2017

An Intelligent Fog Centric Network for IoT-Driven Smart Communities

Nicholas Constant
University of Rhode Island, kabuki4774@gmail.com

AN INTELLIGENT FOG CENTRIC NETWORK FOR IOT-DRIVEN SMART COMMUNITIES

BY

NICHOLAS CONSTANT

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

UNIVERSITY OF RHODE ISLAND

2017

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

OF

NICHOLAS CONSTANT

APPROVED:

Thesis Committee:

Major Professor Kunal Mankodiya

Bin Li

Manbir Sodhi

Nasser H. Zawia
DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND
2017

ABSTRACT

The Internet of Things (IoT) architecture currently being implemented in commercial applications does not fully realize the potential of IoT devices. The IoT devices themselves lack the computational power to perform significant and real-time feedback for the data being gathered at edge devices. The current architecture places most of the computational burden on various cloud servers. This creates a performance bottleneck due to unpredictability in network latency and reliability.

The amount of data flowing through the cloud-to-things continuum will only continue to grow, increasing stress on the cloud and network. This type of bottleneck creates difficulties for devices, such as those in Smart Communities, that require reliable connectivity with in depth real-time feedback. These communities host a level of context that is unique to each of them and can be used to help maximize the usefulness of the local network while sending more contextually relevant information back to the cloud for deeper learning.

The master thesis research was aimed at developing, implementing, and evaluating a Fog Computing based IoT system. The deployment of the framework on several IoT devices will be set up with example data to be processed. These devices will be set up in a mesh topology with fog gateway devices. In this research, we developed three testbeds to test the fog computing architecture and its performance in IoT applications where smart communities are targeted. Each of the three different testbeds will run the most appropriate OS for the device and will be capable of managing communication with the things via Bluetooth, as well as providing access to a cloud service.

The aims is to evaluate the timing, CPU load, and memory load, and network measurements throughout each stage of the cloud-to-things continuum during an experiment for determining features from a finger tapping exercise for Parkinson's Disease patients. It will be shown that there are limitations to the proposed testbeds when trying to handle upwards of 35 clients simultaneously. These findings lead us to an appropriate distribution of processing the leaves the Intel NUC as the most appropriate fog device. While the Intel Edison and Raspberry Pi find a better footing at in the edge layer, bridging communication protocols and maintaining a self-healing mesh topology for "thing" devices in the personal area network.

ACKNOWLEDGMENTS

This thesis could not have been possible without the careful and patient guidance of Professor Kunal Mankodiya, for this and much more, I express my deepest gratitude. The strong support and timely posed questions from Professor Bin Li is greatly appreciated. I would like to thank Professor Manbir Sodhi for his continued support, ideas, and criticisms. A great and truly sincere appreciation is given for Dr. Tim Toolan for his time and patience assisting with the appropriate deployment and debugging of the networking involved with this system.

I would like to extend my gratitude towards all of the members within the Wearable Biosensing Lab for whom have supported this endeavor, providing advice, support and motivation. Without this incredible lab the project would not have been a success. I am thankful to Alyssa Zisk who rigorously proofread the thesis chapters.

Last, but surely not least, I would like to thank my parents for their continuing support and consistent pushes to keep me moving forward.

TABLE OF CONTENTS

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

FIGURE

PAGE

LIST OF BOOK CHAPTERS, JOURNAL AND CONFERENCE PUBLICATIONS

Book Chapters:

- Mohammadreza Abtahi, Nicholas P. **Constant**, Joshua V. Gyllinsky, Brandon Paesang, Susan D'Andrea, Umer Akbar, and Kunal Mankodiya. WearUp: Wearable E-Textiles for Telemedicine Intervention of Movement Disorders. In the Elsevier Book of Wearable Technology for Medicine and Healthcare.
- Harish Dubey, Nicholas **Constant**, Mohammadreza Abtahi, Debanjan Borthakur, Qing Yang, Yan Sun, and Kunal Mankodiya, Fog Computing in Medical Internet-of-Things: Architecture, Implementation, and Application**.** Springer Handbook of Large-Scale Distributed Computing in Smart Healthcare.

Journal Publications:

- Amiri AM, Abtahi M, **Constant** N, Mankodiya K. Mobile Phonocardiogram Diagnosis in Newborns Using Support Vector Machine. vol. 5, no. 1, p. 20. Multidisciplinary Digital Publishing Institute, 2016. DOI:10.3390/healthcare5010016.
- Bahar Farahani, Farshad Firouzi, Victor Chang, Mustafa Badaroglu, Nicholas **Constant**, Kunal Mankodiya, Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare, Future Generation Computer Systems, Available online 24 May 2017, ISSN 0167-739X, https://doi.org/10.1016/j.future.2017.04.036. [Impact factor: 2.88]

Conference Proceedings

- Harish Dubey, Nick **Constant**, Kunal Mankodiya, "RESPIRE: A Spectral Kurtosis-based Method to Extract Respiration Rate from Wearable PPG Signals". In the Proceedings of the 2nd IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE), July 2017, Philadelphia, USA.

- Nick **Constant**, Kunal Mankodiya, "Fog-Assisted wIoT: A Smart Fog Gateway for End-to-End Analytics in Wearable Internet of Things". Sensors to Cloud Architectures Workshop 2017. The 23rd IEEE Symposium on High Performance Computer Architecture, Texas, Austin.
- Lauren Plant, Berly Noriega, Arjun Sonti, Nick **Constant**, Kunal Mankodiya. Smart E-Textile Gloves for Quantified Measurements in Movement Disorders. The 2016 IEEE MIT Undergraduate Research Technology Conference, Cambridge, MA, Nov 4-6, 2016.
- Harish Dubey, Jing Yang, Nick **Constant**, Amir Amiri, Kunal Mankodiya, Qing Yang. "Fog Data: Enhancing Telehealth Big Data Through Fog Computing." Proceedings of the 5th ASE International Conference on Big Data, 2015.
- Nick **Constant**, Orrett Douglas-Prawl, Smauel Johnson, Kunal Mankodiya. "Pulse-Glasses: An Unobtrusive, Wearable HR Monitor with Internet-of-Things Functionality." In Wearable and Implantable Body Sensor Networks (BSN), 2015 12th International Conference on, IEEE, 2015.

LIST OF DEFINITIONS

Term	Definition	Source
Actuator	“An actuator is a mechanical device for moving or controlling a mechanism or system. It takes energy, usually transported by air, electric current, or liquid, and converts that into some kind of motion.”	[Sclater et al. 2007]
Architecture	“The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution”.	[IEEE-1471-2000]
Authentification	Authentication is the process of verifying a user’s true identity. This may involve the use of one or more means of proof of identification, also known as factors, such as PIN codes and smart cards.	[Nexsus]
Authorization	Granting of rights, which includes the granting of access based on access rights.	[ISO 7498-2:1989]
Cloud	Or, "The Cloud," is generally used as shorthand for Cloud Computing. The name "Cloud" comes from the fluffy cloud typically used in Visio-style network diagrams to represent a connection to the Internet.	[IoT Guide]
Cloud Computing	A general term for the delivery of various hosted services over the Internet. The "as-a-Service" moniker is used for cloud services such as Software-as-a-Service, Platform-as-a-Service and Infrastructure-as-a-Service. The back-end for many IoT devices may be delivered via the Cloud.	[IoT Guide]
Edge computing	This concept places applications, data and processing at the logical extremes of a network rather than centralizing them. Placing data and data-intensive applications at the Edge reduces the volume and distance that data must be moved.	[IoT Guide]
Edison	Intel's development board, slightly larger than a standard SD-card, targeting the wearable vertical. Edison Developer Kits are embedded with Yocto and have development support for the Arduino IDE,	[IoT Guide]

	Eclipse, Intel XDK and Wolfram.	
Embedded System	A computer system dedicated to a specific task or tasks, operating within a larger mechanical or electrical system. Program instructions written for embedded systems are referred to as firmware, contrasting with the general purpose computer systems programmable for multiple tasks. Embedded systems often operate in isolation.	[IoT Guide]
Fog Computing	A term coined by Cisco, Fog Computing is an approach to networking backhaul that places a substantial portion of the data and processing at the edge of the network (on routers) rather than in the Cloud. Cisco associates the term IOx with the architecture that combines Linux with IOS. Fog Computing, which is meant to contrast with "Cloud Computing," is evocative of fog being close to the ground and is also known as Fog Networking and Fogging.	[IoT Guide]
Fog node	The physical and logical network element that receives feeds from IoT devices using any protocol, in real time; Runs IoT-enabled applications for real-time control and analytics, with millisecond response time; Provides transient storage, often 1–2 hours; Sends periodic data summaries to the cloud	[Cisco]
Industrial Internet of Things	A subdiscipline of IoT, encompassing IP-enabled systems such as factory-floor monitoring, HVAC, smart lighting and security. Also referred to as Industry 4.0 (Industrie 4.0) and Industrial IoT.	[IoT Guide]
Industrie 4.0	Invoking a fourth industrial revolution, Industrie 4.0 creates intelligent manufacturing networks where decentralized smart factories can communicate and react to each other autonomously. For example, in an Industrie 4.0 factory, self-predictive systems would trigger maintenance processes autonomously and automatically adapt logistics to the resulting changes in production. The term, also known as Industry 4.0, was first used at the Hannover Messe in 2011.	[IoT Guide]
IoT	A network of physical devices that are connected via	[IoT Guide]

	the Internet and that can communicate their status, respond to events or even act autonomously. Using sensors and actuators, the IoT connects digital devices and even everyday objects, thereby extending the Internet into the physical world. "Things" can include sensors recording the physiological measurements of humans.	
IoT Services and People	An extension if IoT that keeps people as the decision makers, who program and control the production processes and activities performed by Things. Used primarily in the IIoT context, the term was coined by SICS and ABB.	[IoT Guide]
Local Area Network (LAN)	A network of devices in relatively close proximity, prior to the point of transmission over leased telecommunication lines. The two most common communications technologies used in LANs are Ethernet and WiFi.	[IoT Guide]
Mesh Network	An ad-hoc network infrastructure where the nodes communicate directly with each other without the need to pass through a central structure such as an ISP. The only way to shut down a mesh network is to eliminate every node, and one of the most dramatic demonstrations of the technology was during the Hong Kong protests of October 2014 during which the direct communication between protesters' devices confounded the government's ability to block communication. The adaptivity of mesh networks makes them ideal for IoT applications.	[IoT Guide]
Middleware	Middleware is computer software that provides services to software applications beyond those available from the operating system. It can be described as "software glue". Middleware makes it easier for software developers to implement communication and input/output, so they can focus on the specific purpose of their application.	[Wikipedia, Middle 2017]
Multi-tenancy	The term "software multitenancy" refers to a software architecture in which a single instance of software runs on a server and serves multiple tenants. A tenant is a group of users who share a common access with	[Wikipedia, Multi 2017]

