud computing to edge devices of networks. By abstracting the IoT edge components such as data streams or by virtualizing the devices/actuators, it becomes possible to avoid or resolve access conflicts. The edge computing benefits various smart applications that use a distributed network for data analytics and amenities. Different from the existing cloud management solutions, edge computing needs to move cloud management services towards distributed assorted edge nodes for multi-tenant user applications. However, existing cloud management services do not offer remote sharing of multi-tenant user applications on the cloud of edge nodes. IoT systems that support multi-tenancy tend to use cloud-hosted device/thing virtualization. Through this paper, we present the idea of light-weight virtual resources that can be hosted on edge devices and thus offer the same abstraction/virtualization without latency.**

Keywords: IoT, Edge-computing, Multi-tenancy, virtual resources.

I. INTRODUCTION

Cloud computing platforms, like Amazon Web Services, Microsoft Azure, and Google Cloud Platform, have grown popular for providing universal access to services across a variety of user devices. Cloud platform providers cash in of economies of scale by managing and operating resources in an exceedingly centralized manner. Cloud computing provides reliable, scalable, and versatile resources, which is motivating more and more developers to use their services and applications within the cloud. Netflix and Dropbox are samples of popular cloud-based services. Users can obtain high-performance computing and enormous storage resources everywhere with Internet access at any time. Cloud-based services centrally managed resources and applications to assist users by reducing requirements to put in and configure software locally on mobile devices with comparatively low-performance computing resources and tiny cupboard space. Google Docs and Adobe Creative Cloud are samples of cloud services that conveniently provide regular office tools. Cloud services require developers to host services, applications, and data on off-site data centers, i.e. the computing and storage resources are spatially distant from users' devices. However, with the rise within the number of high-quality services wants computational tasks to be located nearby. These services require lower latency, greater responsiveness, improved user experience, and more efficient use of network bandwidth. As a result, over the last decade, several research efforts have adopted the requirement and benefits of making edge-computing services that distribute computational functions closer to the client devices. We present an edge-computing platform, Para Drop implemented on Wi-Fi access points (APs) or other wireless gateways like set-top boxes. Para Drop allows third-party service developers to bring computational functions into homes and corporations. The Wi-Fi APs are ubiquitous, always "on" and available, and sits directly within the data path between the net and user devices. For these reasons, we've got selected the Wi-Fi APs for Para Drop because of the edge-computing node.

II. EDGE COMPUTING

Edge computing is a technology that allows the computation taking place at the vicinity of data being produced on distributed micro "data centers". If the IoT-generated data is processed, stored, analyzed, and operated close to, or at the edge of the network, the problem of high network bandwidth requirements for future IoT applications will be solved. Besides, the edge computing node acts as both consumer and producer, which decreases the data processing pipeline and increases the event response time. For future IoT applications, e.g., Smart City monitoring, traffic sensing, security enforcement, fast response is the key to ensure smart city service qualities. However, to enable edge computing for smart city applications, multiple challenges need to be solved, for example, remote device programmability, naming, data abstraction, service management, privacy, security, and other optimization metrics.

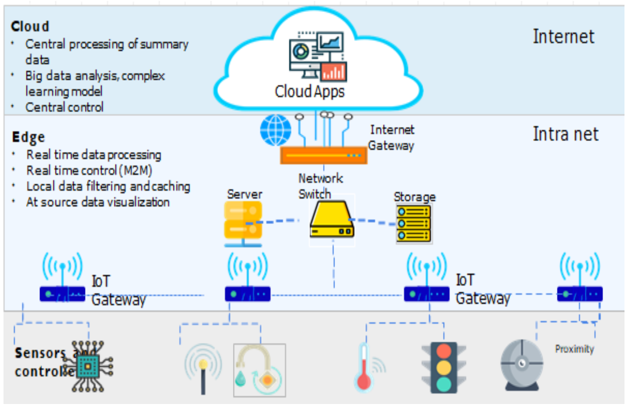

Figure 1: Edge computing

We have identified the following five needs that motivate computing excitable nodes.

