# Fog Computing - A Brief Analysis in the Implementation of Internet of Things

1st Vaibhav Sundharam
*The Bradley Department of ECE*
*Virginia Tech.*
Blacksburg, USA
vaibhavsundharam@vt.edu

2nd Shagun Johari
*The Bradley Department of ECE*
*Virginia Tech.*
Blacksburg, USA
shagunjohari@vt.edu

***Abstract*—The advancement of the Internet of Things has revolutionized the world we live in today. With this expansion, there comes a need for methods that can handle the vast amount of data being generated by the IoT. Fog computing is one such paradigm. It mitigates high-latency and provides better bandwidth utilization, thereby increasing agility and productivity of a network. This paper starts with a brief review of what fog computing is, its components and architecture. Next, to demonstrate the superiority of fog-computing as compared to cloud-based infrastructures, we developed an accident detection application to compare the difference in the response time of fog and cloud. We monitored the response time as well as data usage and have presented plots for the same.**

**Our project team has two members, namely, Ms. Shagun Johari and Mr. Vaibhav Sundharam. Mr. Vaibhav was responsible for the coding of the fog server while Shagun's responsibility was to achieve the same on the cloud server. The analysis of the fog and cloud servers was done in collaboration.**

***Index Terms*—IoT, fog computing, cloud computing, latency, network utilization**

## I. Introduction

Today's day and age sees society becoming more and more dependent on smart devices and computers for their daily tasks. These devices consist of sensors and applications which continuously generate data. As a result, a lot of data is being generated and stored on a daily basis. Such devices that generate and share data through the internet are called the Internet of Things (IoT). The IoT is responsible for making the world smarter and more receptive. With the sudden increase in the number of IoT devices that are being used today, the volume of data being produced has also increased exponentially. Managing and structuring such huge volumes of data has become a challenge [5]. As traditional methods to process data cannot be used anymore, there has become a need to look at other ways such as big data analytics. Many organizations are analyzing all the data that they get to make better decisions for their customers [5]. Cloud computing has been pivotal in the reforming of the IT industry. All the data from the IoT devices, which require storage or analysis, are sent to data centers in the cloud [12].

On the other hand, as the IoT devices keep proliferating, the amount of data coming from the billions of sensors, cannot keep being transferred to the cloud. It also may not be efficient to move all data from each device to the cloud as some computation needs to be done faster than possible in the cloud. For example, in certain cases such as autonomous vehicles or patient monitoring, data needs to be processed faster as these applications are time-sensitive and the cloud will not be able to overcome the disadvantage known as latency. There are also other reasons why sending data to the cloud could be problematic, such as bandwidth issues or privacy concerns for certain applications. Fog computing is one solution that has been presented for the above problems by academia and the industry alike [7]. Cloud computing relies on centralized components while fog computing is a decentralized computing method. Fog computing is basically a concept similar to cloud computing except that it enables various services offered by the cloud, such as storage, processing, networking, on nodes close to the IoT devices, i.e., in the proximity of the IoT device [7] [18]. However, this does not mean that fog computing has replaced cloud computing. It is still dependent on the cloud for complex computation. Many other concepts, such as edge computing, mist computing, cloudlets, etc., are also being researched and are good solutions for the problems faced by cloud computing.

In this paper, we have explained fog computing and its components and compared it with cloud computing and other related paradigms. We have also created a fog architecture consisting of IoT devices, fog node and a cloud server (hosted on Amazon Web Services) to compare actual response time and network usage of fog and cloud computing.

The paper is formulated as follows: Section II gives a brief introduction to fog computing followed by section III in which the difference between fog and cloud computing is explained. Section IV discusses other related computing paradigms and is succeeded by section V which examines the components and the architecture of a fog-based computing system. In section VI, we elucidate on the relation of IoT with big data followed by section VII in which the infrastructure of our fog network is described. We then discuss the results in section VIII which is followed by the conclusion in section IX.

## II. A Review of Fog Computing

The term 'Fog Computing' was given by Cisco Systems in 2012. Fog computing is a distributed network that administers

computation, networking services, and storage between IoT devices and cloud servers usually, but not always, located at the edge of the network [3]. Fig. 2 shows a simple architecture of how fog computing relates to IoT applications [7]. However, it is important to note that computation at the edge of the network is an approach that has been talked about before. This concept came forth in the 2000s [17], [8] and another concept, named 'cloudlets', was also popularized in around 2009 [10]. Cloudlets and fog computing are both related to computation at the network edge but cloudlets are used in mobile processing while the fog is used for IoT connected networks. An important thing to note is that the services that fog provides, may not exactly be at the network edge. It is known that providing computation near devices would enhance the performance of the services, but only if the computation amount is not that high. There are many actuators and sensors that will be connected to a fog node. Whenever the processing is necessary, it is performed at the fog device and there is also access to storage for a short amount of time. This type of environment for computation of data becomes important when there is a need for time-sensitive computation. This can also be elucidated by looking at an example. In autonomous cars, there's an on-board system that comprises of many sensors for various needs such as sensors that learn distance from other objects, sensors which help to read road signs, sensors which record speed, acceleration, GPS, etc. These sensors are continuously taking in real-time data and need continuous processing. If all this data is sent to the cloud server for processing, it would be very problematic. This is due to the fact that when it comes to driving, very quick decisions need to be made as it is a matter of life and death. Hence, we need fog servers so that the data can go to these nodes and be processed as that would reduce latency and help in improving the performance of the car.