	specific privileges to the software instance. With a multitenant architecture, a software application is designed to provide every tenant a dedicated share of the instance - including its data, configuration, user management, tenant individual functionality and non-functional properties. Multitenancy contrasts with multi-instance architectures, where separate software instances operate on behalf of different tenants.	
Orchestration	A type of composition where one particular element is used by the composition to oversee and direct the other elements Note 1 to entry: The element that directs an orchestration is not part of the orchestration (Composition instance) itself. Note 2 to entry: See ISO/IEC 18384-3:2016, 8.3.	[ISO/IEC 18384-1:2016]
Reliability	Ability of a system or component to perform its required functions under stated conditions for a specified period of time.	[ISO/IEC 27040:2015]
Security	The correct term is 'information security' and typically information security comprises three component parts: - Confidentiality. Assurance that information is shared only among authorized persons or organizations. Breaches of confidentiality can occur when data is not handled in a manner appropriate to safeguard the confidentiality of the information concerned. Such disclosure can take place by word of mouth, by printing, copying, e-mailing or creating documents and other data etc.; - Integrity. Assurance that the information is authentic and complete. Ensuring that information can be relied upon to be sufficiently accurate for its purpose. The term 'integrity' is used frequently when considering information security as it represents one of the primary indicators of information security (or lack of it). The integrity of data is not only whether the data is 'correct', but whether it can be trusted and relied upon; - Availability. Assurance that the systems	[ISO/IEC 27001]

	responsible for delivering, storing and processing information are accessible when needed, by those who need them.	
Sensor	A device used to measure a specific characteristic of the surrounding environment, such as temperature. The use of sensors and actuators to connect Things to the physical world are a key component of IoT. A properly implemented sensor ideally should be sensitive only to the characteristic being measured, and should not interfere with what's being measured nor be influenced by other characteristics.	[IoT Guide]
Ubiquitous Computing	The concept of embedding microprocessors in everyday things so they can communicate information continuously. Ubiquitous devices are expected to be constantly connected. Utility smart meters are an example of ubiquitous computing, replacing manual meter-readers with devices that can report usage and modify power settings on ubiquitous appliances.	[IoT Guide]
WiFi	Wireless technology that lets a device connect to a network via an access point (AP). IEEE 802.11 is the generic name given to the WiFi family of wireless technologies. The 802.11b specification has been rendered obsolete by the newer Wi-Fi standards 802.11g, 802.11n. The 802.11p and 802.11ah standards are tailored for IoT applications.	[IoT Guide]
Wireless Sensor Network	Autonomous sensor nodes that are connected to one, or sometimes several, other sensors to cooperatively pass their data through wireless connections to a main location. Use cases include area monitoring, health care monitoring, environmental sensing, industrial monitoring and more. Modern WSN's can operate bi-directionally, enabling control of the nodes' activity.	[IoT Guide]

CHAPTER 1. Introduction

Definition: The Internet of Things (IoT):

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 thesis 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. This is an ad-hoc network infrastructure where the nodes communicate directly with each other without the need to pass through a central structure such as an ISP [IoT Guide].

Low -End Device Classification Levels		
Class 0	*Class 1*	*Class 2*
Constrained Resources	*Limited Resources*	*More Resources*
Less than 10kB RAM *Less than 100kB Flash*	*Approx. 10kB RAM* *Approx. 100kB Flash*	*Greater than 10kB RAM* *Greater than 100kB Flash*
Body Sensors	*Wireless Sensors*	*Gateway Devices*
PPG, IMU, PCG, Gemma	*nRF52XX, LilyPad*	*Intel Curie, Raspberry Pi*

Table 1: Low - End Device Classification Levels is a breakdown of which systems resources limit the practicality of using a device within the IoT.

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. 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 [Miorandi et al. 2012]. 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.

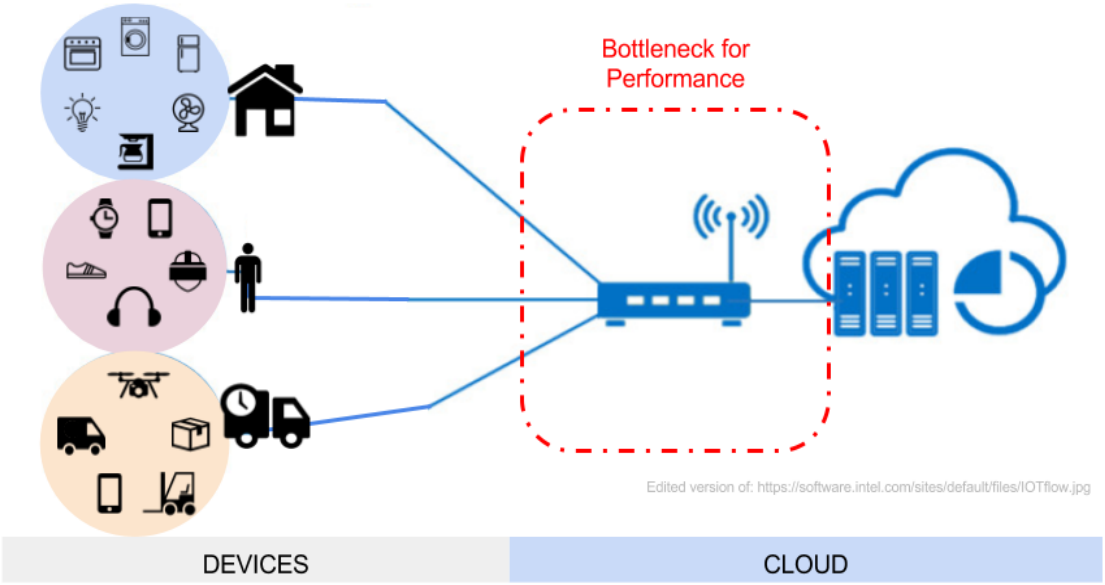


Figure 1: This shows the general layout between things (left) their gateway device (center) and the cloud (right). As new networks become available for the things, their bandwidth will increase, forcing pressure on the network between the gateway and cloud. This trend towards congestion continues as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

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. This device places applications, data and processing at the logical extremes of a network rather than centralizing them. Placing data and data-intensive applications at the edge reduces the volume and distance that data must be moved [Miorandi et al. 2012]. 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.

The role of 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 [Cisco, 2015]. 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. This continuously floods cloud data centers at a rapid rate creating a need for the cloud to move closer to the device. Therefore, the paradigm of fog computing is increasingly identified as a useful tool to establish remote intelligence in the context of IoT. In this thesis we discuss the recent development of a fog gateway and its role in orchestrating the process of *data acquisition*, *conditioning*, *analysis*, and *storage*. 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.

Defining an intelligent fog centric network for IoT-driven smart communities:

This master thesis research 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 master thesis. More emphasis was given to the fog computing and its performance analysis. This fog concept coined by CISCO aims to minimize latency,

conserve bandwidth, improve security, maintain reliability, and move data to the best place for processing [Cisco], thereby creating a greater reliance on the local network to function.

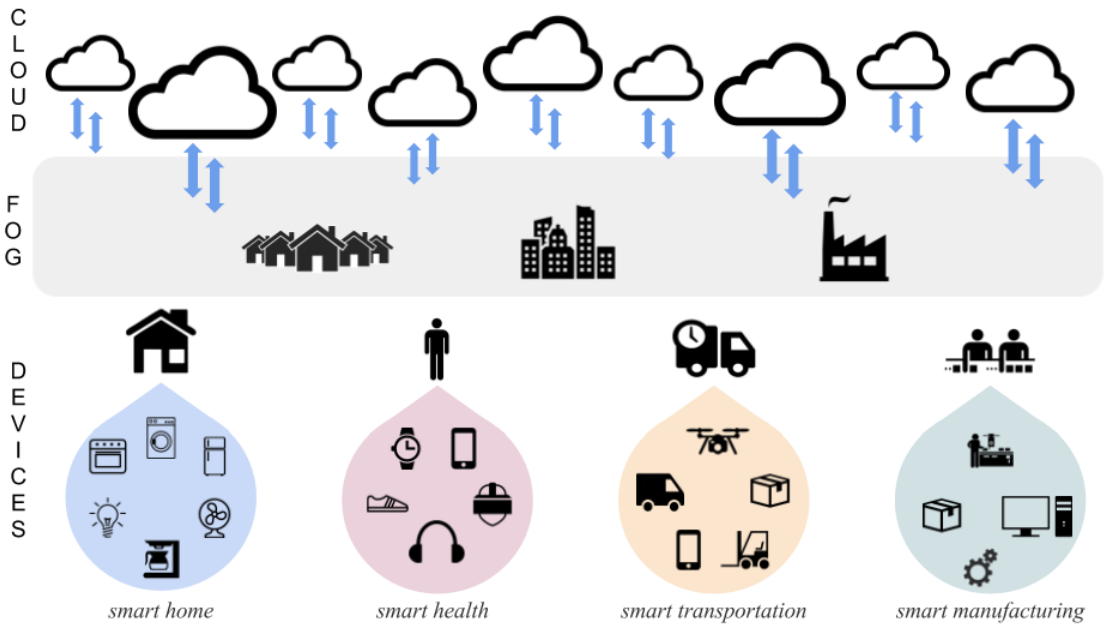


Figure 2: The thing to cloud continuum: Devices with limited functionality, computational power and resources appear at the bottom. These devices exist and interact with a small number of other "things", and typically in a usual context. Contexts such as home monitoring, personal health and entertainment, 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.

The performance of the described fog architecture is evaluated on several fog testbeds including the Intel Edison, Raspberry Pi and Intel NUC. They will run through an experiment use case for a telemedicine facility that could analyze the motion of the patients. The results demonstrate that the fog gateway provides a way to improve the interactions among the general purpose sensing devices, smartphones, and the cloud. In particular, the focus is to evaluate the timing, CPU load, memory load, and network measurements throughout each stage of the cloud-to-things continuum.

This thesis looks to provide 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.

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

The inclusion of the fog layer and its concept allows for more sensors, or edge devices, to interface with the cloud on a much larger scale than previously seen. Since the cloud is not setup for this volume and variety of data, changes in the systems closer to the edge than the cloud must be adapted to better utilize the cloud services available. That is why in this paper we will discuss the development of a smart fog to reduce the data sent

to the cloud by orchestrating the processes of data acquisition, conditioning, analysis, and short-term storage.

This thesis is structured by in the following manner:

- Chapter 2 will provide a detailed architecture of the fog environment by defining the roles of an IoT device, a fog node, and cloud. The chapter will explain what fog is and how it can be used, finally concluding with a thorough explanation of the fundamental framework composing the fog system.
- Chapter 3 will provide a description of the specific configurations used by the communication protocols, along with details for the overall fog system setup used in the experiment.
- Chapter 4 will run through the results of the experiment. The experiment will be discussed with a focus on each category of *data acquisition*, *conditioning, analysis* and *storage* as reducing the data transfer bottleneck. A comparison of the results for our experiment to those of realistic payloads will be used to explore limitations in scale. Thus leading into a breakdown of performance successes and concerns.
- Chapter 5 will summarize the system and address current and future research topics that will help with some uncovered performance concerns.