1)Decentralized Cloud and Low Latency Computing

Centralized cloud computing won't always be the foremost effective strategy for applications that are geographically distributed. Computing must be performed closer to the source of the data to spice up the service that's delivered. This benefit is generalized for any web-based application [1]. Location-aware applications like Foursquare and Google Now became popular among mobile users. Similarly, multimedia applications, like on-demand gaming on modern cloud infrastructure, pose similar latency issues for the gamer [2]. Here, to boost the computations performed on the cloud, edge nodes that are located closer (for example, routers or base stations one-hop away from a grip device) to users are often leveraged to chop back network latency.

2)Surmounting Resource Limitations of Front-end Devices

User devices, like smartphones, have relatively restricted hardware resources compared to a server during an information center. These front-end devices capture sensory input within the sort of text, audio, video, touch, or motion and are processed by a service provided by the cloud [3]. The front-end devices cannot do complex analytics due to middleware and hardware limitations [4]; this might sometimes be possible at the risk of draining the battery. Hence, often data must be sent to the cloud to satisfy the computational demands of processing data and significant information is then delivered back to the front-end. However, not all data from a front-end device will be utilized by the service to construct analytical workloads on the cloud. Potentially, data could also be filtered or even analyzed at edge nodes, which may have spare computational resources to accommodate data management tasks.

3) Sustainable Energy Consumption

There is a substantial body of research that has investigated the energy consumption of the cloud data center's [5], [6]. Data centers within the following decade are likely to consume 3 times the most amount of energy consumed today and there is a greater need for adopting energy-efficient strategies that will minimize energy usage.

4)Dealing with Data Explosion and Network Traffic

The number of edge devices is growing at an infinite rate. Consequently, the number of knowledges which will be generated also will increase; it's anticipated that 43 trillion gigabytes of information are generated in 2022. This places the need for expanding data centers to support monitoring and analytical workloads, but this again raises concerns about sustainable energy consumption of data centers.

5) Smart Computation Techniques

Edge nodes can facilitate computations nearer to the source of information (or where data is generated) and will incorporate strategies for remotely enhancing capabilities of front-end devices.

III. CHALLENGES

For performing such a framework, we imagine that the following five research challenges at the hardware, middleware, and software layer have to be addressed.

Challenge 1 - General Purpose Computing on Edge Nodes

The first challenge within the software space is to develop solutions that are portable across different environments. there's research in upgrading the resources of edge nodes to support general-purpose computing. for instance, a wireless home router is upgraded to support additional workloads [7]. Intel's Smart Cell Platform uses virtualization for supporting additional workloads. Replacing specialized DSPs with similar general-purpose CPUs gives an alternate solution but this needs a large investment.

Challenge 2 - Discovering Edge Nodes

Discovering resources and services in a very distributed computing environment is a region that's well explored. Reliably and proactively addressing faults on the node and autonomically recovering from them are going to be desirable. Existing methods utilized in the cloud won't be practical during this context for the invention of edge nodes.

Challenge 3 - Partitioning and Offloading Tasks

Evolving distributed computing environments have resulted in the development of various techniques to facilitate the partitioning of tasks that may be executed at multiple geographic locations [8], [9]. for instance, workflows are partitioned for execution in numerous locations [10], [11]. Task partitioning is sometimes expressed explicitly during a language or management tool. However, making use of edge nodes for offloading computations models the difficulty of not simply partitioning computational tasks efficiently, but automatically doing this without explicitly defining the capabilities or location of edge nodes.

Challenge 4 – Determined Quality-of-Service (QoS) and Experience (QoE).

The challenge here is to ensure that the nodes achieve high throughput and are reliable when delivering for their intended workloads if they accommodate additional workloads from a datacenter or from edge devices.

Challenge 5 - Using Edge Nodes Publicly and Securely

Hardware resources that are owned by data centers, supercomputing centers, and personal organizations using virtualization will be transformed to supply computing as a utility. The associated risks for a provider and users are articulated, thereby offering computing on a pay-as-you-go basis. This has resulted in a very competitive marketplace with numerous options and choices to satisfy computing consumers by meeting Service Level Agreements (SLAs) [12]. Secondly, the expected purpose of the device, as an example, a router managing internet traffic, can't be compromised when used as a grip computing node. Thirdly, multi-tenancy on edge nodes will only be possible with technology that places security as a foremost concern.