Quality of Service (QoS) is another guarantee we need whenever sending data through a network and, unfortunately, sending large amounts of data can be problematic. This can be understood when looking at how fog computing is essential in healthcare. Many health monitoring systems also consist of IoT devices which are regularly collecting data. If all this data is sent to the cloud, then there is a high chance of increased latency in data transmission, bit errors, and packet dropping rate. This is because, as the probability of errors increases packet re-transmissions also increase [9]. In monitoring the health of a patient, one small error in the data could cause wrong treatment decisions and affect the health of a patient tremendously. So, reducing the amount of data that is sent to the cloud is essential in maintaining QoS. The fog layer provides an extra layer between the cloud layer and the end device. The pre-processing that occurs at the fog layer helps in reducing the volume of data sent to the cloud, guaranteeing QoS, and also saves network bandwidth [16].

## III. Difference between Fog and Cloud Computing

The concept of cloud computing was first conceived in late 2007 and gained popularity due to its capabilities of offering a dynamic and flexible IT infrastructure [9]. Apart from that, it also offers a guaranteed QoS computing environment and considerable software services. Cloud computing has helped in expanding storage, processing, and networking framework. The main question is how does cloud computing actually work?

Cloud computing is basically when users move their data and applications to a remote server called a 'cloud' and then ingress their data in a straightforward and ubiquitous way [16]. Hence, this is a centralized process. One of the most important aspects of a cloud is its data centers. These are locations where data is stored, processed, and distributed by networking and computing systems. Virtualization is also a key component in clouds [15]. The cloud has many benefits when we look at the services it provides. Users access resources from the cloud servers and then run their applications on it. There are many services provided such as HaaS (Hardware as a Service), SaaS (Software as a Service), and DaaS (Data as a Service) [9]. In Haas, a user can buy a data center on a "pay-as-you-go" subscription basis. Examples are Amazon's EC2 service or IBM's Blue Cloud project [16]. SaaS, on the other hand, provides software or application to the user over the internet [9]. An example is Google Apps. Lastly, DaaS gives the user a way to access data on a network [9]. An example is Amazon's S3, which can be used to store and retrieve data from anywhere on the web IaaS [9]. HaaS, SaaS, and Daas together make a way for LaaS (Infrastructure as a Service) like the Google App Engine.

On comparing the features of cloud computing with fog, it's clear that while clouds use a centralized framework, fog computing uses a distributed framework. Also, fog computing architectures have multiple fog devices that help in processing the data, while in cloud architecture there are data centers. This causes high energy consumption and operational costs in cloud computing as compared to fog [11]. Comparing in terms of latency, fog is positioned much closer to the user so devices can reach it in a few hops while it takes many more hops to reach the cloud. That's why the latency of cloud is more and cannot be used for real-time interactions [11]. On observing data security, fog computing provides more security as it is a decentralized network system. In cloud computing, there is one centralized server and if that server becomes vulnerable in some way, it becomes hard to mitigate the threat. Meanwhile, in fog computing, processing occurs at different local endpoints which makes it easier to find threats. When it comes to the rate of failure though, fog has a higher rate due it being a decentralized system and because of wireless connectivity to IoT devices [11].

It is an important fact to state that fog computing is not a replacement for cloud computing. Fog nodes cannot be trusted for heavy computations, and thus in such cases, the data must be processed at the cloud.

## IV. Other Related Computing Paradigms

There are other computing technologies that exist which are similar to fog (Fig. 1). These include Mobile Cloud Computing

(MCC), Mobile-Edge Computing (MEC), and Edge Computing.

