A Novel Fog-IoT based Framework for Improving Performance of Cloud Computing

Rahul Bhatt, Parag Jain, Mayank Aggarwal

Abstract**: *Internet-of-Things (IoT) has been considered as a fundamental part of our day by day existence with billions of IoT devices gathering information remotely and can interoperate within the current Internet framework. Fog computing is nothing but cloud computing to the extreme of network security. It provides computation and storage services via CSP (Cloud Service Provider) to end devices in the Internet of Things (IoT). Fog computing allows the data storing and processing any nearby network devices or nearby cloud endpoint continuum. Using fog computing, the designer can reduce the computation architecture of the IoT devices. Unfortunitily, this new paradigm IoT-Fog faces numerous new privacy and security issues, like authentication and authorization, secure communication, information confidentiality. Despite the fact that the customary cloud-based platform can even utilize heavyweight cryptosystem to upgrade security, it can't be performed on fog devices drectly due to reseource constraints. Additionally, a huge number of smart fog devices are fiercely disseminated and situated in various zones, which expands the danger of being undermined by some pernicious gatherings. Trait Based Encryption (ABE) is an open key encryption conspire that enables clients to scramble and unscramble messages dependent on client qualities, which ensures information classification and hearty information get to control. Be that as it may, its computational expense for encryption and unscrambling stage is straightforwardly corresponding to the multifaceted nature of the arrangements utilized. The points is to assess the planning, CPU burden, and memory burden, and system estimations all through each phase of the cloud-to-things continuum amid an analysis for deciding highlights from a finger tapping exercise for Parkinson's Disease patients. It will be appeared there are confinements to the proposed testbeds when endeavoring to deal with upwards of 35 customers at the same time. These discoveries lead us to a proper conveyance of handling the leaves the Intel NUC as the most suitable fog gadget. While the Intel Edison and Raspberry Pi locate a superior balance at in the edge layer, crossing over correspondence conventions and keeping up a self-mending network topology for "thing" devices in the individual territory organize.***

Index Terms**: *Cloud computing, fog computing, fog node, IoT, edge devices, edge computing*.**

I. INTRODUCTION

The Internet of Things (IoT) is a network of devices, or things, connected to the Internet that can communicate their status, respond to events or even act autonomously. The IoT devices discussed in this paper consist of sensors and actuators capable of transmitting their collected data via Bluetooth Low Energy. These devices are limited in power, processing and storage. The network topology connecting these devices varies but here we will remain focus on implementing a mesh topology. The main role of the IoT device is to periodically sample real world data. This data can be transmitted to other devices for actions such as decisions and storage [1].

Through the use of sensors and actuators, the IoT connects digital devices and everyday objects, bringing the connectivity of the Internet into the physical world. Some of the "things" devices can include sensors for recording the physiological measurements of humans [2]. So as the things environment becomes more complex the need for a reliable connection to the Internet either directly or via gateway devices becomes increasingly important.

Figure 1 shows the general layout between things (left) their gateway device (center) and the cloud (right). As new networks become available for the things, their bandwidth will increase, forcing pressure on the network between the gateway and cloud. This trend towards congestion continues as varieties of sensors and devices are developed with the intention of increasing the understanding of the environment where they are deployed [3].

Such a gateway device can be considered to occupy the edge of the network. It is the first device in the chain from thing to cloud. The device acting as a gateway is called an edge device. However, an edge device, such as a smartphone, may have constrained resources such as limited battery life, storage, and/or computational power. This limited functionality may require processing work to be offloaded to external resources [4].

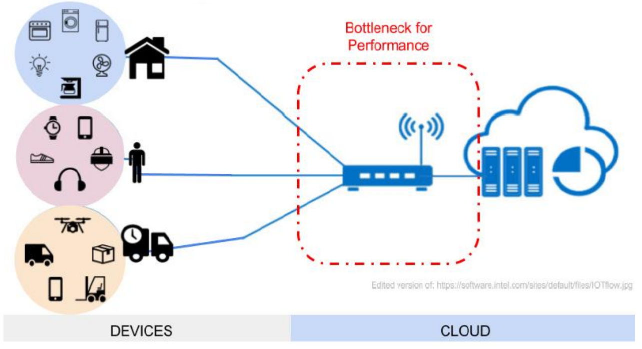

Bottleneck for Performance
Edited version of: https://software.intel.com/sites/default/files/IOTflow.jpg
DEVICES
CLOUD

Fig. 1: General layout between devices and cloud through gateway.

Revised Manuscript Received on October 31, 2019.