IV. PARA DROP

Para Drop uses a light-weight virtualization framework through which third-party developers can create, deploy, and revoke their services in several APIs. These services are housed in isolated containers (called “chutes,” short for “parachutes”—as in parachuting a service on demand) that allow them to retain user state and move with users as they modify their points of attachment. Because Wi-Fi APs are likely to be resource-limited for several differing kinds of applications, like video analysis and data caching, Para Drop allows for tight resource control through a managed policy design. Unlike cloud infrastructure where providers own all devices, Para Drop assists users to contribute their resources to the platform. This different ownership model results in more complex permission management and resource policies than in cloud computing. The Para Drop platform has three key components: a virtualization substrate within the Wi-Fi APs that hosts third-party computations in chutes; a cloud-side (controller) through which all of the third-party services are dynamically installed, instantiated, and revoked; and a developer API through which developers can manage the resources of the platform and monitor the running status of Apps and chutes.

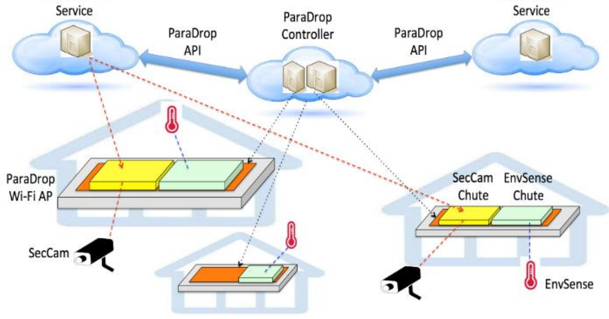

Fig. 2. Overview of Para Drop. There are three components in the system: Para Drop backend, Para Drop gateways, and the developer API. Developers can deploy services in containers (we call them chutes in Para Drop platform, as in parachuting a service on-demand) to the gateways through the backend server. The chutes are created using the developer API. Two different chutes are illustrated in this figure: 1) Sec Cam: a wireless security camera service that collects video from an in-range security camera and does some local processing to detect motion; and 2) Env Sense: a system deployed in buildings to monitor temperature and humidity with sensors.

In addition to low latency and decreased network traffic, Para Drop also has unique privacy advantages. For example, a third-party developer can create a Para Drop service that allows sensitive user data such as a video feed to reside locally in the Para Drop AP, rather than sending a continuous feed over the open Internet.

DESIGNING AN EDGE COMPUTING PLATFORM

With Para Drop, we introduce a complete framework for management and operation that connects edge nodes and exposes their resources. The framework forms a platform including a pool of devices distributed in close proximity to mobile devices and users. The platform provides an API for both developers and users to manage services and resources securely and transparently.

V. DESIGNING A PARA DROP

Here, we address the difficulties of implementing an edge-computing platform and how we overcome them in Para Drop.

Highly effective virtualization in edge nodes

The current generation of Para Drop AP relies on the single-board computer PC Engines APU1. we decide Ubuntu Core—an embedded OS designed for IoT devices—as the OS for the sting nodes. Ubuntu Core features a smaller footprint than Ubuntu’s desktop and server versions, and it includes a package-management system that supports transactional updates. Unlike servers in data centers, it's hard to repair the hardware and software on an AP after extending Para Drop APs at the network edge (a user’s home or office). Ubuntu Core provides secure and transactional update capability and supports image backup and rollback features in order that the software running on APs are often managed easily and flexibly. Ubuntu Core includes a similar run-time environment as other Ubuntu versions—the similarity of the sting nodes to cloud servers in both hardware and software simplifies the task of deploying services from the cloud to the sting.
Wi-Fi APs have relatively limited resources compared to servers in data centers for cloud computing. Therefore, efficiency was a vital consideration in choosing the virtualization technique for Para Drop. Para Drop implements virtualization with Docker,[13] an application layer virtualization solution supported Linux containers.[14] This technique allows Para Drop to effectively use the limited resources available at the network edge. It provides higher performance virtualization

than hypervisor-based virtual machines (VMs), especially in I/O speeds,[15] which is very important for latency-sensitive applications. Moreover, a container is more lightweight and faster than a VM because it doesn't run a full OS on the virtual hardware. we would have liked to be very careful about resource policies because the resources on the sting nodes are shared between services and with the resource owners (the owner of the Wi-Fi AP). the primary objective is to ensure the services running on the sting node will never impact the users' tasks—in other words, the policy enforcement module within the AP must reserve resources for the fundamental function of the AP (Wi-Fi network access). Also, owners (users or ISPs) of the Wi-Fi APs grant permissions to use some resources on the APS to Para Drop, which successively grants the permissions and quotas to the services deployed within the APS. Different services running on one AP are isolated from one another.