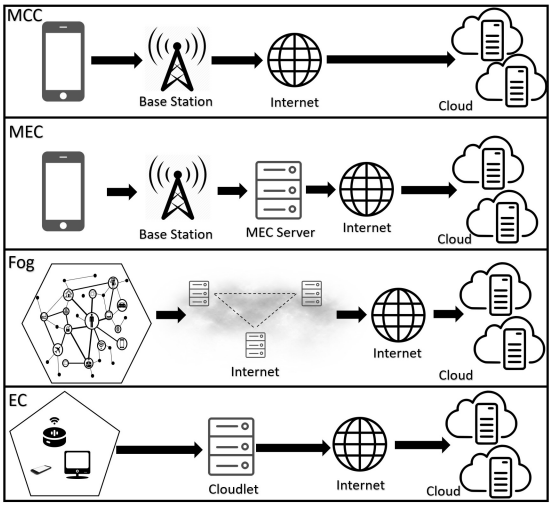


Fig. 1. Different computing technologies

### A. Mobile Cloud Computing (MCC)

Another computing method that has been developed from cloud computing is termed as Mobile Cloud Computing (MCC), a mixture of three diverse technologies which are cloud computing, mobile computing, and networking [12]. It focuses on cloud computing performed through portable (or mobile) devices. The infrastructure for data processing, storage, and networking is present outside the portable device [18][13]. The main aim of this type of system is to build up computing capabilities of portable devices (smartphones etc.) while conserving battery, and extending storage space [12]. This is done by having a small cloud server, called the cloudlet, placed at the edge of a network. In today's time, there are many applications running computer vision, natural language processing, and machine learning algorithms which will require edge processing along with cloud computing to meet their requirements of heavy computation [11]. For this, there are many features that an MCC system would need such as low latency in the middle tier, smooth remote execution, and optimization of the cloud infrastructure [11][13].

### B. Mobile Edge Computing (MEC)

The objective behind MEC is to reduce the stress on the network by deviating the computational work from the internet to the edge [6]. Base stations' task is to forward traffic, they do not administer computing capabilities apart from network connectivity. Hence, MEC devices, hosted at or near base stations, will be deployed for storage and processing duties. This will make these new devices, termed as the MEC servers, capable of processing requests by users, rather than sending them to an internet service where they could be processed instead [11][13]. Many services could be used through this paradigm like IoT, augmented reality, etc. Despite the advantages, MEC still struggles with scalability and the fact that MEC servers would have to be installed at or near base stations whose only job would be related to MEC services [11].

### C. Edge Computing (EC)

In all the computing paradigms explained above, 'edge' has been a term that keeps on coming. Edge devices are computing devices that come between the cloud and the data source [14]. Generally, edge computing does not substantially relate to any cloud-based service and is mostly used when talking with respect to IoT devices. An edge device may not necessarily have to be a server. It could be a cloudlet when talking in terms of a mobile application or a smartphone if a cloud application is working on a smartphone [11]. It should just not be more than one hop away from the data source [18]. Edge computing is central for IoT devices being used currently. Though fog and edge computing sound similar, there are differences between the two. In fact, edge computing and fog computing are sometimes used erroneously as the same. It is important to note that fog computing provides computation, storage, networking, and acceleration from the fog to the devices whereas edge computing only provides computation [18].

## V. Fog Architecture and Components

There is no standard architecture that is available for fog computing, but many research works have provided insight into their versions of fog computing architecture. A fog computing architecture usually has three distinct layers, as shown in Fig. 2. The most crucial layer is the fog layer. It consists of all the computing devices or 'fog nodes' on which processing, storage, and networking are done. Virtualization can also be employed in this layer, similar to cloud. All the data from the IoT devices is sent to the fog layer where it is processed and the result is returned to the device. Despite processing the generated data, the big data dilemma may still exist. There are billions of IoT devices all generating big data, hence it becomes a necessity to have small or medium level big data processing capabilities available at this layer [11].

The lowest layer in this architecture consists of the IoT devices, and hence we can call it the IoT plane [11]. These devices are the ones that have sensors and actuators on board, continuously generating data that needs to be processed. The final layer of this architecture is the cloud layer. If processing is too complex and needs more computation power or more storage, then it can be done on the cloud [11]. The fog layer will administer what is to be sent to the cloud for processing. The user will always get whatever service it requires, whether it is from the fog or the cloud server.

The basic components of fog computing architecture are explained below [11]:

- Physical layer: This includes all data sources such as sensors. The sensors could be for checking temperature, humidity, heart rate, surveillance, traffic monitoring, etc. There are also virtual sensors that exist. These are primarily software that work based on more than one

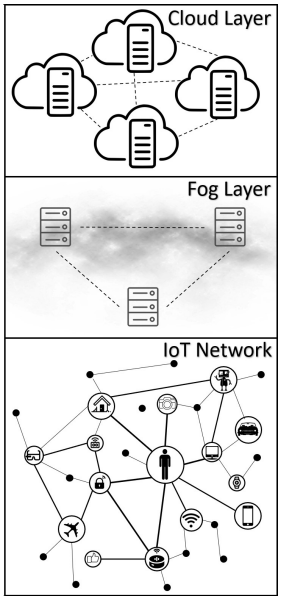


Fig. 2. Fog-computing architecture

sensor. Sometimes a single sensor is not enough to gain information, so virtual sensors are used.

- Fog nodes and fog servers: Fog servers are responsible for monitoring fog nodes. They are equipped with high computing power and high storage to manage different fog nodes connected to it. On the other hand, fog nodes are devices to which a group of IoT devices is connected. Fog nodes are also equipped with computing units to process data coming from IoT devices.

- Monitoring layer: It tracks the performance of the system and the resources, services, and responses required.