Latika Kharb, Professor, Jagan Institute of Management Studies, Delhi, India
Deepak Chahal, Professor, Jagan Institute of Management Studies, Delhi, India
Mayank Aggarwal, Professor, Jagan Institute of Management Studies, Delhi, India

A. Role of Cloud-IoT in Smart Communities and Smart Cities

If the edge device (named edge for its existence at the end of a network) relies heavily on external resources such as the cloud, it can put a huge stress on the network resulting in bottlenecks leading to and from the cloud [5]. This pattern towards blockage proceeds as assortments of sensors and devices are created with the aim of expanding the comprehension of nature where they are conveyed. This constantly floods cloud server farms at a fast rate making a requirement for the cloud to draw nearer to the gadget. In this manner, the worldview of fog figuring is progressively recognized as a valuable apparatus to build up remote insight with regards to IoT. In this paper we examine the ongoing improvement of a fog door and its job in arranging the procedure of information securing, molding, examination, and capacity. In particular the aim is to identify a way to reduce data transfer bottlenecks by pushing machine learning capabilities away from the cloud and closer to the things.

B. Intelligent fog centric network for IoT-driven smart communities

This research paper focuses on finding a balance in performance on devices within the local network and the cloud, while reducing the transfer of data to the cloud. The primary role of the fog gateway is to reduce the amount of, and reliance on, data sent to the cloud by orchestrating the process of data acquisition, conditioning, analysis, and short-term storage. The IoT devices are expected to collect and clean the data, as well as adapt to the information passed back from the fog gateway or another IoT device itself. While the cloud plays a vital role in the ecosystem overall, it is not widely researched in this paper [6]. More emphasis was given to the fog computing and its performance analysis.

Figure 2 shows the devices with limited functionality, computational power and resources appear at the bottom. These devices exist and interact with a small number of other “things”, and typically in a usual context. Contexts such as home monitoring, personal health and entertainment, asset delivery, and product manufacturing are just a few examples. The fog aims to connect contexts with mutual interests into smart communities. This broader, but still local network creates the fog environment that can deliver the most contextually relevant data to the cloud.

This paper provides insight on an approach for maintaining the real-time response of things; significant to the IoT industry. The response time is vital for the medical population, such as medical emergencies where every second counts. The infrastructure set up would allow the information leading up to the emergency to be easily accessible by medical personnel as the devices would be contextually aware. This can be extended further into telehealth living facilities to provide around the clock home monitoring with emergency services being alerted immediately in case of a serious problem [7].

Take for example the nursing home environment. In such communities there exists a larger than usual number of senior citizens living.

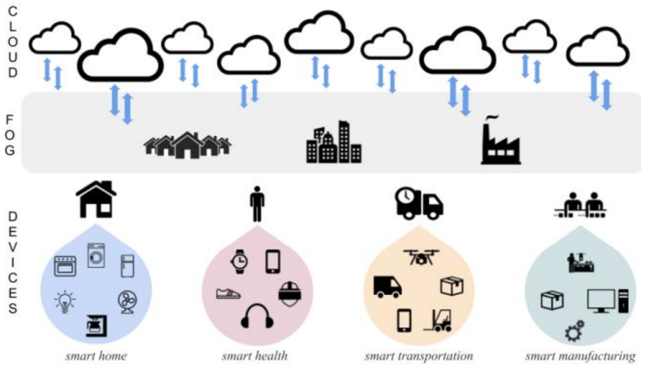


Fig. 2: The introduction of fog layer to cloud continuum.

Also, found among the community members are varieties of devices used in their daily lives. These devices are just as heterogenous in application as they are in capabilities. The goal of a smart telehealth living facility is to utilize the data collected by these devices to improve the wellbeing of the uses by providing them with near real-time feedback that can be descriptive as in “what”, diagnostic as in “why”, predictive as in “when”, and lastly prescriptive as in “how”. The fog layer gathers data a level broader than edge devices but geographically closer to the edge device than the cloud.

II. INTELLIGENT FOG CENTRIC NETWORK

Fog is a system-level horizontal architecture that distributes resources and services of computing, storage, control and networking anywhere along the cloud-to-thing continuum [8]. In essence the fog is a middle man aimed at managing the constrained resources across a larger pool of otherwise unused devices, as an effort to accelerate the speed at which decisions can be made accurately.

Figure 3 shows a general breakdown showing where services are performed in the path from device to cloud. Devices in IoT-driven communities with limited functionality, computational power and resources are specific in function. Moving through the continuum to the top we approach fog, which serves as the foundation for services to be incorporated between cloud and devices, but physically closer to the user than the cloud [9]. They also provide the closest functionality to that of a cloud, but do not have nearly as much processing power on board.

This fog-centric architecture addresses a specific subset of challenges in bandwidth, latency and communication challenges associated with the next generation of networks. The aim is to construct a local intelligence away from the cloud servers and close to the edge devices in IoT environments. The fog works in conjunction with both IoT devices and the cloud by combining the use of low-latency storage devices, computational power not available to constrained edge devices, low-latency local communication. All in conjunction with wide area communication, management for network measurements controls and configurations nearest the end user [10]. In its simplest form, the fog moves the cloud closer to the user.