Scalable orchestration framework

Generally speaking, the Para Drop platform has to be capable of managing a bigger number of nodes than a cloud-computing platform. to form matters worse, the sting nodes are distributed, and in many cases, the controller doesn't have direct access to the sting nodes thanks to firewalls. Therefore, it's difficult for developers to manage and debug services running on APs. Creating a virtual private network (VPN) is one approach to beat this challenge; however, with a VPN, a centralized controller has to maintain connections to all or any APs. Developers also have to configure the VPN on their machines to access the API. To analyze the controller implementation and developers' work, we used web protocols— HTTP and WebSocket—to enable bidirectional communication channels between the Para Drop controller and also the APS. With WebSocket, we will transmit time-sensitive messages among the controller, developer tools, and APs with low latency. because the number of managed APs and developers increases, the controller is often scaled up by using readily available tools and techniques for web applications like replication and cargo balancing other edge-computing platforms build the orchestration framework with controllers the same as those used for cloud computing. as an example, Cloudlet builds the controller by extending OpenStack. We believe building a specialized, scalable controller for Para- Drop can satisfy the unique requirements of edge computing.

VI. PARA DROP API

Para Drop implements an API for developers to make and deploy edge services. The API are visiting be divided into two parts: the Para Drop core interfaces implemented by the Para Drop controller within the cloud to imply the capabilities of the management framework, and also the sting node interfaces implemented by the Para Drop daemon running on the sting nodes (Para- Drop APs) to reveal the sting nodes' capabilities. The Para Drop core interfaces the resource provision, monitoring, and edge service (chute) management. With these interfaces, users can register new user accounts, add new APs to Para Drop and assign resources to use by Para Drop, monitor APs' running status or change APs' configurations, create/edit chutes, and manage (configure, install, remove, start, or stop) chutes on APs. Based on these interfaces, we have developed an internet frontend for users and system administrators. Developers must register their microservices with the Para Drop controller as chutes before integrating Para Drop with their applications. We implemented a permission management framework so users can control which and also the way many resources are visiting use by third-party applications through the chutes. The edge node interface provides local context information for the sting services and exposes hardware-specific functionality. samples of edge node functions include listing devices connected to the sting service's Wi-Fi network and their signal strength, listing Bluetooth devices within the vicinity, and playing or recording audio.

System implementation

Para Drop is an ongoing research. we've got fully implemented the 2 major system modules: the Para Drop daemon on the AP and therefore the Para Drop controller within the cloud. These modules work together to supply the Para Drop API. The Para Drop daemon is implemented with the Python-based Twisted framework, and also the Para Drop controller is implemented with Node.js.

VII. DEVELOPING PARA DROP APPLICATIONS

Developers can develop two categories of applications for Para Drop: standalone and cloud/edge hybrid. Both categories can use Para Drop's edge node interfaces within the chutes. Para Drop's target applications are those requiring low latency, high bandwidth, or high privacy. thanks to the relatively low-performance processor of the present Para Drop APs, we might not recommend that developers deploy computation-intensive tasks on the APs for now. However, Para Drop(16) supports high-performance hardware, and that we are acting on the support for specialized processors (GPUs) on Para- Drop APs.

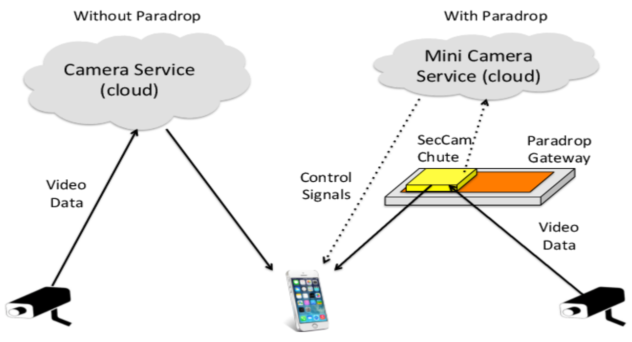

Figure:3 Comparison of a web protocol camera service without Para Drop and with Para Drop. Without Para Drop, the camera has to continuously push video data to the cloud for processing and storage. However, we are able to deploy part of the cloud service on the AP with Para Drop. If the mobile device encompasses a direct reference to the service on the AP, it'll control the camera and pull video data from it directly without counting on the service deployed within the cloud. If not, the service on the AP can still do most of the work (for example, motion detection), and it can create a peer-to-peer connection to the mobile device with the assistance of the simplified camera service within the cloud