Detailed configuration files, data sheets showing hardware configurations, and pseudo-code for the experiments can be found in the Appendices.

CHAPTER 2. Background -Intelligent Fog Centric Networks

2.1 What is Fog Computing?

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 [Chiang et al. 2016]. 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.

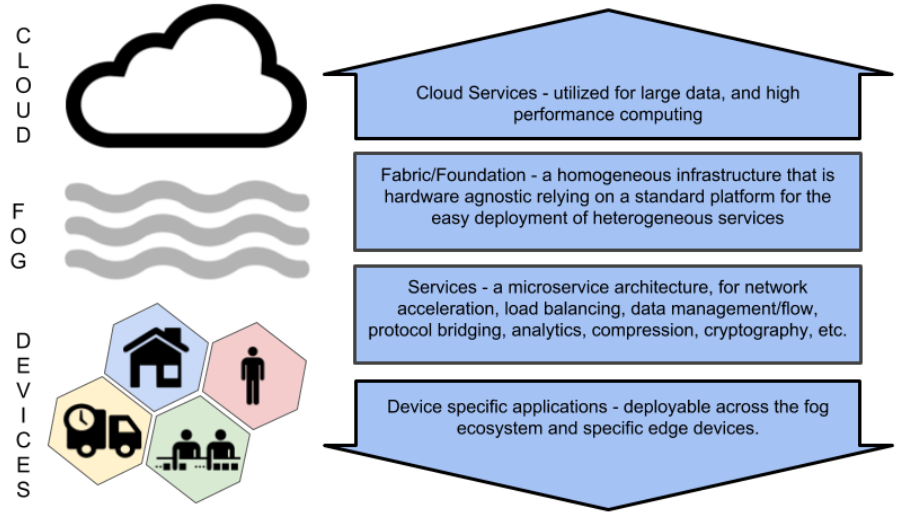


Figure 3: This is a general breakdown showing where services are performed in the path from device to cloud. Devices in IoT-driven communities with limited functionality, computational power and resources are specific in function. Moving through the continuum to the top we approach fog, which serves as the foundation for services to be incorporated between cloud and devices, but physically closer to the user than the cloud. 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. In its simplest form, the fog moves the cloud closer to the user.

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 [Bahar et al. 2017]. 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.

- 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.

2.2 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

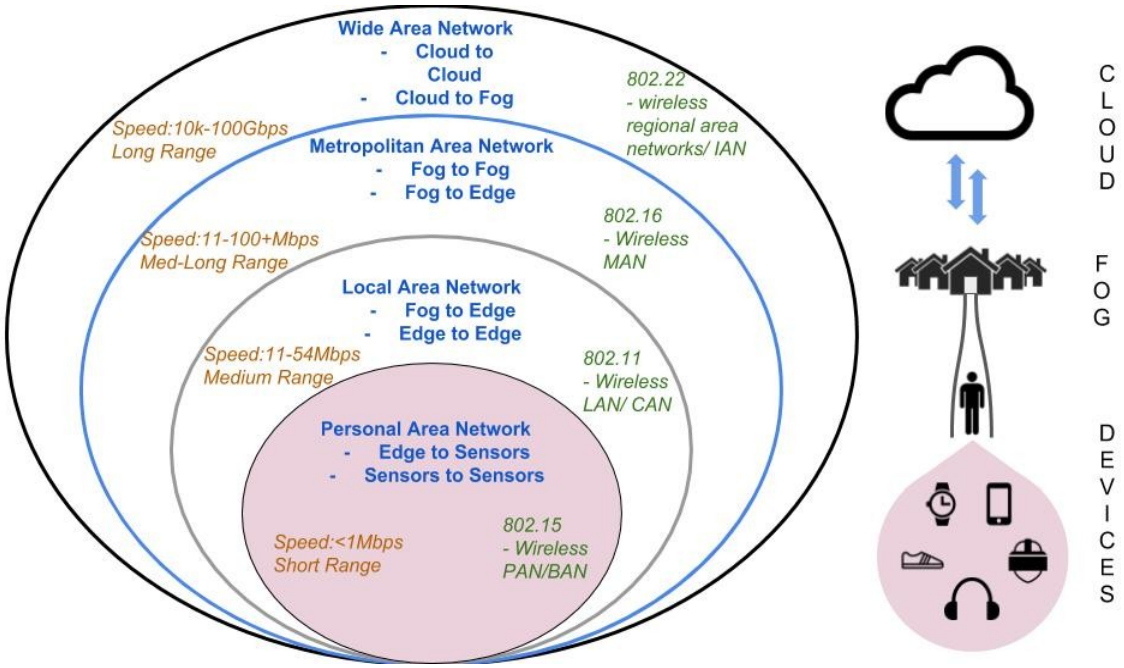


Figure 4: An outline of the types of networks that are most applicable for communication between the layers of IoT. The idea is to highlight the appropriate range and bandwidth limitations given the network communication focal point.

services, agriculture and general transportation [Guerra et al. 2017, Ha et al. 2014, Zao et al. 2014]. 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.

Case Study: Telemedicine and in-home technologies

WearUp is a study focusing on Parkinson's Disease (PD), its interventions, and existing wearable technologies for movement disorders. Along with the advancements of smart textiles, this study looked into the design aspects of the electronic textiles, wearable systems, and experiments for validating the involved technology. The WearUp project describes the algorithms required to process data recorded from sensors integrated into the WearUP glove, Figure 5 depicts this process. In the experiment, the WearUP glove was used to quantify the repeatability and accuracy of finger tapping in both time and amplitude. The setup realized that sensing, communicating, and processing in real-time requires orchestrating information between the glove and mobile device. Mobile devices such as smartphones or tablets are less constrained as far as battery life, storage, and processing capacity.

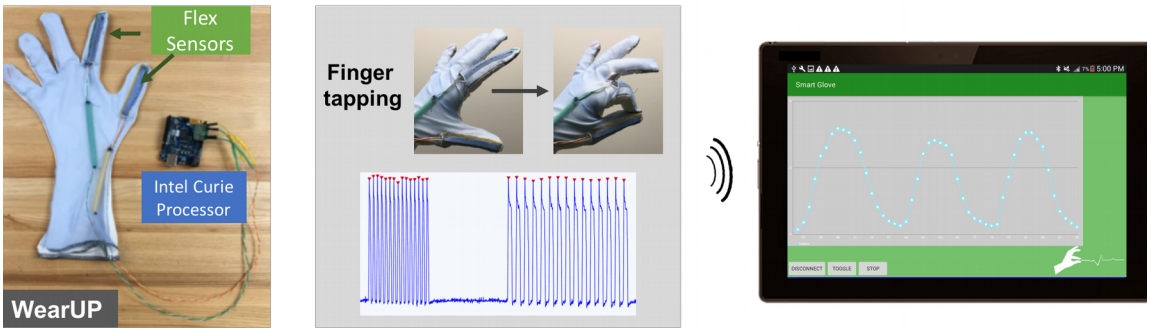


Figure 5: An overview of WearUP, a smart glove technology for Parkinson's Disease. The left side depicts, the glove to be worn during the exercise along its is the embedded Arduino 101 board responsible for streaming the raw data to the tablet. The middle shows finger tapping mid-exercise. The right is a real-time stream of processed data originating from the glove.

In the pilot study, nine healthy human participants were asked to perform a finger tapping task. The task involves tapping the pointer finger against the thumb 15 times, relatively quickly. Participants then rested their hands in the fully-opened position for five seconds. The participants were then asked repeat the finger tapping task at a slower speed. Data was collected from each participant for 3 rounds of this procedure. The

participants were instructed to determine their own speeds to evaluate the performance of the WearUp glove in detecting variations in finger tapping velocity. After human patients completed the exercise, a robotic hand repeated the procedure in order to calculate the standard deviation of error when known movements were measured by the WearUP glove.

At the conclusion of the experiment the system was able to consistently detect finger taps and calculate their frequency. The system can also assess the amplitude of the tapping motion. The experiment showed that the breaking down of required processing and delegating those processes to external devices is effective in providing an IoT Service for humans with near real-time feedback.

Example: Smart City in action

New York City has a couple example smart city initiatives [Guerra et al. 2017]. There are multiple projects, with a common general purpose or mission between the applications. The aim is to improve the efficiency of basic city operations to enable more civic services within existing budgetary constraints. While working towards this goal, it is important for the city to address concerns of bandwidth and connectivity issues that pop up during implementation stages. This is one reason why the push for 5G will be a great relief to many of these concerns [Spectrum].

Two eye catching projects in New York City are showing progressive initial results:

- The first is from the ***Accelerated Conservation and Efficiency (ACE) program***. They focus on smart indoor lighting to reduce greenhouse gas emissions in city agencies. It is especially useful for agencies which must operate 24/7, such as

firehouses. They successfully implemented this project and found a 3 million kilowatt-hour reduction in energy consumption. That is to say, it reduced gas emissions by roughly 520 metric tons of carbon dioxide annually. This shows how appropriate solutions utilizing new forms of communication to automate control of previously unwatched devices can reduce both the cost and environmental impact of these devices.

- ***LinkNYC*** is another relatively appreciated project. It is a free to use communication network replacing over 7,500 payphones with new Link structures. These structures provide free public WiFi, phone calls, devices charge sets, and a tablet for city information about services, maps and directions.

These two specific use cases are by no means exhaustive. However, these cases highlight some of the complexities and advantages of integrated fog computing. In keeping with the order, healthcare and activity tracking is another booming area of interest trying to integrate this technology most appropriately.

Example: Fog for Efficient Energy Consumption

The third case addresses utility services. Here, the EHOPES data-centered fog platform for smart living successfully reported more fine-grained energy consumption detail to users through their mobile devices [Li]. These reports help improve insights on energy generation along with identifying the primary culprits consuming the most power. It also provided an increase in the transparency of energy usage while decreasing the cost associated with the billing from both the provider and user. The system is a fog system, as indicated by its low latency, close proximity, real-time interaction, and availability to multiple subscribers. The server can also condition, analyze and temporarily store data before providing feedback to the user. Ideally these fog analytics are descriptive as in

“what”, diagnostic as in “why”, predictive as in “when”, and lastly prescriptive as in “how”.

Developing these insights can be seen in this collection of use cases. While they only address some areas of current research in the IoT community they highlighted the characteristics and purpose of fog devices. In particular, it showed how low-latency and location awareness enrich services found at the edge of a network. Such features are shown to be of importance in high bandwidth applications such as gaming, video streaming, augmented reality, and virtual reality. The widespread geographical distribution leads to appropriate deployments that can reliably provide streaming to moving devices like vehicles via proxies and access points, such as along highways and tracks, in sharp contrast to the centrally located cloud [Gubbi et al. 2013]. That mobility allows direct communication with mobile devices commonly found in the hands of many end users. Other characteristics such as the possibility of large scale, predominant wireless access, and streaming for real and near real-time applications are functional characteristics. Finally, a common characteristic of great importance in both the quality of data being collected and the security of the system itself is the ability to incorporate heterogeneous devices into the system based on trust.