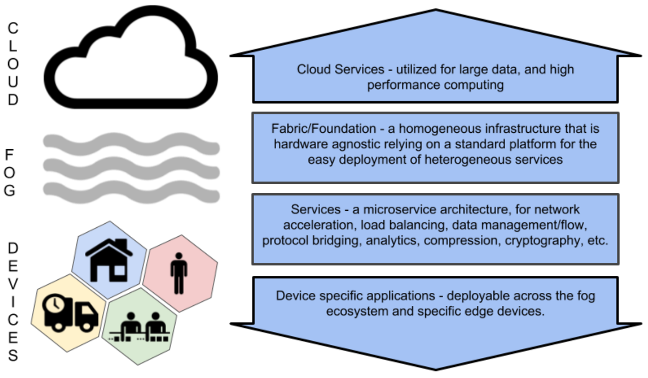

CLOUD
FOG
DEVICES
Cloud Services - utilized for large data, and high performance computing
Fabric/Foundation - a homogeneous infrastructure that is hardware agnostic relying on a standard platform for the easy deployment of heterogeneous services
Services - a microservice architecture, for network acceleration, load balancing, data management/flow, protocol bridging, analytics, compression, cryptography, etc.
Device specific applications - deployable across the fog ecosystem and specific edge devices.

Fig. 3: List of services performed from device to cloud.

The conjugation of these fog features mitigates the effects of wide area network latency and jitter concerns caused by traffic congestion proves costly to time sensitive tasks, frequently involving medical devices, cyber-physical systems, and industrial equipment. In such contexts, uninterrupted service is key, while connectivity to the cloud is bound to be intermittent [11]. By providing uninterrupted service along with prioritized service we can create several key features that were not feasible before the integration of fog.

The fog provides four key features at a critical time in the growth and acceptance of the internet of things. While being ever mindful of security threats it strives to allow us the ability to make meaning from previously unstructured junk data collected and streamed to the cloud:

- The first feature is cognition. Since the fog device is more closely connected to its local network, it can determine where to carry out computational, storage, and networking tasks. Fog devices are built to be aware of the primary end users specific requirements.
- The second feature is efficiency. Fog can orchestrate the dynamic pooling of resources from previously unused end user devices. These resources can include computational power, low-latency storage, and control functions currently spread across the cloud to thing continuum. This would allow applications to make use of idle user owned edge devices, such as phones, tablets, tvs, and more. Note that this continuum is just that, a smooth gradient of devices types. Figure 3 provides a more detailed explanation of this continuum [12].
- The third feature is agility. This allows developing the technology and integrating it into the systems created by manufacturers, product developers, and the like, to affordably scale with their needs under a common infrastructure thereby helping to propel rapid innovation. This can come in at the angle of testing and experimentation for advanced systems and quickly pivot towards implementing the application. The motivation for this goes hand and hand with openness, provided by the use of open standards. The openness allows developers to work together creating a collection of diverse applications, where individuals may develop standard application interfaces (APIs) and open software development kits (SDKs) to manage the proliferation of new devices into the continuum. This flow for innovation would follow the model of develop, deploy and then operate, all within the open fog system.
- The fourth (and perhaps the most critical) feature is latency. The fog provides a feasible way for complex systems to return near real-time feedback. This feedback provides processing and cyber-physical system control to the end user/device. This enables data networks to develop more intelligence from analytics of network edge data. It can provide this intelligence in time-sensitive situations with the potential to save thousands of dollars in areas like nuclear reactors and lives in the area of advanced medical devices. Reduced latency can enable embedded artificial intelligence (AI) applications to react in milliseconds, unnoticable by humans, but providing more time for critical tasks such improving localized cyber security checks [13,14].

A. Use-cases for fog centric IoT-driven communities

As this fog technology grows, it opens a gateway for new, previously impractical applications. The devices and data used in these new potential applications vary widely. This heterogenous environment creates a need for integrating these new applications into the larger community for further innovation. A standard interface would therefore be of great value. While no standard has yet to be fully accepted, there are attempts at developing the systems that will be using the interface. Some of these systems are applied in areas such as smart cities, health services, agriculture and general transportation [15, 16]. These cases will be expanded on, but it is instructive to note that the big picture fog platform must orchestrate the available devices and resources to create a symphonic experience of various technologies.

An outline of the types of networks that are most applicable for communication between the layers of IoT shows in figure 4. The idea is to highlight the appropriate range and bandwidth limitations given the network communication focal point.

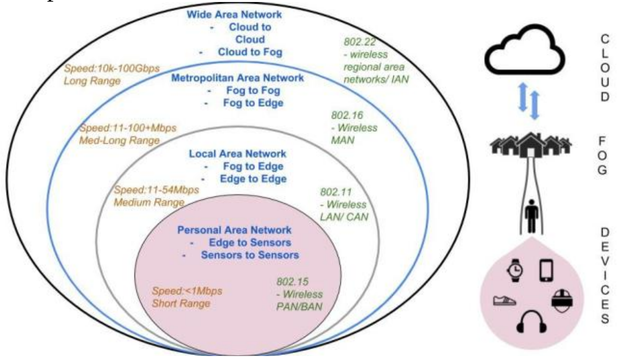


Fig. 4: Types of networks applicable for communication between layers of IoT.