Standalone Para Drop applications are almost like smartphone applications. Through the local dashboard within the Para- Drop AP, users can browse the accessible chutes and choose which to put in or remove. From this perspective, we are able to think about a Para Drop AP as a sensible router with a controller within the cloud. The cloud/edge hybrid Para Drop applications are those applications having back-end servers combined with the Para Drop core interfaces. These applications must be registered at www.Para Drop.org to urge application IDs and secret keys, and to induce permissions from resource owners before the applications use Para Drop resources. For those applications, the chutes running within the Para Drop APs are microservices of the applications. The back-end server deployed within the cloud calls the Para Drop core interfaces to dynamically deploy microservices (chutes) into the acute edge (Para Drop APs). Through lightweight virtualization, Para Drop provides maximum flexibility to developers. Developers can choose the programming languages and frameworks they like and target the applying logic development. they will also easily port code originally written for the cloud to the Para- Drop platform with minimal effort. A chute has to provide a Docker file to define the Linux container to execute the appliance logic; a configuration file to specify the environment requirements of the container, like Wi-Fi and Bluetooth interfaces; and a few application-specific files to implement application features.

VIII. IOT APPLICATIONS

The IoT is becoming a large part of the networking world, but many IoT devices depend on back-end services within the cloud. Using Para Drop, we are able to push that intelligence from the cloud to the sting (the APS).

SecCam

As mentioned earlier, we implemented a security camera service employing a wireless video camera with a Para Drop AP. For this service, we require web interfaces to speak with the camera and the Internet additionally should provide enough storage for images. Figure 3 compares the camera service deployed within the cloud and also the Para Drop platform. The SecCam chute is implemented with Node.js and Python. it's responsible for:

1)Creating a wireless network, which provides an isolated Wi-Fi network and subnet to the protection cameras.

2)Capturing and processing images. A program accepts user-defined characteristics like a threshold of motion, time of day, and rate of detection. Then it captures images from the camera, calculates differences to detect motion, and stores those images to disk.

3) Providing a neighborhood server and a cloud service agent. The local server allows users to look at live video, check logs, and look at motion-detection images stored on the Para Drop AP. The cloud service agent communicates with the cloud service to implement device and image archiving.

EnvSense

This service could be a wireless environmental sensor designed as a component of the Emonix research platform.[22] Because the service was fully implemented for a cloud-computing platform, we only must migrate a part of the service into the Para Drop platform. the first service runs within the cloud to gather data from the sensors, then processes, stores, and visualizes the info. We divide the EnvSense cloud service into many microservices and extend some microservices on the Para Drop platform. The information collection and processing microservices are used on Para Drop, while the info storage and visualization microservices still run within the cloud. because of the similarity of the Para Drop edge nodes' runtime to the servers' runtime within the cloud, we were ready to port the code written for cloud servers to Para-Drop chutes with minor effort. Para Drop may be a flexible edge- computing platform that users and developers can use to deploy diverse services at the acute fringe of the network.

IX. CONCLUSION

Para Drop is accessible as an open-source project from www.Para Drop.org, which also includes documents and tutorials that enable developers to start working with Para-Drop. Para Drop can be used to develop services like SecCam, which provides preliminary evidence of its easy use. We are proposing a platform to support extra peripheral devices, like Bluetooth and audio, and that we are polishing the Para Drop API and improving development tools to support flexible application development and deployment. Besides, we are investigating edge-computing applications for other uses based on para drop.

REFERENCES

[1] J. Zhu, D. S. Chan, M. S. Prabhu, P. Natarajan, H. Hu, and F. Bonomi, “Improving Web Sites Performance Using Edge Servers in Fog Computing Architecture,” in Proceedings of the International Symposium on Service-Oriented System Engineering, 2013, pp. 320–323.