2.3 The pillars of a fog system

Figure 6: This figure, adopted from the OpenFog Consortium depicts the essential elements of a successful fog architecture. Each pillar relies on the other seven to be fully functional. Depicted within the pillar-supported cloud are the services that can be best provided by the fog.[Image Credit: OpenFog Consortium]

This section discusses the required foundations for which a fog architecture can appropriately be built. While various researchers make progress on widespread possible future applications, the OpenFog consortium has been developing a common reference architecture. They focus on what they refer to as the "pillars of a fog system" while keeping the bigger picture in mind [OpenFog Architecture 2017]. The pillars are shown in Figure 6.

Together, the pillars support a system built for contextually aware client centered tasks. Also to assist other objects by enabling precise and learned autonomy. All of this is alongside the efficiency brought to fruition by the dynamic pooling of available resources within the users side of the architecture. Further creating the ability to scale for rapid

innovation leading into the reliable and real-time feedback and control. These advantages are termed CEAL, for cognition, efficiency, agility, and latency [OpenFog Consortium 2017].

Hierarchy

A well-structured but scalable hierarchy is required to maintain the flow of data and computational workload appropriately within the context of a given situation. The fog centric hierarchy is locally based, and tightly integrated with the devices in the user's environment. This type of hierarchy is not inconsistent with current consumer electronics, including personal home assistants and hubs. These personal home assistants, such as the Amazon Echo and GoogleHome have started to place fog nodes in user's homes. However, these devices do not yet deploy the appropriate hierarchy. Instead they rely heavily on data sent to and from the cloud for every request. Apple's new HomePod, however, differs from the other two in that it enacts a fog-appropriate hierarchy. In the HomePod system, user generated data is not sent to the cloud upon every inquiry. Instead, it is processed and understood from within the local network, as should be the case in a fog system.

Continuing down the road of common consumer electronics, it is worth looking at the hierarchy being enabled by the HomePod [Apple Keynote] as being most appropriate and the ones of the Amazon Echo and GoogleHome as one off. Regardless of the application, voice control for the system means that voice recognition must be reliable and readily accessible. As such accomplishing this fundamental task, should not rely on the cloud. This voice control allows many consumer devices to provide similar sensors and devices for us to use such as screens and speakers. Many are also are specific in

function while sharing communication capabilities like WiFi, Bluetooth, Zigbee, and Thread. They can therefore communicate and provide feedback through similar interfaces. This is part of the standard interface, where many devices located near the user can learn and inform at a local level. This is the groundwork for building cognition within the fog hierarchy.

Openness

Stepping where fog nodes, edge devices and the like have taken the time to develop standard interfaces for openness a participant using the WearUp glove -- or a smartwatch with a different telehealth application -- ready for their practice session could use their tv in place of their misplaced phone. In such an open setting the glove would not rely on the user to possess the phone to enable a practice session. Instead a voice command could start the home therapy practice relaying the phones prompts to be sent to a nearby edge device, such as a tablet, computer, TV, or even the fog node itself. The fog node would be able to communicate with devices nearest the user, like the tv screen in front of them, and use the nearby device to relay those session prompts and feedback from the misplaced phone to the user. Let us examine what exactly would be involved to create this open environment.

Devices in the IoT ecosystem are incredibly diverse in nature and functionality. This heterogenous population records an even more complex set of data. This data is not always, tied to the developers making the systems, with the likelihood of such a tie decreasing over time. Each device tries to extract useful information from the gathered data, but is often limited to its own knowledge base, making the analysis near sighted and narrow in comparison to the potential results when many knowledge bases are properly

integrated. This is beginning to be addressed for commercial users of cloud services from companies including Amazon, Google, and IBM.

These cloud services are powerful with the right data streams feeding into them. However, the current data stream being generated by the IoT devices is fairly unstructured. This unstructured data is given to the cloud with hopes that the cloud can make sense of it. This tendency causes a great deal of stress on the cloud, while the IoT devices expect reliable and timely responses to their queries. This is one area in which fogs openness can be useful.

The fog devices should be operable with many of these cloud services and bridge gaps between these services. This blanket coverage over various cloud platforms creates an area for fog services to be deployed. These services can then be used by multiple devices. This openness allows an edge device using a particular application to invoke a reaction in a separate device running a different application without needing to orchestrate the needed chain of events. That is because the orchestration would instead take place on the fog device.

This idea of being open and the interoperability of cloud services on a fog layer rests heavily on the development of horizontal services within middleware. This would incorporate services for network acceleration, content delivery, device management, video encoding, complex event processing, compression, crypto, and analytics platforms algorithms and libraries. Developers could then work on creating the fog as a service for business opportunities not yet available. However, this depends on the incorporation of the other pillars into the design.

Scalability

Scalability is a driving force behind the development of fog, and as being a product of necessity it is important to make it foundational to the concept itself. Scalability provides users with the ability to go from a small working deployment to a larger one with smooth accommodation in the network it relies on to function. This is common in the IoT economy with businesses employing pay as you grow strategies which could be excited by the predictive capabilities of this system. But this trend relies on the maintenance of performance, capacity, reliability, security, and software to handle the management of the any growth needed by the users.

One way to improve scalability involves virtualization and containerization. These allow for dynamically allocating resources to specific applications or services that require them, especially during bursty periods. This can lead the way to the prioritization of services in an elastic, demand-driven environment.

Programmability

Programmability of fog nodes enables highly adaptive setups. The standard interfaces allow the support for an adaptive infrastructure to change with the needs of software and hardware layers. This assists developers to maximize the resources available to them via virtualized and contained services creating efficient deployments. The programmability on both software and hardware layers gives a way to automatically apply patches at all levels in response to any new or imposed security threats. The essence of which is to provide standard and open interfaces at the general level for computational tasks and acceleration needs.

Reliability, Availability, Serviceability

The importance of reliability, availability and serviceability (RAS) applies to the areas of hardware, software and applications. That is, each area must be covered so that application uptime is consistently maintained.

Reliability means that the integrity of the data generation must be maintained by design. It must also be possible to autonomously meet the external computational requirements of the edge devices at any given time. The ideal is to initiate any requests for additional resources before they are demanded. Providing reliability means it is necessary to provide maintenance and diagnostic tools to the system as needed.

Availability means there is the ability to isolate faults through fault syndrome detection and machine learning. Availability is not measured in uptime as reliability is, but by the mean time to repair or MTTR. Availability requires redundancy in devices, mesh networks, secure remote access and boot, along with the ability to provide these features through backend support.

Serviceability means ensuring that the system as a whole is easy for users to configure, upgrade and repair. Providing autonomous servicing from the cloud backend by the manufactures themselves helps the deployment maintain a fluidity across the span of time, such that the fog system being used grows.

Autonomy

In fog computing, autonomy means enabling decision making at all levels of the hierarchy. This includes autonomous device discovery and management, with a secure way to come online to perform requested operations. This allows devices to be added to the system checking in with the cloud for authentication, authorization, or management.

The fog system can then function properly even when its network connection to the cloud is unavailable. When the network is available, the fog can serve as a proxy for the device looking to connect. As a concrete example, my request to preheat the oven for cookies shouldn't require my voice to travel to the cloud. My fog device should have the ability to autonomously handle such common queries. Allowing this type of decision making can provide more contextual awareness to the edge network. If the cloud is not available the data collected does not become information for knowledge to progress into intelligence.

Agility

When agility is paired with autonomy you can provide insights. Contextually based well aggregated and analyzed insights are then sent to the cloud for more in depth intelligence. This reduces the latency induced by transmission of vast amounts of unstructured data to the cloud which would otherwise be required. The idea is to move context creation as close to the data generation as possible without imposing too much strain on the devices doing the collecting. As a result, more fine tuned strategic decisions on system-wide and policy management can be maintained within the layers of fog.

Security

The security pillar is discussed last because a full understanding of security's importance depends on a foundational understanding of the other pillars. Fog devices serve in some sense as the gatekeepers to the larger network of cloud services, making their security essential. In scenarios where IoT devices have constrained resources, it is up to the fog to maintain a secure and safe connection from device to device and to the cloud. In the event of a security breach the should help to maintain localized infiltration, not allowing the wide area network to spread the breach. However, heterogeneous

devices and needs mean that a single security solution can not fit all use cases. However, a few security protocols are selected in this thesis to comply with the proposed architecture.

Fog Gateway Protocols

The fog and IoT devices exist in a network that is meant to be safe without malicious users. However, for the sake of caution and security, the network should be viewed as hostile in some senses. That is the main reason behind ensuring that security, like all other systems in the ecosystem, exists in every layer. Therefore, virtual private networks should be used to shield the fog system from outside and there must be internal encrypted communication.

Methods for encryption can be homebrewed for secrecy. However, when generally accepted practices are used, needed patches can essentially be crowdsourced, which is an advantage. For this reason, the connection and socket for transferring data from fog and edge devices is secured through TCP (for reliability) wrapped in Secure Socket Layer/ Transmission Layer Security (SSL/TLS). Secured sockets provide a secure communication framework for devices using different protocols such as TCP and UDP.

Transmission Control Protocol (TCP) is a networking protocol that allows for guaranteed and reliable delivery of files. It is a connection-oriented and bidirectional protocol. In other words, both devices can send and receive files using this protocol. Each point of the connection requires knowledge of the others Internet Protocol (IP) address and a port number to make a connection with a specific device. Furthermore, we wrapped the TCP sockets in SSL Sockets to ensure the security and privacy of data collected from the users/patients.

Secure Sockets Layer (SSL) is a network communication protocol that allows encrypted authentication for network sockets from the server and client sides. To implement it in the proposed Fog architecture, we used two python modules, namely SSL and socket. To create the certifications for the server and client, we also used the command line program called OpenSSL [OpenSSL 2015]. OpenSSL is an open-source project that provides a robust, commercial-grade, and full-featured toolset for the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols.

Once all the SSL certification keys are built, the client (things and edge devices) can use secure sockets on the server and continuously listen for data transfer connections to be made. The exact implementation used for these fog devices are explained in the methodology chapter.

CHAPTER 3. Methodology - Fog Computing Design, Development, and Evaluation

The methodology for this thesis 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 chapter 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.