III. METHODOLOGY

The methodology for this paper consists of goals in the development, implementation, and analysis of the overall system. The development of the framework will be set up with example data to be processed for testing purposes. The involved devices will be set up in a mesh topology with the fog gateway devices setup in a star topology. The fog will consist of three different testbeds. Each will be running the most appropriate OS for the device and will be capable of managing communication with the things via Bluetooth or WiFi, as well as provide access to a cloud service.

Once the sensors, fog test beds, and cloud have been deployed, communication throughout the system will be addressed. Once communication is in place, machine learning algorithms are deployed on the fog nodes. Rankings on memory and power usage are recorded and analyzed to evaluate the performance of the framework. This evaluation will take into account the amount of data being transferred through each stage of the cloud-to-things continuum. It will also consider memory requirements for particular algorithms. After the framework passes for simple workloads, it will be incorporated into the mock day to day operations of a smart nursing home where it will serve 35 clients. While it is being deployed, an analysis of the load of data and the fluctuations in latency will be observed. Any anomalies or difficulties will be looked into and discussed.

This paper will provide an explanation of the fog framework including protocols, topologies, and hardware along with the reasons for choosing each of the devices. The framework deployment process will be explained in enough detail for replication, if desired. Each of the layers shown have expected applications and system constraints. IoT Devices being the most limited can communicate at is shown in figure 5.

A. Fog architecture - from atom to Adam

As we further our discussion on the fog architecture, it is instructive to consider the whole picture. The goal of fog is to minimize the latency of the reaction of an application. It seems appropriate, then, to examine the flow of information, from atom to Adam the user. The thing device sensing, the edge node filtering and relaying, the fog orchestrating analytics and data flow, the cloud developing deeper insights, all tied to the user whom the technology is designed for play roles in the design considerations of a fog platform. As the hardware for technology becomes geometrically smaller and lower in power consumption, a door opens to new, previously impractical use cases. Following this trend is the increased intimacy technology can maintain with its user, particularly in the case of new, wearable technology.

In this light, we will discuss the components involved through the lens of Smart Glove, an electronic textile smart glove system designed for PD.

Smart Glove as a Thing device - Flex Sensors, Inertial Motion Units, and Microphones: This information is at the tip of your finger, literally. The concern is reliably and accurately collecting all the relevant information. We need to understand how the hand is positioned in space. The glove uses single angle flex sensors and inertial motion units (IMU). So what are these devices and what do they provide for the application?

The things used in the testbed perform a variety of measures and are tied to applications that require the derived data points to be accurate and transported appropriately. The first thing device is a set of Flex Sensors tied to a glove in a application called Smart Glove. Along with measuring the flex for each finger, we needed to quantify the movement of the hand. We do this using an inertial motion unit (IMU). This device is found in many handheld devices, including smartphones. This device measures the hand's acceleration, orientation, and direction. It exploits physical phenomena and converts that into digital values.

The gyroscope has a varying geometry but uses a reference point to maintain a "level" plane. The angular changes in the device's current orientation and the "level" plane help us identify tilt. Whereas, the magnetometer measures the magnetic field and uses that to derive a sense of direction. Our gloves use the LSM9DS1 set to provide a linear acceleration full scale of ±8g, a magnetic field full scale of ±8 gauss and an angular rate of ±500 dps. We set the sample rate to 250Hz. This data is initially stored as an unsigned 16-bit integer. These same values are passed directly to the Bluetooth payload. One concern in design for the glove is this passing of information. The IMU communicates serially so the wires connecting the IMU to the Bluetooth device need the bandwidth for a clock.

B. Fog Edge Node - An Android Phone, Intel Edison and Raspberry Pi

The Fog edge nodes plays a role in protocol bridging, data cleaning, and orchestrating the direction of the data flow. All of the sensors are connected to the MCU which collects the data from the flex sensors. In order to preserve the subject's freedom of movement, we designed an application to wirelessly collect data from the glove. The data from the Arduino board is sent to the Smart Glove smartphone application. Smart Glove is designed to run on any Bluetooth 4.0 enabled device running Android 4.4 or higher.

The glove streams its data to the users edge device. This edge device prompts the user on the correct gestures to make. The edge device interacts with the fog for user authentication/authorization and feature extraction.

IV. Implementing the Fog Distributed MQTT

MQTT is a machine to machine (M2M) publish/subscribe messaging protocol, designed to be lightweight for IoT devices. It is designed to be scalable for utility sized data. This is ideal for fog, where the aim is to send clean structured data rather than massive quantities of data. Edge computing from the clients thereby allows for reduced data transfer. Properly implementing this messaging protocol can assist in providing services that fit with the pillars of fog.