- Pre-and-Post-processing layer: It's main objective is to analyze and filter the received data. All generated data may not be necessary. So it becomes critical to only store the data that is necessary.

- Storage layer: Storage is accomplished in this layer through storage virtualization.

- Security layer: Cybersecurity is a topic on which so much research is going on today. Protecting data is utterly important. Encryption and authentication are all done in this layer.

## VI. IoT and Big Data

In this section, we explore how fog computing is transforming the big data paradigms for the Internet of Things. We will look at the definition of big data and how IoT infrastructure is motivating the development of fog Computing.

According to Gartner, a global advisory and research firm, big data can be defined as "high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation."[3] More simply put, big data refers to a combination of complex and voluminous data that is received at a very high rate. This data can be used to provide insights, perform automation, and derive statistical inferences.

Today one can find IoT devices everywhere, from our home to offices to farms to big industries. Most modern homes use smart thermostats and smart locks. Offices now come equipped with energy-saving heating and lighting systems that reduce power consumption by automatically adjusting lighting and heating [2]. Farms nowadays have IoT tags for animals and crops. This helps farmers monitor the health and location of farm animals as well as monitor the humidity and nutritional content in the soil. But, the biggest use case of IoT devices is their ubiquitous use in Industries. Deploying IoT devices help speed up awareness and reduce response time to events, which is really important in industries such as manufacturing, transportation, oil and gas, and mining. Let us take an example of an oil rig. IoT sensors are deployed in oil pipelines that constantly measure the oil pressure. Any sudden change in pressure will cause the pumps to shut down automatically. Here a fast response is critical. In the time it takes for the pressure reading from the IoT sensor to travel to the cloud server for processing and the trigger signal to shut down the pumps to reach the sensor, the opportunity to avert a disaster might be lost.

These IoT sensors generate a large amount of data in the form of observational streams or time-series data. Moreover, the rate at which the data is transmitted varies from application to application. Applications that are time-sensitive, such as the ones mentioned above have a high data rate as compared to other applications such as smart utility meters, which may update say every 15 minutes. IoT sensors in oil rigs generate 500 GB of data every week whereas commercial airplanes generate 10 TB of data for every 30 minutes of flight [1]. Thus, it becomes impractical to send a large amount of data to the cloud for processing. Neither it is necessary, as most of the critical analysis does not require large-scale cloud-based processing.

In essence, IoT devices today generate high-volume, varying-velocity, and high-variety of data. But, today's cloud infrastructure is not designed to sustain such volume, velocity, and variety of data. The Statista Research Department forecasts that by 2030 there will be around 50 billion IoT devices in the world [4] . Such a huge network of interconnected IoT devices will inherently generate Terabytes of data every second. Sending all this data to the cloud for processing will choke the entire network. In such a scenario, fog computing can be our knight in shining armor. Since the compute units or fog nodes are much closer to the devices generating data, analysis of this data can be performed on these fog nodes reducing the dependence on the cloud. This will reduce net-

work traffic and will conserve network bandwidth. Moreover, by being close to the device, data analysis can be performed much more quickly. This will result in reduced latency and the difference between averting a disaster and a complete system failure.

In the next section, we develop a fog computing-based IoT infrastructure and demonstrate practically that by using fog computing, one can reduce latency as well as conserve network bandwidth.

## VII. Implemented Infrastructure

### A. System design

In this section, we first provide an overview of our IoT infrastructure followed by a deep dive into each of the components in the next sub-section. In the last part of this section, we will discuss how this infrastructure is implemented in code. A generic IoT network will have a set of sensors that relay important information such as pressure, humidity, temperature, etc. This information is sent to a server over a wireless channel such as cellular networks, satellite networks, Wi-Fi, Bluetooth, etc. for further processing.

To demonstrate the capabilities of fog computing in reducing the latency as well as network bandwidth, we have designed an accident detection IoT infrastructure that is able to detect road accidents and intimate first responders about the incident. Today, most of our smartphones come equipped with a variety of sensors such as accelerometer, gyroscope, orientation sensor, etc. We have used a smartphone to resemble an IoT sensor array on-board a vehicle, that is used to monitor the vehicle in real-time. This sensor array is synonymous with any IoT sensor deployed in the industry, in the sense that it generates real-time data of some important parameter. This real-time sensor data is transmitted via Wi-Fi to the fog node. The fog node continuously monitors the vehicle's state by analyzing this stream of sensor data. On detecting an accident, the fog node will find the nearest first responders and relay the accident location to them. Also, for future reference, if the information is needed about when and where the accident occurred, then we can also send the GPS and timestamp of the accident to the cloud server for storage. Moreover, as the number of such smart vehicles increase the amount of traffic will grow in the network. Now imagine if all this real-time data from the vehicles was transmitted to the cloud server for monitoring and processing. Doing so will not only cause the network to choke but will also result in a high delay. Fig. 3 shows the pictorial representation of our IoT infrastructure.