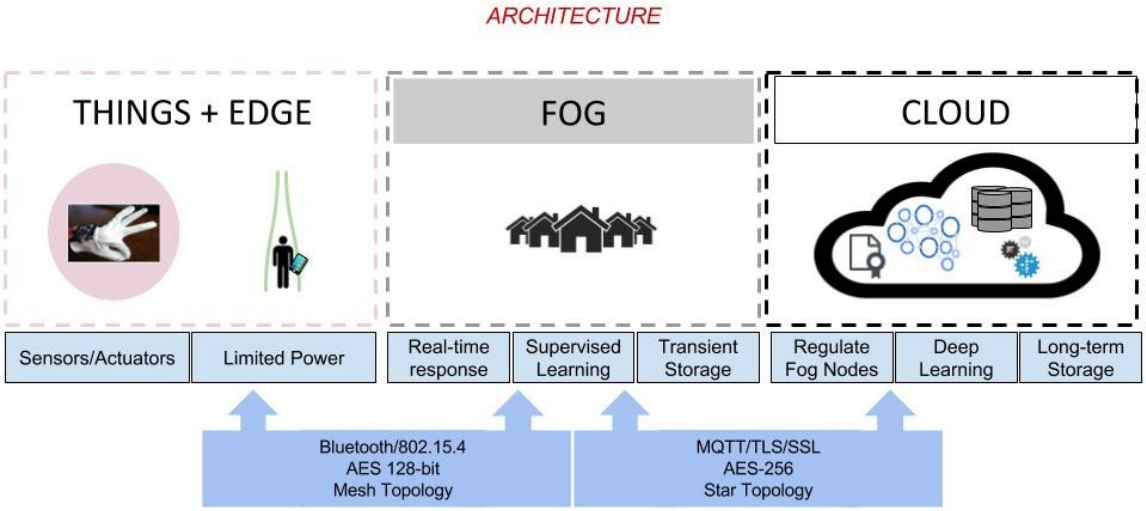


Figure 7: Each of the layers shown have expected applications and system constraints. IoT Devices being the most limited can communicate at

3.1 The 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 WearUP, an electronic textile smart glove system designed for PD. As mentioned earlier, we have carefully designed experiments on WearUP to test its repeatability, precision,

and sensitivity in quantifying the motor symptoms present in the finger tapping task. WearUP was tested using a robotic hand programmed to make precise finger tapping movements.

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. This requires knowing the relative spatial orientation of the fingers and palm. 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 WearUp. The flex sensors used are in essence variable resistors using conductive ink that increases in resistance as their surface area increases. These are placed between two flexible conductive traces housed in a clear insulative material. The resistance when flat is roughly 20kΩ and when bent to a 90° angle increases to roughly 70kΩ.

Measuring this physical state requires a static state for comparison. We create this static state by introducing a standard resistor. To measure the relative change in state between the standard resistor and flex sensor, we design a simple voltage divider. This allows us to extrapolate the desired information using Ohm's Law. A final consideration for this set up is to maximize the voltage range to increase the signal to noise ratio. We chose to use a standard 10kΩ resistor, which provides a potential voltage swing of

approximately 1v. Later, we mapped this range from 0 to 1024 as the flex sensors resistance reaches its maximum at a 90° angle.

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 accelerometer uses Newton's first law "An object at rest will remain at rest unless acted on by an unbalanced force. An object in motion continues in motion with the same speed and in the same direction unless acted upon by an unbalanced force". This is also known as the law of inertia. A thought experiment of a ball resting on a plate can help illustrate the physics of the accelerometer.

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.

The MCU must be small enough to fit on the back of a hand, consume very little

power such that the battery life of the glove extends beyond one practice session, and have the processing power to compute the simple mapping of the analog to angular values at close to real-time speeds. Currently, the MCU being used meets those requirements and implements extra high-frequency filtering to reduce false readings.

Edge device - Intel Curie-Based, Low-Power Embedded System:

The edge device of interest is a microcontroller unit (MCU) used in a pair of smart gloves for the WearUp application. It must be small enough to fit on the back of the hand, consume minimal power so the glove's battery life extends beyond one practice session, and have the processing power to map the analog values to angular values in near real time. Arduino 101, the currently implemented MCU, meets these requirements. The Arduino 101 is a learning and development board which contains an Intel Curie Module. This board is designed to integrate the Curie's low power-consumption and high performance with the Arduino's ease-of-use. The Arduino 101 also provides 19 channel 12-bit ADCs, Bluetooth Low Energy capabilities, and power management circuitry to ensure stability in reference to analog readings.

Once the glove is paired with a Bluetooth device, it begins to capture angular values from the finger flex sensors and store them as unsigned 8 bit integers in a first in, first out (FIFO) array. Once this array is filled, it is sent over an available BLE connection using the standardized GATT UART Service. This service provides the name, sensor location, and control points of the detecting device. An instance of the GATT UART Service can be updated by calling the object's notification function. This function is defined in the GATT UART Service header. Updating objects through functions is the standard method in BLE libraries. Once the object is updated, changes will be transmitted

to the paired device. The Smart Glove is always updating its local angular finger values with regard to a connected device. It is handled by an interrupt since requiring it to continue reading the values of the flex sensor without a paired device would waste power sampling unnecessarily.

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 WearUp smartphone application. WearUp is designed to run on any Bluetooth 4.0 enabled device running Android 4.4 or higher.

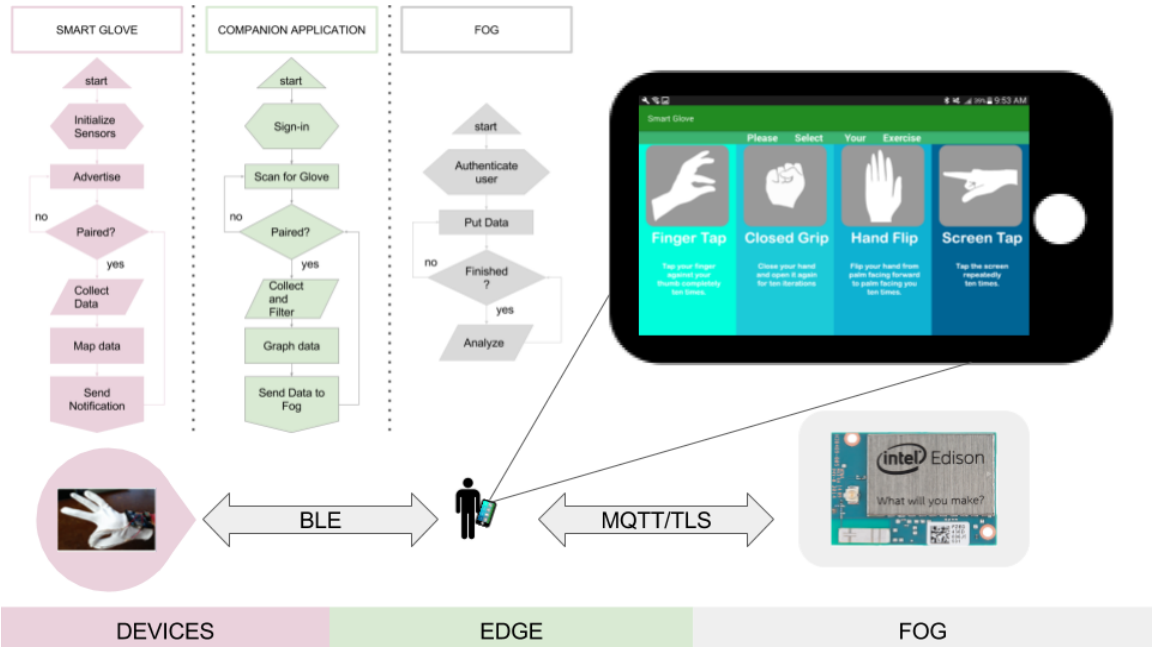


Figure8: This figure depicts the framework deployed for the WearUp application. The glove streams its data to the users edge device. This edge device prompts the user on the correct gestures to make. The edge device interacts with the fog for user authentication/authorization and feature extraction.

The mobile Android companion application serves as a bridge between WearUp's hardware and the user. The Android device running the WearUp application utilizes the Bluetooth Low Energy (BLE) technology available on both the Arduino 101 and the device itself to receive updates from the glove every 100 milliseconds. This update comes in the form of a byte array containing essential information from the Arduino 101, including raw sensor data from all enabled sensors. The mobile application initially scans for available BLE devices. Once BLE devices are discovered the mobile application must filter the discovered devices to consider only Smart Glove hardware and ultimately pair (bond) with the Smart Glove. After the device has paired with the mobile application, data transfer can be initiated over the BLE protocol. After data is acquired the application can then upload said data to a remote server for persistence, redundancy, and future access. The application should also display information to the user and physician. The current mobile application is able to scan for BLE devices, display the available devices along with MAC address, device name, local device name, signal strength, and service ID, connect with the Smart Glove, transfer patient data information, and display the fingers angular information to the user via a dynamic graph.

Visual feedback such as graphs are important for applications. A good user interface (UI) will keep a user engaged while communicating appropriately with users. A flat, layered, UI design was chosen for the WearUp companion application to coincide with the design language of the Android OS. The dynamic graph section of the application shows the user's finger movement over time in a visually appealing way. After cleaning and displaying the feeds, the Android device then parses this data and sends it to another fog edge node for further processing.

The Intel Edison platform used in this application is also part of the edge. It was designed with a dual-core, dual-threaded Intel Atom CPU at 500MHz and a 32-bit Intel Quark microcontroller at 100MHz. It has connectivity interfaces capable of Bluetooth 4.0 and dual band IEEE 802.11a/b/g/n via an on-board chip antenna.

This platform came with a Linux environment called Yocto. Yocto is not an embedded distribution of Linux, but rather provides an environment to develop a custom Linux distribution. We did not create a Linux distribution, instead deploying a prebuilt distribution of Debian/Jessie for 32-bit systems. We did this so we could deploy the same environment on the Intel Edison and the Raspberry Pi.

The Raspberry Pi Model B platform used in this application has a 900MHz 32-bit quad-core ARM Cortex-A7 CPU, and 1GB RAM. Since the Raspberry Pi does not have built-in WIFI connectivity, a WIFI dongle based on the Real-tek RTL8188CUS chipset was installed. This platform came with a custom Linux distribution called Raspbian. In order to work in a consistent environment, Raspbian was replaced with the Debian/Jessie distribution used on the Intel Edison.

Backend Cloud Network

In a cloud computing component, the server must be able to authenticate multiple users and create sessions for each user. Once a user is authenticated, the mobile application uploads data based on the user ID. The cloud computing component of this project can be expanded to many more platforms including iOS and web. To support the centralized storage of clinical features and analytics, we implemented a backend cloud database using PHP and MySQL. We run a LAMP server, specifically a Linux (Ubuntu 16.04), Apache2, MySQLi, PHP5.4 server. This open source solution uses Linux as the

core operating system, Apache for web servicing, MySQL as database system for management and storage, and PHP for server interaction with applications [Sobell 2013]. The main component of the backend was the relational database development. We designed the database to be easy for the users and fog computers to interact with the database. Three tables were created for the users (patients and clinicians). A fourth table contains information extracted from patient data. Extracted features were obtained from the Fog computer and entered in the data table [Bahar et al. 2017].

3.2 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.

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 9 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.