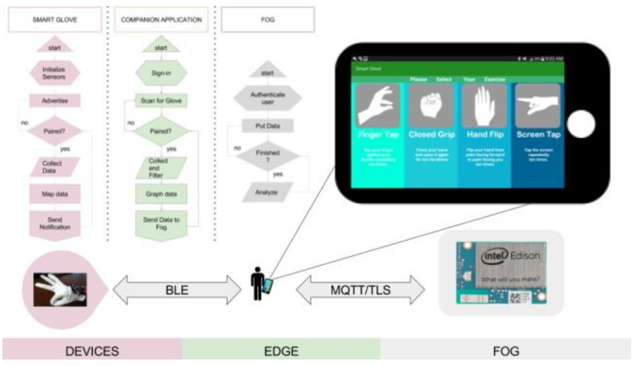
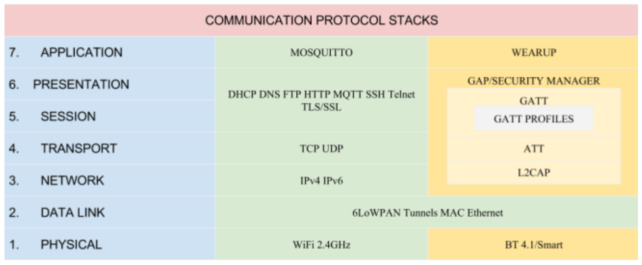


Fig. 5: Framework deployed for the Smart Glove application.



COMMUNICATION PROTOCOL STACKS		
7. APPLICATION	MOSQUITTO	WEARUP
6. PRESENTATION	DHCP DNS FTP HTTP MQTT SSH Telnet TLS/SSL	GAP/SECURITY MANAGER
5. SESSION		GATT GATT PROFILES
4. TRANSPORT	TCP UDP	ATT
3. NETWORK	IPv4 IPv6	L2CAP
2. DATA LINK	6LoWPAN Tunnels MAC Ethernet	
1. PHYSICAL	WiFi 2.4GHz	BT 4.1/Smart

Fig. 6: OSI Layering for the communication protocol.

Security is one of the most crucial pillars of fog. The data being sent to and from all the users devices needs to be protected from the outside and inside. This can only be done by implementing the latest security protocols on all layers within the communication functions. Figure 6 shows the layers involved and the protocols used at each one. We first delve into the MQTT application and its configuration, then discuss a framework for the layers it rests upon.

To be integrated into the fog infrastructure, all nodes must comply with transport layer and socket security. Putting this into action has become more automated due to a public push for security on all data. A free to use service called LetsEncrypt generates SSL certificates through its application programming interface (API). This API relies on answering challenges. We can install and use certbot to provide automatic communication for any of the cryptographic challenges. As the certificates from LetsEncrypt will be standard domain validation certificates, they can be used for transport layer security (TLS) applications like MQTT.

As fogs must be able to communicate with other fogs and the cloud, they should be accessible with standard addressing, such as IPv4, IPv6 or IPSec, just as typical web servers are. For maintaining this within any subnets of the private IPs, they can be pointed to by the DNS server for domain validation; but only interactive from within the private network.

The use of these certificates can be quickly enabled by pointing services like mosquitto. Mosquitto, now housed by Eclipse, is a lightweight implementation of the MQTT protocol. Upon initial setup, mosquitto will provide readable plaintext data transfers. It is freely accessible for any device to join. This means that any device can begin to propagate and collect any data available from that broker. This is clearly not meant to be used off the shelf and must be used with security measures chosen by the designer. Luckily, mosquito is capable of handling much of this automatically. We only need to let the configuration file know the location of the keys and certificates generated by LetsEncrypt to secure the transport layer. We can also move the listening port from 1883 to 8883, the standard port for secure MQTT data communication.

After finishing the configuration of MQTT with a secure socket layer, unique ids, usernames, and passwords will be assigned on devices looking to publish/subscribe for additional protection. All of these will be used to create scenarios of authentication in conjunction with authorization. A potential security weakness lays in the ability to allow one set of usernames and passwords for multiple device connections. This was permitted for convenience in this paper, but with the certificate file, certified authority file, and private server key, a reasonably secure MQTT+SSL is set for use.

For Fog nodes or edge devices looking to host a dynamic web interface, the additional implementation using MQTT over websockets would allow for integration with JavaScript. This only requires changing the listening port to 8083 and defining the protocol as websockets within the mosquito configuration file. The dynamic web hosting implementation was not used for this paper but can be adapted for future work. Now scalability depends on the fog device's ability to broker edge devices and data demands at any given time. This leaves edge devices free to stay in place as the larger ecosystem grows. That is why it is important that MQTT allows the fog device to filter queries based on the topics and types of messages.

As a note regarding queueing, MQTT incoming messages are stored until picked up by the client. This can be adjusted based on Quality of Service (QOS) parameters. It is important to keep this in mind as the broker must deliver the message to every client, not just one. Aside from QOS, queueing can be managed specifically by topics. The more specific a topic, the fewer clients will need to receive messages on that topic, further reducing the queue. The fog can also release holds after a set time to avoid build ups caused by offline devices.

MQTT brokers can be successfully deployed on embedded devices, mobile phones and other edge and fog devices. The only requirement is networking capabilities supporting a TCP/IP stack and room for an executable of ~120kB consuming ~3MB of RAM per 1000 clients. Eclipse reports that tests with 100,000 clients at modest message rates as successful [545]. For further interoperability with other applications the oneM2M, a Java based horizontal framework, can be used. It was not used in this paper but could be considered for a commercial deployment.