[2] S. Choy, B. Wong, G. Simon, and C. Rosenberg, “The Brewing Storm in Cloud Gaming: A Measurement Study on Cloud to End-user Latency,” in Proceedings of the Workshop on Network and Systems Support for Games, 2012.

[3] D. Dimopoulos, G. Kambourakis, and G. Portokalidis, “The Best of Both Worlds: A Framework for the Synergistic Operation of Host and Cloud Anomaly-based IDS for Smartphones,” in Proceedings of the European Workshop on System Security, 2014, pp. 6:1–6:6.

[4] A. Houmansadr, S. A. Zonouz, and R. Berthier, “A Cloud-based Intrusion Detection and Response System for Mobile Phones,” in Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, 2011, pp. 31–32.

[5] A. Beloglazov, J. Abawajy, and R. Buyya, “Energy-aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing,” Future Generation Computer Systems, vol. 28, no. 5, pp. 755–768, May 2012.

[6] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The Cost of a Cloud: Research Problems in Data Center Networks," SIGCOMM Computer Communication Review, vol. 39, no. 1, pp. 68–73, Dec. 2008.
[7] C. Meurisch, A. Seeliger, B. Schmidt, I. Schweizer, F. Kaup, and M. Mu¨hlha¨user, "Upgrading Wireless Home Routers for Enabling Large- scale Deployment of Cloudlets," in Mobile Computing, Applications, and Services, 2015, pp. 12–29.
[8] D. Yin and T. Kosar, "A Data-aware Workflow Scheduling Algorithm for Heterogeneous Distributed Systems," in International Conference on High-Performance Computing and Simulation, 2011, pp. 114–120.
[9] T. Ghafariana and B. Javadi, "Cloud-aware Data-Intensive Workflow Scheduling on Volunteer Computing Systems," Future Generation Computer Systems, vol. 51, pp. 87–97, 2015.
[10] W. Chen and E. Deelman, "Integration of Workflow Partitioning and Resource Provisioning," in Proceedings of the IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2012, pp. 764–768.
[11] W. Tang, J. Jenkins, F. Meyer, R. Ross, R. Kettimuthu, L. Winkler,X. Yang, T. Lehman, and N. Desai, "Data-Aware Resource Scheduling for Multicloud Workflows: A Fine-Grained Simulation Approach," in Proceedings of the IEEE International Conference on Cloud Computing Technology and Science, 2014, pp. 887–892.
[12] S. A. Baset, "Cloud SLAs: Present and Future," ACM SIGOPS Operating Systems Review, vol. 46, no. 2, pp. 57–66, 2012.
[13] D. Merkel, "Docker: Lightweight Linux Containers for Consistent Development and Deployment," *Linux J.*, vol. 2014, no. 239, 2014, arti-cle no. 2.
[14] S. Soltesz et al., "Container-based Operating System Virtualization: A Scalable, High-performance Alter- native to Hypervisors," *Proc. 2nd ACM SIGOPS/EuroSys European Conf. Computer Systems* (EuroSys 07), 2007, pp. 275–287.
[15] W. Felter et al., "An Updated Per- formance Comparison of Virtual Machines and Linux Containers," *IEEE Int'l Symp. Performance Analysis of Systems and Software* (ISPASS 15), 2015, pp. 171–172.
[16] P. Liu, D. Willis, and S. Banerjee, "Para- Drop: Enabling Lightweight Multi- tenancy at the Network's Extreme Edge," *IEEE/ACM Symp. Edge Comput- ing* (SEC 16), 2016; ieeexplore.ieee.org /document/7774349.

ABOUT AUTHORS

Mrs. Mohana leela kalahasthu is working in Sridevi Women's Engineering college Hyderabad as an Assistant Professor in ECE department. She is having 5 years of experience in teaching. Her areas of interest include VLSI, Embedded systems, Communications, IOT, and Big Data analytics

Mrs.G. Bavitha is working in Sridevi Women's Engineering college Hyderabad as an assistant professor in ECE department. She is having 6 years of experience in teaching. Her areas of interest include VLSI, Embedded systems, Communications, IOT, and Big Data analytics