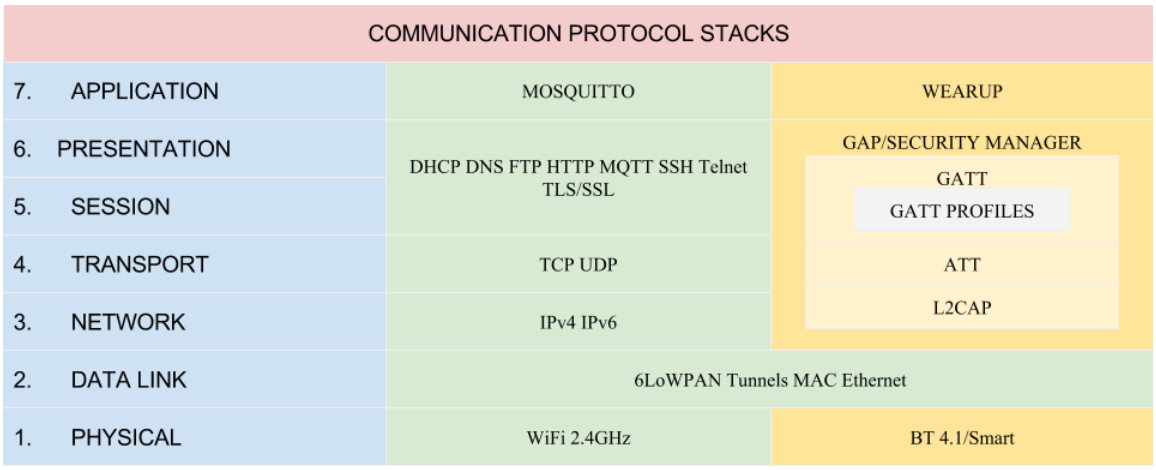


Figure 9: OSI Layering for the communication protocol stacks implemented for both Bluetooth and WiFi communication strategies.

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 thesis, 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 thesis 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 thesis but could be considered for a commercial deployment.

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.

3.3 A testbed for evaluation

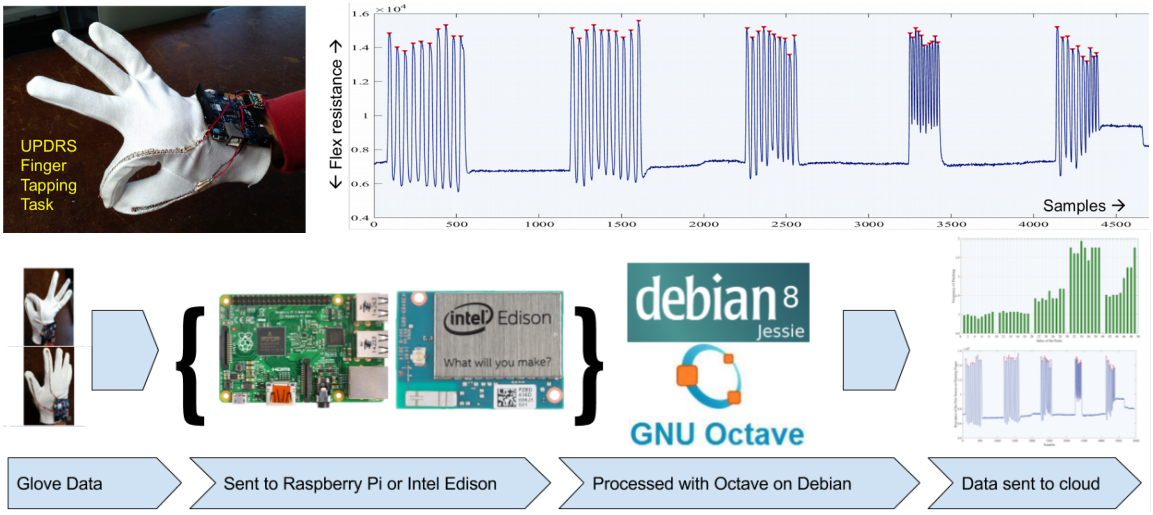


Figure 10: Shows the WearUp prototype glove being worn while the wearer performs a finger tapping exercise. To the right is a sample data plot from that same experiment overlaid with markers at each peak as determined by the fog device.

WearUp with Fog-Driven IoT

The testbed was set up for the WearUp case study. The smart glove, with flex sensors sewn onto the pointer finger and thumb to detect motion features such as tremors, rigidity, and slowness, is shown above. This version uses an Arduino 101 with a Intel Curie Chip. Voltages across the flex sensors are mapped to a range from 0 to 1024. The adopted finger tapping motor task is part of the Unified Parkinson's Disease Rating Scale (UPDRS) [Geotz et al 2015]. Neurologists observe the frequency of tapping as a key feature in the test. They also consider variations in tapping speed and record a clinical

score for this task. We adopted the finger tapping test for its high clinical value and its ease of home performance.

The task requires the patients to tap pointing finger and thumb together 10 times. We had healthy participant perform the finger tapping test five times. The participant was asked to vary the rate of finger tapping frequency in each round. A peak detection algorithm was deployed to measure the tapping's intensity. The basic algorithm checks if the amplitude of the signal is greater than a threshold and greater than nearby samples. Such samples are considered peaks. Since each pinch includes micro movements or small fluctuations which can result in duplicate peaks for a single tap, a temporal threshold is also implemented. In this case, peaks less than ⅓ of a second apart are presumed to come from the same pinching movement. Finally, the temporal separation of peaks is calculated and converted to a frequency.

The experiment was conducted using Intel Edison and Raspberry Pi (version 3), and an Intel NUC. The Intel Edison (to be discontinued this year) used the same i386 Debian/Jessie distribution as the Raspberry Pi 3. However, the Intel NUC ran a full-fledged Ubuntu 16.04 distribution. All three of these devices serve as embedded fog computers, giving us the ability to run Octave 3.8.2-4 and Python for data processing. However, only the Intel NUC hosts a PostgreSQL database for storing data. The Intel Edison and Raspberry Pi only stored data during processing.

The Smart Glove must send sample sets with the collected data every 300ms. The embedded fog computers must act as gateways to the cloud as well as platforms for data processing in between sample sets. The Edison platform has a core system consisting of dual-core, dual-threaded Intel Atom CPU at 500MHz and a 32-bit Intel Quark

microcontroller at 100MHz, along with connectivity interfaces capable of Bluetooth 4.0 and dual-band IEEE 802.11a/b/g/n via an onboard chip antenna. The Raspberry Pi platform has a 900MHz 32-bit quad-core ARM Cortex-A7 CPU, and 1GB RAM. The Intel NUC uses an Intel Core i5-5250U Processor with 3M Cache up to 2.70 GHz and 16GB DDR3L-1333/1600 1.35V SO-DIMM RAM with a maximum bandwidth of 25.6 GB/s. Both the Raspberry Pi and Intel NUC use wired ethernet connections for the best possible connectivity.

CHAPTER 4. Findings

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" [collectd].

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

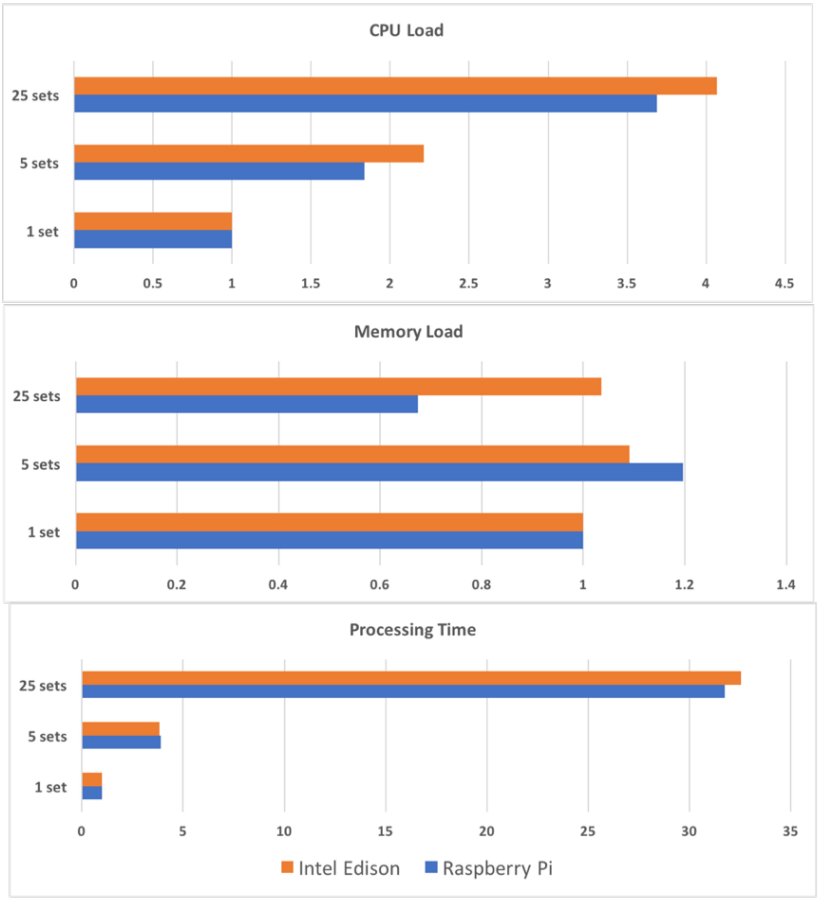
point expressed in seconds. The difference between the toc and the tic is stored as processing time. *Latency* was recorded by measuring the response time of a "ping" payload. One fog node would post the "ping" message in the same queue as the data stream. A toc was called when the "ping" was returned. Again, the difference between a toc and a tic was recorded.

4.1 The resulting performance

- *Intel Edison and Raspberry Pi*

  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.