In short, this system is capable of generating a large amount of real-time data. Also, accident detection and response are time-critical as any delay can potentially lead to loss of life. Hence, this system is a good example of a high-volume and high-velocity data generating IoT infrastructure, which can be used to demonstrate the superiority of fog computing when it comes to IoT.

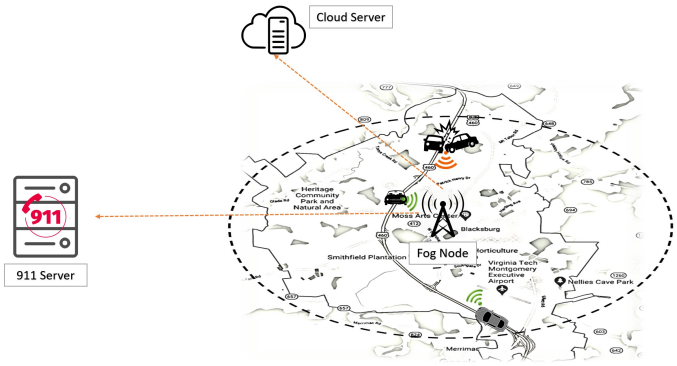


Fig. 3. Implemented architecture

### B. Sensors

We have used a linear accelerometer, orientation sensor, and GPS onboard a smartphone to monitor the state of the vehicle. Linear accelerometer gives the acceleration of the object in x, y, and z-axis, excluding the force of gravity. Fig. 4 shows the dimensions along which the accelerations are calculated. The orientation sensor gives the reading of orientation of the device in the x, y, and z-direction. Fig. 5 gives the pictorial representation of the three-axis of rotation of the device.

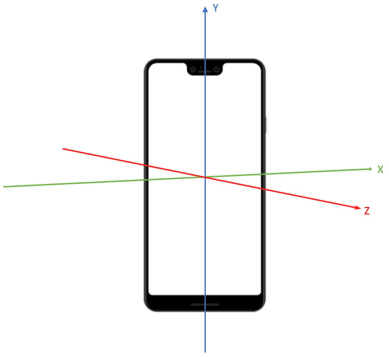


Fig. 4. X, Y, and Z orientation axes of accelerometer relative to a typical smartphone.

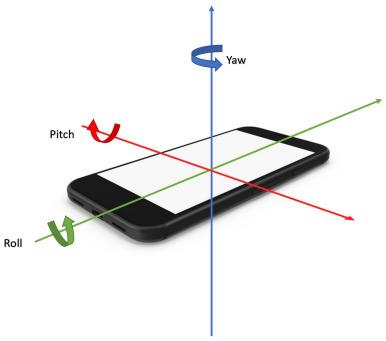


Fig. 5. Axis of pitch, yaw and roll in a typical smartphone

Finally, GPS, or global positioning system, provides us with the geolocation of the device.

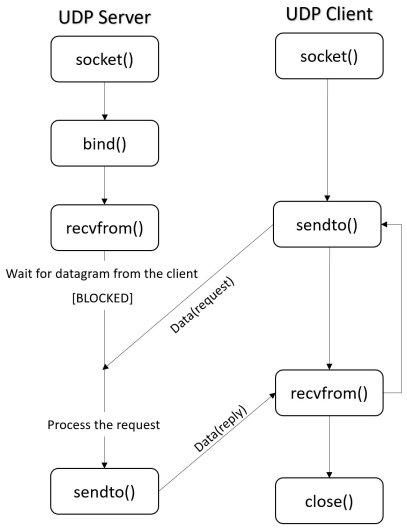


Fig. 6. UDP flow chart

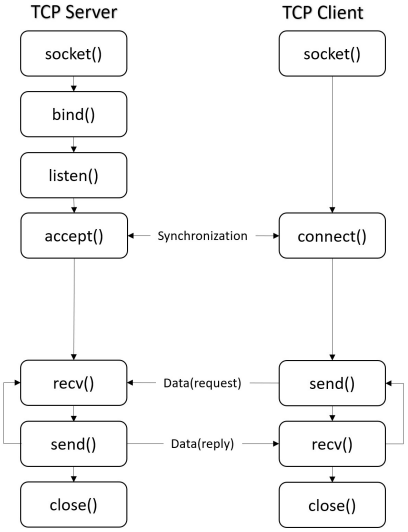


Fig. 7. TCP flow chart

### C. Connectivity

Each of these devices has wireless connectivity for sending data to the fog node. For the first phase of communication, we have used the UDP protocol to send data from the devices to the fog node. The reason for choosing the UDP protocol is that it is a connection-less protocol, i.e., there is no need to establish a connection between the device and the fog node before sending data. This reduces the latency in the system as we do not have to wait for a three-way handshake, like in a TCP protocol, before sending data. Moreover, the UDP packet size is comparatively smaller than a TCP packet. This reduces unnecessary overheads in the system. But, the main reason for using UDP over TCP is the speed of communication, which is important in time-critical scenarios such as this. While it is true that we may lose some packets as UDP is not a reliable communication protocol, in situations like this, it is better to lose some packets than to delay the rest. To accommodate multiple connections simultaneously, we have used threading while programming the fog node as well as the cloud server.