V. AUTHENTICATION AND AUTHORIZATION

A network layer common in VPNs is one layer suitable for a fog node. TLS/SSL is suitable for encrypting on the transport, but must also be usable for edge devices. MQTT usernames and passwords can help by adding authentication on an application layer. Furthermore, MQTT can encrypt payloads for full transport encryption. While a nesting doll of encryption is useful for security purposes, implementing one under the constraints of lower end edge devices can be trying.

From the MQTT application layer, further authentication can be achieved using the client id creatively. The MAC address, the serial number, or a combination of the two could be used for the makeup of a 36 character UUID. This could be on top of a previous transaction of a X.509 certificate during the TLS handshake. This is only the first step because an appropriately authenticated client could still be ill intentioned or behave inappropriately. To prevent this, use of authorization for requests of topics, operations, and service levels should be maintained by the fog node through topic permissions. This chain of trust can maintain a reliable lightweight messaging system. Figure 7 shows the Smart Glove prototype glove being worn while the wearer performs a finger tapping exercise. To the right is a sample data plot from that same experiment overlaid with markers at each peak as determined by the fog device.

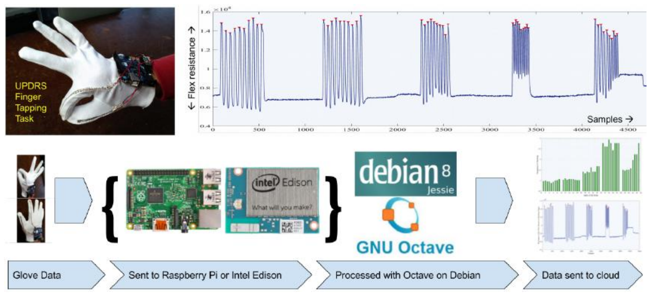

UPDRS Finger Tapping Task
Flex resistance
Samples
debian 8 Jessie
GNU Octave
Glove Data
Sent to Raspberry Pi or Intel Edison
Processed with Octave on Debian
Data sent to cloud

Fig. 7: Movement detection by Fog devices.

VI. Results and Discussions

The developed benchmarking process tests timing, CPU load, and memory load, and network measurements. The results were generated by combining the information collected from the Octave Function Profile and the Linux program collectd. The Profile function running inside Octave collected timing specifics for the particular algorithm, while the collectd program collected timing, CPU and memory loads for the entire process.

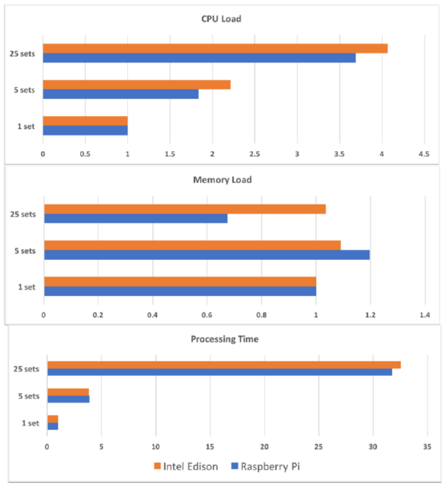
The definitions tied with CPU utilization, memory consumption, network traffic, and overall load, are determined by the collectd application. Definitions are also presented here. First, network traffic is "information about the traffic (octets per second), packets per second and errors of interfaces (of course number of errors during one second)." Next, memory consumption is memory "used, buffered, cached and free – as in consuming power but providing no use -- with the units of Bytes." CPU Utilization is "the amount of time spent by the CPU in various states, most notably executing user code, executing system code, waiting for IO-operations and being idle." Approximately 100 operations can be scheduled per second. If this were reliably precise, this could provide a stable percentage base. However, this is not the case and the use of percentage as a unit for utilization has been removed to avoid misinterpretation. Finally, "The system load is defined as the number of runnable tasks in the run-queue and is provided by many operating systems as a one, five or fifteen minute average".

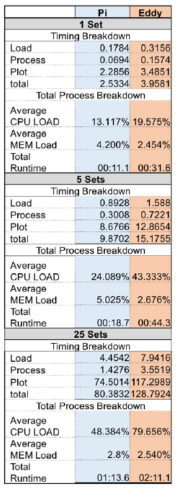
Two other parameters mentioned in the benchmark are latency and processing time. Both of these measurements were taken using a tic toc method in Python. Processes would start with a tic and end in a toc. The current processor time is kept as a floating point expressed in seconds. The difference between the toc and the tic is stored as processing time. Latency was recorded by measuring the response time of a "ping" payload. One fog node would post the "ping" message in the same queue as the data stream. A toc was called when the "ping" was returned. Again, the difference between a toc and a tic was recorded.