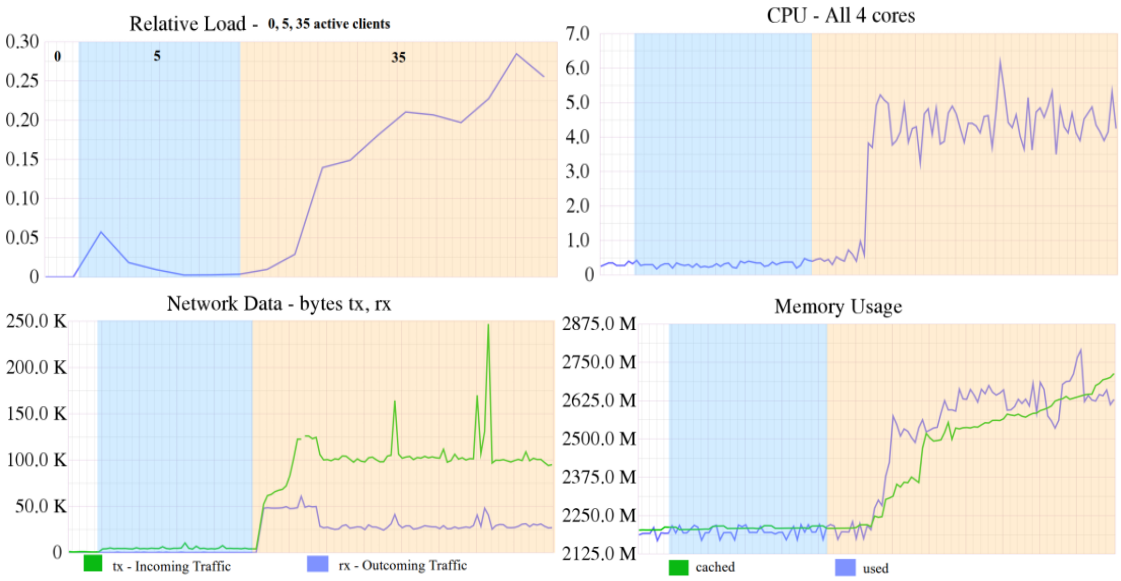


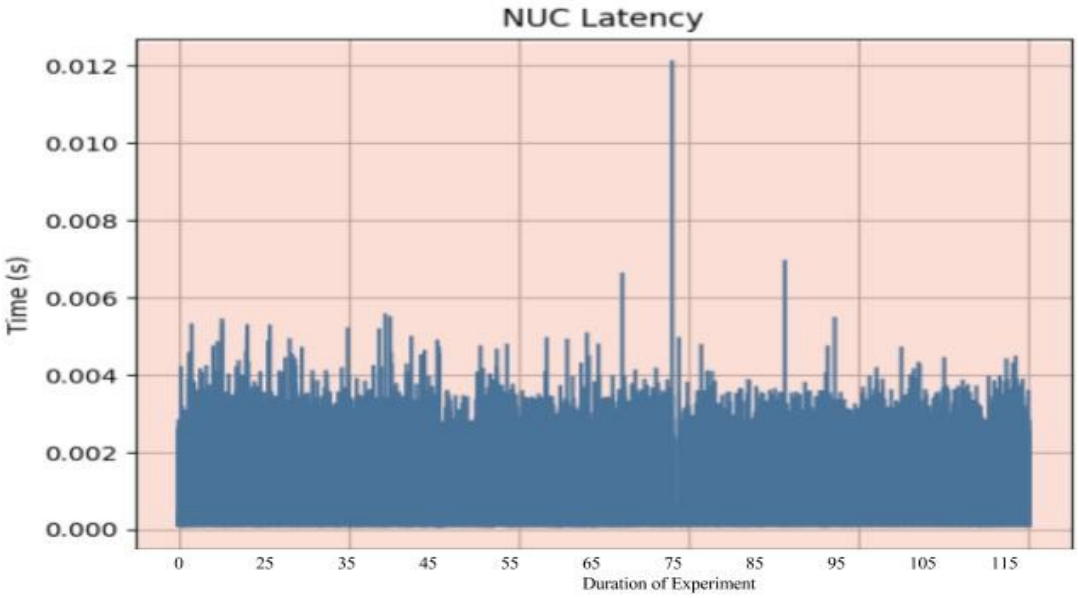
	Pi	Eddy
1 Set		
Timing Breakdown		
Load	0.1784	0.3156
Process	0.0694	0.1574
Plot	2.2856	3.4851
total	2.5334	3.9581
Total Process Breakdown		
Average CPU LOAD	13.117%	19.575%
Average MEM Load	4.200%	2.454%
Total Runtime	00:11.1	00:31.6
5 Sets		
Timing Breakdown		
Load	0.8928	1.588
Process	0.3008	0.7221
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

Figure 11: The above graphs show 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 [Bahar et al. 2017].

The created network places the fog node between the WearUp Glove and the cloud. It is assumed that the arrival and service processes are geometrically distributed. Since the fog node will be placed in locations with only a small number of devices, such as nursing homes, it is assumed that the mean arrival rate for data will be once per minute. The service rate for the fog nodes can be surmised from Fig. 11 under total runtime.

- *Intel NUC*


Relative Load - 0, 5, 35 active clients
0.30
0.25
0.20
0.15
0.10
0.05
0
0
5
35
CPU - All 4 cores
7.0
6.0
5.0
4.0
3.0
2.0
1.0
0
Network Data - bytes tx, rx
250.0 K
200.0 K
150.0 K
100.0 K
50.0 K
0
tx - Incoming Traffic
rx - Outcoming Traffic
Memory Usage
2875.0 M
2750.0 M
2625.0 M
2500.0 M
2375.0 M
2250.0 M
2125.0 M
cached
used


NUC Latency
Time (s)
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0
25
35
45
55
65
75
85
95
105
115
Duration of Experiment

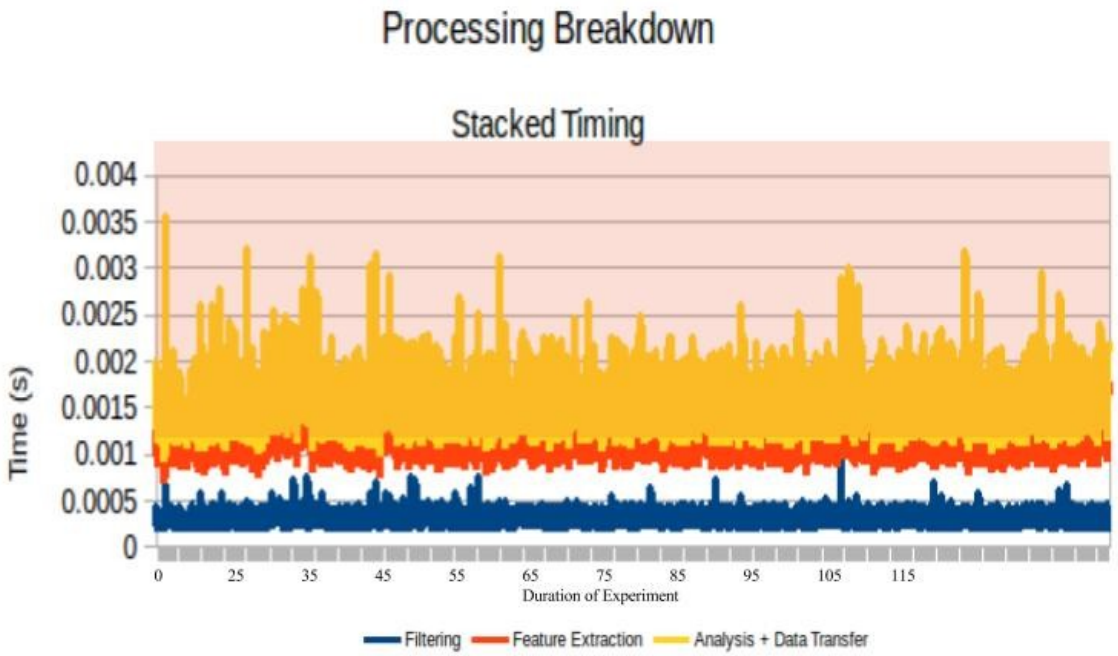


Figure 12: Above is the performance for the Intel NUC. The NUC was first set up to broker 5 publishers, along with 5 subscribers each interested in singular topics. The increase in all parameters were visually almost indistinguishable from an idle state. Once this setup was multiplied by 7x, the load grew by ~5x. However, it maintained acceptable processing times, brokered connections and reduced outgoing traffic with feature extraction.

Using Littles Law we can determine the average wait time for each device. When running one set at a time, we found that the average wait time for the Edison and Raspberry Pi will be roughly 64.65 seconds and 12.39 seconds, respectively. These fog nodes are used to collect data from the WearUp glove, process the data, and post the processed data to a server in the cloud. Fig. 17 shows a breakdown in the average amount of time spent on each step. The Raspberry Pi provided a service rate one third that of the Edison. Furthermore, in active mode, the Raspberry Pi consumes 198mW/s and the Edison consumes 529mW/s (see Fig. 11).

The Intel NUC far exceeds the capabilities of the Intel Edison and Raspberry Pi. This result is to be expected as the NUC is designed for media streaming, while the Edison and Pi were designed for low power collection, process, and transmit functions. Unlike the others, the NUC encrypted and brokered the transactions between 35

interested nodes (referred to as publishers/subscribers). This mix showed the same diversity in parties such that there were subscribers and publishes at layers above and below the fog.

4.2 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.

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.

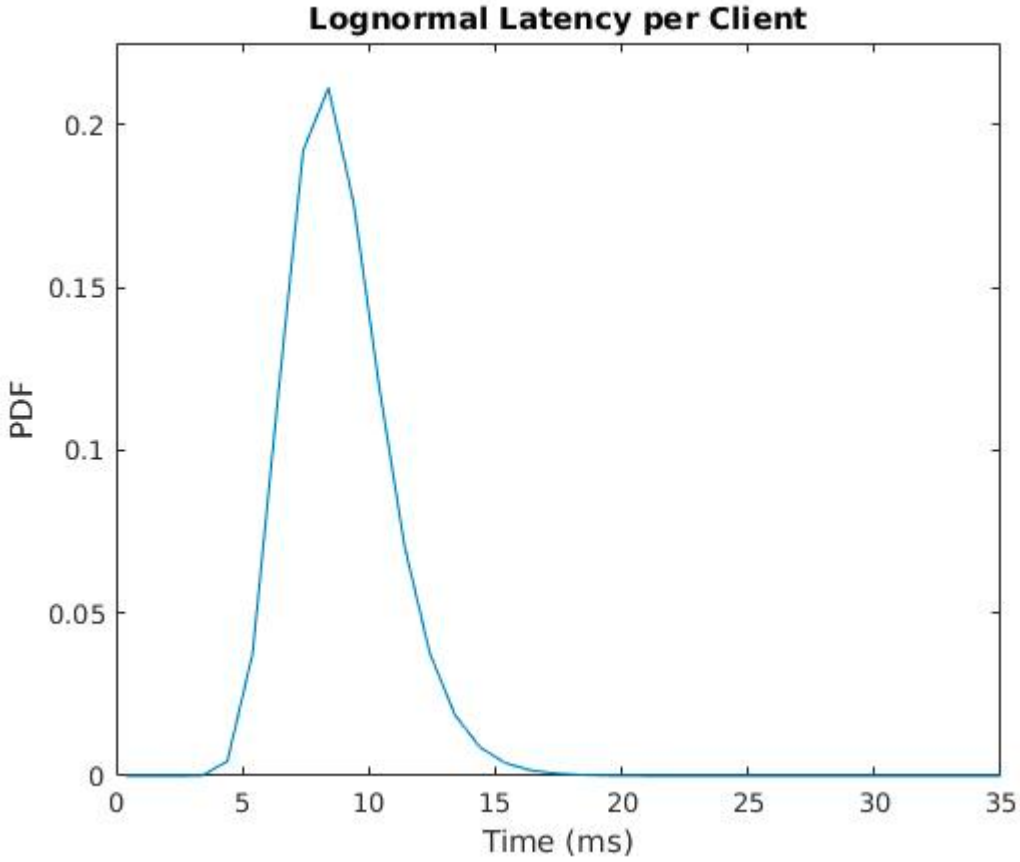


Figure 13: Above is the latency of the fog for the Intel NUC. The latency was fit to a Log-Normal Distribution where the mean was 8.511ms with the 95% confidence interval of 8.251 and 8.771. This pdf was consistent was each client. The test ran with a maximum of 35 clients connected simultaneously.

It should also be noted that both the Intel Edison and Raspberry Pi could wrap all the data in transport layer security without adding a significant time increase in latency between receiving client data and successfully sending it back to the client.