*a) Fog Server:* The fog server constantly monitors the vehicle's sate by processing the data stream coming from the vehicles. We created a fog server using python socket programming and hosted it on a desktop computer. The fog server is programmed to receive and transmit data using both TCP and UDP protocols. UDP is used for getting data from the vehicles, whereas TCP is used to communicate with the cloud server. Fig. 6 and Fig. 7 presents a flow chart of a generic UDP and TCP client-server architecture respectively. If the fog node detects an accident, it uses the GPS data from the vehicle to find the nearest hospitals and transmits accident location and search data to a hypothetical 911 server. We have used the Google Maps API to get the location of hospitals closest to the accident.

*b) Cloud Server:* We hosted the cloud server on Amazon web services rather than on a local network to get a better and realistic understanding of latency and bandwidth utilization when it comes to IoT data. The server was developed on a Linux operating system using python socket programming. In the event of an accident, the fog node transmits the accident location to the cloud server via a TCP connection for storage and analysis.

### D. Coding and logistics

*a) Sensor data:* To gather the sensor data we have used an android based application called IMU+GPS stream from Google Play Store. This application acquires sensor data from the smartphone and relays it to the fog server. Fig. 8, shows the screenshot of the application's "preference" tab. Under this tab, we can specify the target IP address and port number. The values given here are the IP address and port number of the fog node. Under the "toggle sensors" tab in Fig. 9, we can select different sensors, and stream the values acquired from these sensors to the fog node.

*b) Fog node and Cloud server:* In this subsection, we will discuss how the fog server and the cloud server were programmed. The algorithms for fog node and cloud server are given below as Algorithm. 1 and Algorithm. 2 respectively. The source code of both the fog node and cloud server can be found in the appendix.

*c) Accident Detection:* Vehicles in our infrastructure send raw sensor data to the fog node, where it is processed. Algorithm. 3 is used to detect an accident. The source code for accident detection can be found in the appendix.

**Algorithm 1:** FOG SERVER

**Result:** Fog node with TCP and UDP ports successfully created
initialization;
define fog node IP, fog node port#;
define cloud server IP, cloud server port#;
create UDP socket;
bind fog node IP, fog node port # to UDP socket;
connect to cloud server;
**while** *True* **do**
  recv data from client;
  process data from client;
  **if** *accident occur* **then**
    get GPS location of accident (Algorithm 3);
    find nearest first responders;
    send information about the accident to first responders;
  **else**
    continue monitoring vehicle;
  **end**
**end**

**Algorithm 2:** CLOUD SERVER

**Result:** Cloud server successfully created
initialization;
define cloud server IP, cloud server port #;
create TCP socket;
bind cloud server IP, cloud server port # to TCP socket;
listen for connections from client;
**while** *True* **do**
  accept connections;
  process data;
  store received data;
  send confirmation to fog node;
**end**

**Algorithm 3:** ACCIDENT DETECTION ALGORITHM

initialization;
define thresholds for $g_{force}$ & $orientation$;
decode data received from client;
$a_x, a_y, a_y$ = acceleration along x,y,z axis;
$g_{force} = \sqrt{(\frac{a_x}{9.81})^2 + (\frac{a_y}{9.81})^2 + (\frac{a_z}{9.81})^2}$;
$o_x, o_y, o_z$ = orientation along x,y,z axis;
$lat, long$ = GPS location;
**if** $g_{force} > threshold$ *or* $orientation > threshold$ **then**
  accident detected;
  get co-ordinates of accident
**else**
  continue monitoring sensor values;
**end**

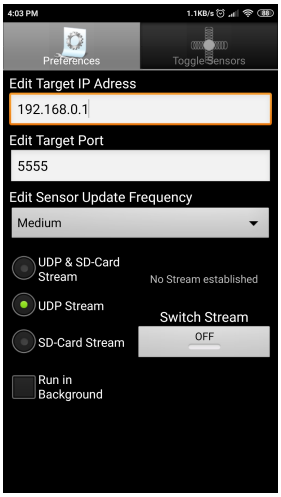


Fig. 8. Preferences tab on IMU+GPS application


Fig. 9. Toggle sensor tab on IMU+GPS application

## VIII. Results

### A. Latency

The application IMU+GPS is designed to be a unidirectional application, i.e. it can only transmit data to a client, but, is unable to receive anything from it. To calculate response time, it is important to know the round trip time. Form our analysis, we realized that the IMU+GPS application was generating a data stream at an average rate of 1.5 kilobytes/sec. Thus, we simulated the data on a different server that behaved like an IoT device. This server was capable of both sending and receiving data. To compare the response time for fog and cloud-based infrastructure, the data generated by this server was sent to the fog and the cloud for processing.