VII. Resulting Performance

The measurements are shown in the total process breakdown in the figure below; they include the load added on by starting an instance of Octave. While the Intel Edison and Raspberry Pi were set up identically, the Intel NUC -- being much less constrained in processing power and storage was capable of implementing a few Python libraries rendering Octave unnecessary. This is important to note in the results, as each instance of Octave required additional startup time. We observed that the run time for an Edison (4s) was much greater than that of the Raspberry Pi (2.5s), and that an increase in data sets (N) produced a processing time of order Nlog(N). The Raspberry Pi could complete the process almost 2x faster than the Edison. Furthermore, it could scale to 125+ datasets while the Edison would gracefully crash.

The Intel Edison was not capable of maintaining the timing window needed for 300ms samples to be processed and published to subscribers. Furthermore, the Intel Edison failed to remain active during the more demanding sections of benchmarking. While neither device provides enough resources to be appropriate fog platforms, the Raspberry Pi is capable of acting as protocol bridge, or as an edge device. Figure 8 shows the graphs comparable performance between the Intel Edison and Raspberry Pi. It can be seen that while the computations did not put a major stress on the CPU Load for either device, the growth in processing time quickly became unreasonable for a fog device and thus indicated a termination of increasing data sets [17].





	Pi	Eddy
1 Set		
Timing Breakdown		
Load	0.1784	0.3156
Process	0.0694	0.1574
Plot	2.2856	3.4851
total	2.5334	3.9581
Total Process Breakdown		
Average CPU LOAD	13.117%	19.575%
Average MEM Load	4.200%	2.454%
Total Runtime	00:11.1	00:31.6
5 Sets		
Timing Breakdown		
Load	0.8928	1.588
Process	0.3008	0.7221
Plot	8.6766	12.8654
total	9.8702	15.1755
Total Process Breakdown		
Average CPU LOAD	24.089%	43.333%
Average MEM Load	5.025%	2.676%
Total Runtime	00:18.7	00:44.3
25 Sets		
Timing Breakdown		
Load	4.4542	7.9416
Process	1.4276	3.5519
Plot	74.5014	117.2989
total	80.3832	128.7924
Total Process Breakdown		
Average CPU LOAD	48.384%	79.656%
Average MEM Load	2.8%	2.540%
Total Runtime	01:13.6	02:11.1

Fig. 8: Performance comparison between the Intel Edison and Raspberry Pi.

VIII. Fog Performance Review

Some of the concerns raised by the testbeds include, power consumption, reliability, and cryptography. The performance each of these devices could provide varied drastically. These concerns were most prevalent in the Intel Edison and Raspberry Pi, and less so in the Intel NUC.

The Intel Edison was unable to maintain a steady uptime when faced with more than 25 clients. The primary reason for this was not in managing the wireless clients, but rather in running multiple instances of octave. This caused the device to crash even when each instance of a client connected was contained within separate virtual environments for processing. Furthermore, when the Edison was running these algorithms, it was warm to the touch.

This heating indicated that this device was not appropriate to use as fog device, and should instead remain an edge device, for bridging protocols and devices onto the wide area networks previously not reachable to those devices constrained by wired communication and personal area network limits.

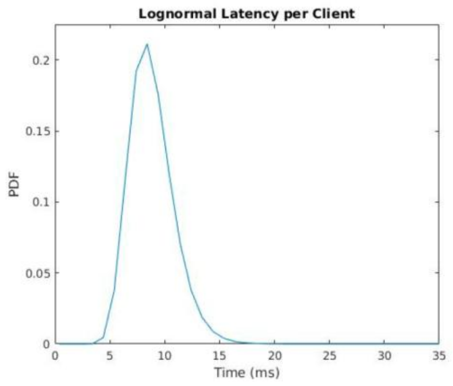


Fig. 9: Latency of the fog for the Intel NUC.

The Raspberry Pi was capable of maintaining a reliable connection to the various clients, unlike the Edison. However, it suffered from slow processing times. I believe this can be overcome by deploying more appropriate algorithms. It was noticed that the numpy and scipy python libraries weighed down the upstart time for each instance of a virtual environment. By importing only the required packages this time could be reduced.

Even after that alteration, peaks could not be determined quickly enough for a near real-time response to data received from a client. These findings lead to the conclusion that while the Raspberry Pi is a better fit as an edge device than the Edison, it does not provide the reliability and data crunching required of a fog device.

Figure 9 shows the latency of the fog for the Intel NUC. The latency was fit to a Log-Normal Distribution where the mean was 8.511ms with the 95% confidence interval of 8.251 and 8.771. This pdf was consistent was each client. The test ran with a maximum of 35 clients connected simultaneously.

IX. CONCLUSION AND FUTURE SCOPE

Throughout this paper we discussed the paradigm of fog computing. We identified a few strong examples such as LinkNYC and the Efficient Utilities projects, showing how the fog is increasingly a useful tool to establish remote intelligence in the context of IoT. We further tested three different testbeds as potential fog devices. Each of these testbeds were responsible for orchestrating the process of data acquisition, conditioning, analysis, and storage for up to 35 clients simultaneously. The aim was to identify a way to reduce data transfer bottlenecks by using these testbeds, in particular by pushing machine learning capabilities away from the cloud and closer to the things. While the Intel Edison and Raspberry Pi were determined to fit better in the edge layer, we found that the Intel NUC was a capable device for the fog layer.