It should be noted that synthetic data from this server was used here only to demonstrate the difference between fog and cloud-based infrastructure. For accident detection and network utilization, real-time data from the smartphone's sensors were used.

Fig. 10, and Fig. 11 depict the response time for an IoT device communicating with the fog node and cloud server respectively. It was observed that the average response time for the fog node was around 2.825 ms, whereas, for the same process to run on the AWS cloud server, it took an average of 37.87 ms. This is around 13 times greater than the time it took for the fog node to process and respond to the request. This difference becomes crucial if we consider time-critical IoT applications like autonomous vehicles as mentioned in section II. Hence, the results clearly demonstrate that the latency of fog-based infrastructure is very low as compared to cloud-based systems.

Next, we increased the number of IoT devices to five to see the impact it had on the response time of the system. Fig. 12 and Fig. 13 presents the response time under this new condition for fog-based and cloud-based system respectively. From the graphs, we observed that it took an average of 60 ms for the AWS cloud-based system as compared to 4.15 ms for the fog-based infrastructure. Here, too fog-based infrastructure outperformed cloud-based systems.

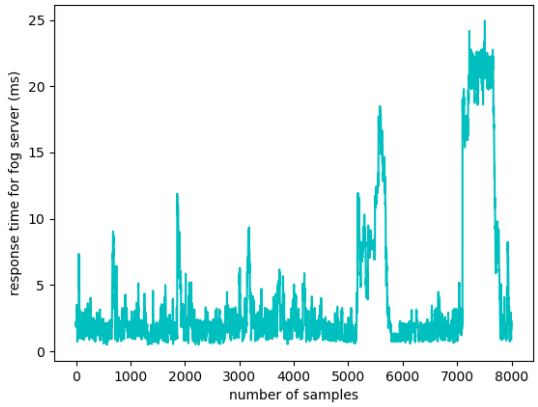


Fig. 10. Response time of fog node with 1 vehicle transmitting data

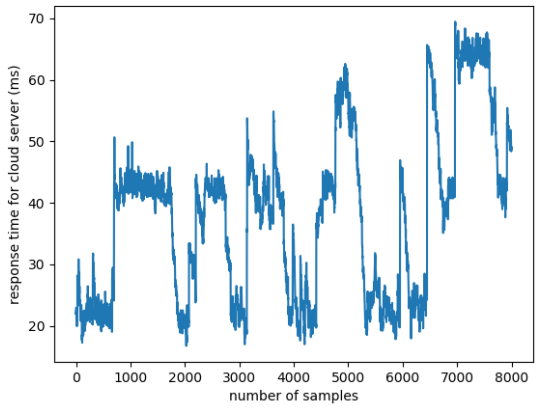


Fig. 11. Response time of cloud server with 1 vehicle transmitting data

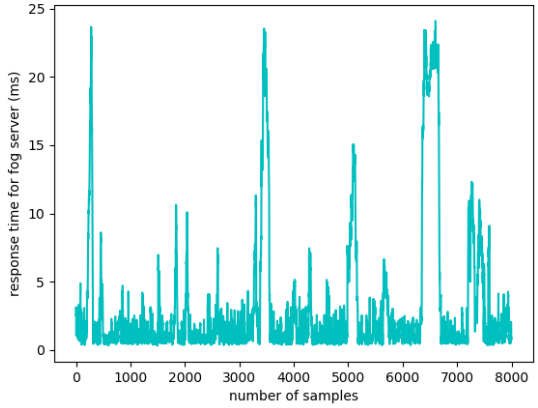


Fig. 12. Response time of fog node with 5 vehicles transmitting data

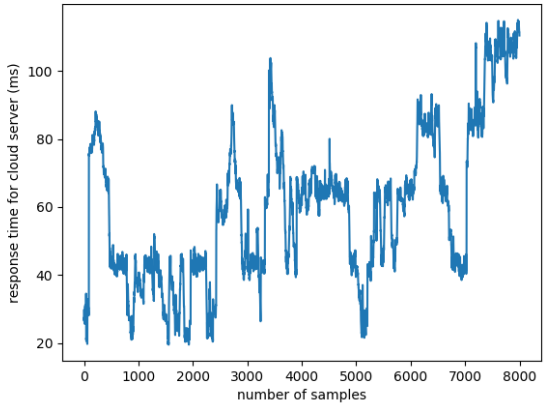


Fig. 13. Response time of cloud server with 5 vehicles transmitting data

### *B. Network Utilization*

Next, we looked at network utilization in the context of both fog and cloud-based systems. Using only cloud-based resources may result in the network getting overwhelmed by IoT data alone. Fig. 14 depicts the amount of data transferred by a single smartphone to the cloud for monitoring and processing the vehicle if fog-computing wasn't employed. As the number of such IoT devices increase the network will become more congested with only IoT packets. With fog computing, data will not travel far from the IoT device. Fog computing will help in avoiding the need for expensive bandwidth additions to industries by offloading, from the core network, gigabytes of network traffic generated by IoT devices.

## IX. Conclusion

The Internet of Things is fundamental in today's world in providing many benefits to industries and enhancing digital transformation. Fog computing will help to further accelerate this growth. The objective of this project was to look at how fog computing can provide better productivity and agility in terms of latency and network utilization as compared to cloud

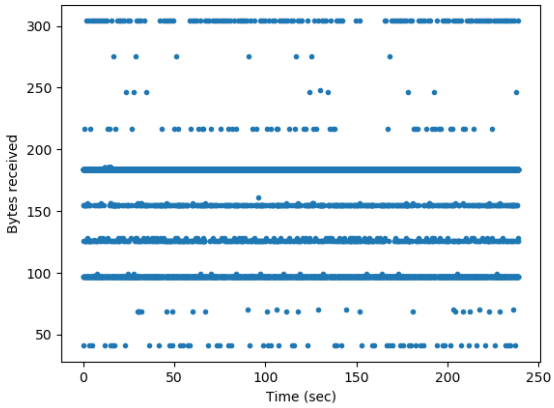


Fig. 14. Network Usage

computing. One of the aims of fog computing is to provide low latency so that it can be used in time-sensitive applications. From the results, it was observed that when data from the mobile application, IMU+GPS, was sent to the fog server, the latency was much less as compared to when the data was sent to the cloud server.

Further, it is important to note that IoT devices generate big data. Hence, to observe network utilization, we plotted the real-time data generated from the IMU+GPS application as shown in Fig. 14. This was only the data from one application. In reality, there are billions of IoT devices and storing all this data in the cloud servers will be problematic. We can infer from the results of this project that fog computing will be vital in the upcoming times as we move towards autonomous technology and 5G and more research needs to be done to make this system more robust and efficient.

## REFERENCES


[1] Fog computing and the internet of things: Extend the cloud to where the things are.

[2] Iot in the workplace: Smart office applications for better productivity.

[3] nformation technology, gartner glossary.

[4] Number of internet of things (iot) connected devices worldwide in 2018, 2025 and 2030.

[5] Marcos Assuncao, Rodrigo Calheiros, Silvia Bianchi, Marco Netto, and Rajkumar Buyya. Big data computing and clouds: Trends and future directions. *Journal of Parallel and Distributed Computing*, 75, 01 2014.

[6] Michael Beck, Martin Werner, Sebastian Feld, and Thomas Schimper. Mobile edge computing: A taxonomy. 01 2014.

[7] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, page 13–16, New York, NY, USA, 2012. Association for Computing Machinery.

[8] P. P. Gelsinger. Microprocessors for the new millennium: Challenges, opportunities, and new frontiers. In *2001 IEEE International Solid-State Circuits Conference. Digest of Technical Papers. ISSCC (Cat. No.01CH37177)*, pages 22–25, 2001.

[9] T. N. Gia, M. Jiang, A. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen. Fog computing in healthcare internet of things: A case study on ecg feature extraction. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 356–363, 2015.

[10] Shadi Ibrahim, Hai Jin, Bin Cheng, Haijun Cao, Song Wu, and Li Qi. Cloudlet: towards mapreduce implementation on virtual machines. pages 65–66, 01 2009.

[11] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan. Fog computing: Survey of trends, architectures, requirements, and research directions. *IEEE Access*, 6:47980–48009, 2018.

[12] B. Ravandi and I. Papapanagiotou. A self-learning scheduling in cloud software defined block storage. In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pages 415–422, 2017.

[13] Zohreh Sanaei, Saeid Abolfazli, Abdullah Gani, and Rajkumar Buyya. Heterogeneity in mobile cloud computing: Taxonomy and open challenges. *IEEE Communications Surveys Tutorials*, 16:369–392, 2014.

[14] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[15] Luis Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: Towards a cloud definition. *Computer Communication Review*, 39:50–55, 01 2009.

[16] Lizhe Wang, Gregor von Laszewski, Andrew Younge, Xi He, Marcel Kunze, Jie Tao, and Cheng Fu. Cloud computing: a perspective study. *New Generation Comput.*, 28:137–146, 04 2010.

[17] Xing Xie, Hua-Jun Zeng, and Wei-Ying Ma. Enabling personalization services on the edge. page 263, 01 2002.

[18] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P. Jue. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, 98:289 – 330, 2019.


## APPENDIX

Following are the codes for the IoT infrastructure that we established.

- File Name: $FOG\_NODE\_F.py$ shows fog server coding

- File Name: $CLOUD\_SERVER.py$ shows cloud server coding
- File Name: $car\_1.py$ shows sensor data simulation code.
- File Name: $get\_hospital\_location.py$ shows how to get nearest hospital location.
- File Name: $data\_rate.py$ shows how to calculate data rate
- File Name: $latency.py$ shows chow to calculate response time.