The tests performed on each of these devices were not fully comprehensive in that they neglected to include penetration testing. This will be an important focus when it is time for real-world deployment. However, the setup for a reasonably secure environment existed. The entire network existed within a virtual private network with TLS/SSL enabled for all TCP/IP connections within the network, and Bluetooth security enabled for connections outside of this reach. The MQTT connections forced SSL connection of the designated secure port, 8883, for all clients connecting from outside of the network. Furthermore, initiating communication required a username and password. Inside of this, each user had particular permissions limiting their control over communications. No cellular communication protocols were used.

This fog layer could open the gates for a variety of future projects and research focuses. For one, it could be a useful tool in mitigating the effects of a security breach at the infrastructure level. The level of connectivity enabled by the fog will provide opportunities to improve the quality and depth of feedback to telehealth queries. The future of context awareness with a shorter response time increased reliability could include increased accuracy in machine learning algorithms in a variety of fields.

REFERENCES

1. S. DRA¨XLER, H. KARL, AND Z. A. MANN, "JOINT OPTIMIZATION OF SCALING AND PLACEMENT OF VIRTUAL NETWORK SERVICES," IN IEEE/ACM CCGRID, 2017.
2. M. PEUSTER AND H. KARL, "UNDERSTAND YOUR CHAINS: TOWARDS PERFORMANCE PROFILE-BASED NETWORK SERVICE MANAGEMENT," IN IEEE EWSDN, 2016.
3. "PROFILE YOUR CHAINS, NOT FUNCTIONS: AUTOMATED NETWORK SERVICE PROFILING IN DEVOPS ENVIRONMENTS," IN IEEE NFV-SDN, 2017.
4. "DATA FROM THE EXPERIMENTS IN THIS PAPER." [ONLINE]. AVAILABLE: HTTPS://UNI-PADERBORN.SCIEBO.DE/INDEX.PHP/S/G9Q2HMUNG4N8LEG
5. D. GMACH, J. ROLIA, L. CHERKASOVA, AND A. KEMPER, "CAPACITY MANAGEMENT AND DEMAND PREDICTION FOR NEXT GENERATION DATA CENTERS," IN IEEE ICWS, 2007.
6. S. RASOOLZADEH, M. SAEDPANAH, AND M. R. HASHEMI, "ESTIMATING APPLICATION WORKLOAD USING HARDWARE PERFORMANCE COUNTERS IN REAL-TIME VIDEO ENCODING," IN 7TH INT. SYMPOSIUM ON TELECOMMUNICATIONS (IST), 2014.
7. Q. FAN AND Q. WANG, "PERFORMANCE COMPARISON OF WEB SERVERS WITH DIFFERENT ARCHITECTURES: A CASE STUDY USING HIGH CONCURRENCY WORKLOAD," IN IEEE HOTWEB, 2015.
8. X. XU, T. XU, Y. YIN, AND J. WAN, "PERFORMANCE EVALUATION MODEL OF WEB SERVERS BASED ON RESPONSE TIME," IN IEEE CONF. ANTHOLOGY, 2013.
9. A. V. DO, J. CHEN, C. WANG, Y. C. LEE, A. Y. ZOMAYA, AND B. B. ZHOU, "PROFILING APPLICATIONS FOR VIRTUAL MACHINE PLACEMENT IN CLOUDS," IN IEEE CLOUD, 2011.
10. I. GIANNAKOPOULOS, D. TSOUMAKOS, N. PAPAILIOU, AND N. KOZIRIS, "PANIC: MODELING APPLICATION PERFORMANCE OVER VIRTUALIZED RESOURCES," IN IEEE INT. CONF. ON CLOUD ENGINEERING, 2015.
11. [ONLINE]. AVAILABLE: HTTPS://PEACH.BLENDER.ORG/DOWNLOAD/
12. [ONLINE]. AVAILABLE: HTTP://WWW.IMDB.COM/TITLE/TT2608732/
13. [ONLINE]. AVAILABLE: HTTP://WWW.IMDB.COM/TITLE/TT3996164/
14. [ONLINE]. AVAILABLE: HTTPS://WWW.YOUTUBE.COM/WATCH?V=ERTIWNFX93
15. A. J. SMOLA AND B. SCH¨OLKOPF, "A TUTORIAL ON SUPPORT VECTOR REGRESSION," STATISTICS AND COMPUTING, VOL. 14, NO. 3, PP. 199–222, 2004.
16. M. ILLIAN, "PREDICTION OF RESOURCE REQUIREMENTS AND PERFORMANCE OF VIRTUALISED NETWORK FUNCTIONS IN A VIDEO STREAMING CONTEXT," BACHELOR'S THESIS, PADERBORN UNIVERSITY, 2017.