The third testbed, and only one that could be a workable fog device, is the Intel NUC. This device was capable of handling the same tasks given to the Intel Edison and Raspberry Pi while not exceeding 10% of its available resources. On a separate test, not

conducted on either the Intel Edison nor Raspberry Pi, required the Intel NUC to evaluate training sets for machine learning whilst maintaining the reliable streams to each client. The NUC successfully did this but required 80% of the available resources. The details of the machine learning program used can be found in the appendix and a full discussion on this is beyond the scope of this thesis.

Ultimately, no single operating system best works across all three of these devices. The closest to working is CentOS 7, but since it is still in its infancy for the Raspberry Pi, many bugs do still exist [CentOS Bug]. The bugs found on the Intel Edison are less supported and will only decrease in support as the device undergoes discontinuation in December 2017. However, even with these differences in operating systems, it was possible to deploy the same algorithms at the python package level of abstraction.

CHAPTER 5. Conclusion

Throughout this thesis 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.

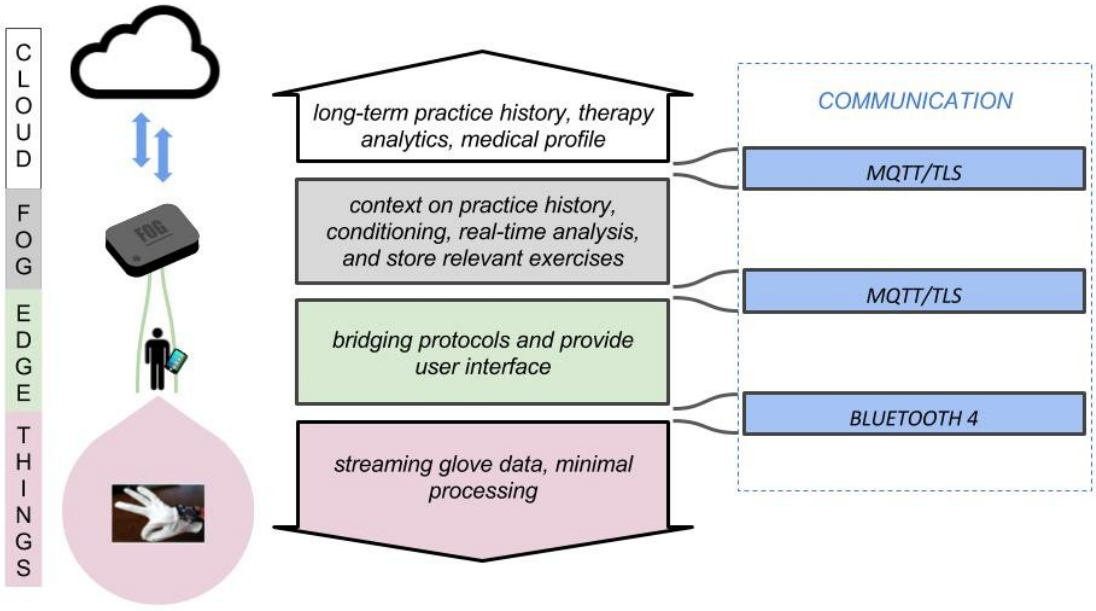


Figure 14: The most effective layering scheme deployed during the experiment.

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.

5.1 Additional opportunities

As the need for continuous connectivity increases and the reliance on high bandwidth applications follows, the current generation of cellular protocols must be improved. This type of improvement is expected in 5G networks, but is not yet ready. As a reduction in bandwidth is a current bottleneck in 4G communications, there are opportunities to increase 4G capabilities by deploying the discussed fog network.

5.2 Future research opportunities

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.

APPENDICES

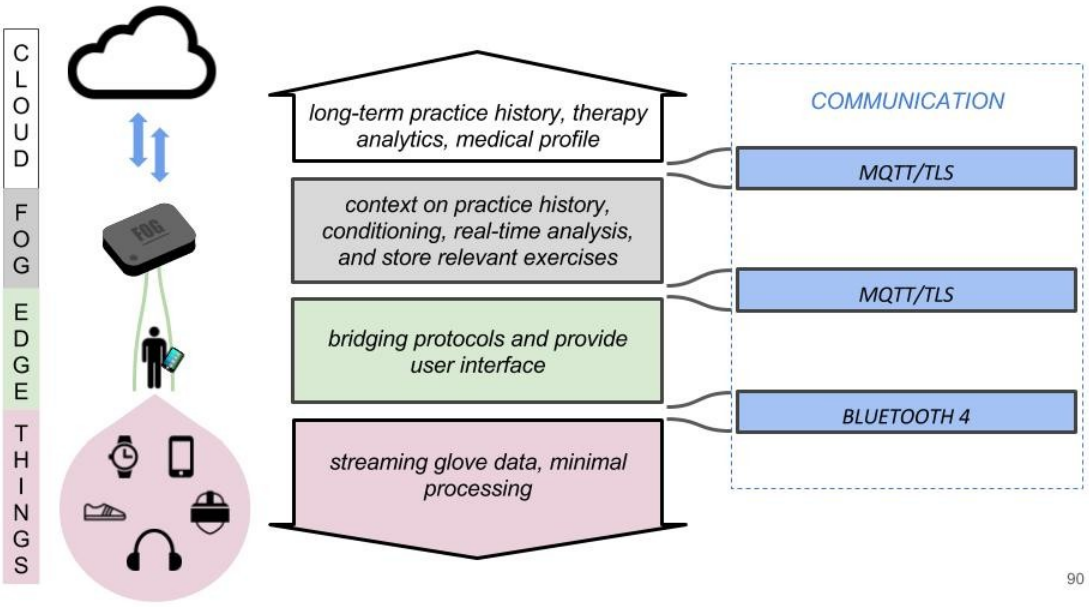
Stacks Implemented on device

COMMUNICATION PROTOCOL STACKS				
7. APPLICATION	MOSQUITTO		WEARUP	
6. PRESENTATION	DHCP DNS FTP HTTP MQTT SSH Telnet TLS/SSL		GAP/SECURITY MANAGER GATT PROFILES GATT	
5. SESSION				
4. TRANSPORT	TCP UDP		ATT	
3. NETWORK	IPv4 6LoWPAN Tunnels MAC		L2CAP	
	IPv6			
2. DATA LINK	Ethernet WiFi		LE LINK LAYER	
1. PHYSICAL			BT 4.1/Smart	
PLATFORM ABSTRACTION				
CHIPSET	BCM	QSD	NRF	CURIE
OS	LINUX	ANDROID	MBED	ARDUINO
PROCESSOR	INTEL		ARM	

121

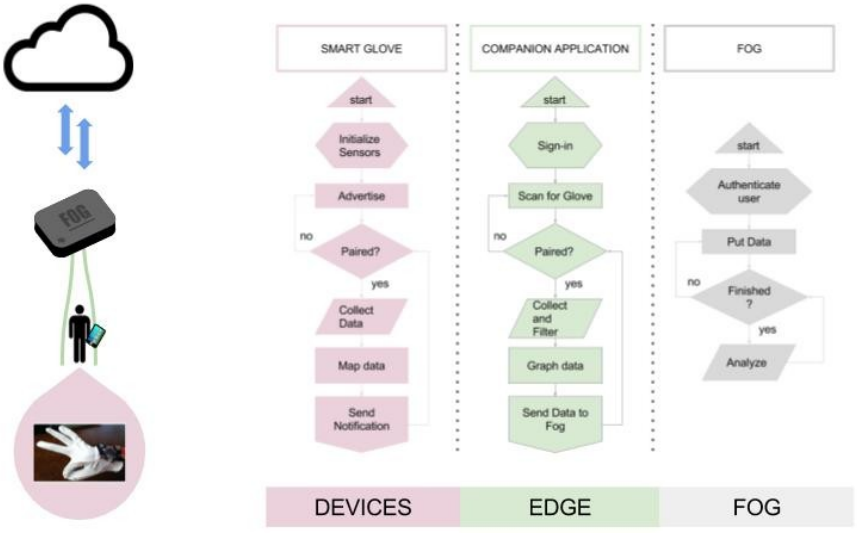
SoC	BCM2837	22nm Intel SoC	Intel Dual Core SoC
CPU	Quad Cortex A53 @ 1.2GHZ	Dual Core Atom-32 @ 500 MHz and 32-bit Intel Quark @ 100 MHz	Intel Core i5-5300U dual-core processor
Instruction Set	ARM v8-A	x86	x86_64
GPU	400MHz VideoCore IV	none	Intel® HD Graphics 5500
RAM	1GB SDRAM	1GB LPDDR3	16GB 1.35V DDR3L SO-DIMMs
Storage	micro-SD	4 GB EMMC	500GB SSD
Ethernet	10/100Mbps	10/100Mbps	10/100/1000Mbps
Wireless	802.11n;Bluetooth 4	802.11a/b/g/n; Bluetooth 4	802.11a/b/g/n/ac; Bluetooth 4

123



90

Case Study: WearUP



89

BIBLIOGRAPHY

1. Abu-Elkheir, Mervat, Mohammad Hayajneh, and Najah Abu Ali. "Data management for the internet of things: Design primitives and solution." Sensors 13.11 (2013): 15582-15612.
2. Al-Fuqaha, Ala, et al. "Internet of things: A survey on enabling technologies, protocols, and applications." IEEE Communications Surveys & Tutorials 17.4 (2015): 2347-2376.
3. Alizadeh, Mohammad, and Tom Edsall. "On the data path performance of leaf-spine datacenter fabrics." High-Performance Interconnects (HOTI), 2013 IEEE 21st Annual Symposium on. IEEE, 2013.
4. Apple Keynote. Online at: https://www.apple.com/apple-events/june-2017/
5. Atzori, Luigi, Antonio Iera, and Giacomo Morabito. "The internet of things: A survey." Computer networks 54.15 (2010): 2787-2805.
6. Bahar Farahani, Farshad Firouzi, Victor Chang, Mustafa Badaroglu, Nicholas Constant, Kunal Mankodiya, Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare, Future Generation Computer Systems, Available online 24 May 2017, ISSN 0167-739X, https://doi.org/10.1016/j.future.2017.04.036.
7. Bonomi, Flavio, et al. "Fog computing and its role in the internet of things." Proceedings of the first edition of the MCC workshop on Mobile cloud computing. ACM, 2012.
8. Cao, Yu, et al. "FAST: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation." Networking, Architecture and Storage (NAS), 2015 IEEE International Conference on. IEEE, 2015.
9. Chiang, Mung, and Tao Zhang. "Fog and IoT: An overview of research opportunities." IEEE Internet of Things Journal 3.6 (2016): 854-864.

10. Cisco Computing overview. Online at:

http://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf
11. Fog Computing And The Internet Of Things: Extend The Cloud To Where The Things

Are. 1st ed. Cisco, 2015. Web. 6 June 2017.
12. https://collectd.org/wiki/index.php/Plugin:Load
13. D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, “Internet of things: Vision,

applications and research challenges,” Ad Hoc Networks, vol. 10,no. 7, pp. 1497–

1516, 2012.
14. Dastjerdi, Amir Vahid, and Rajkumar Buyya. "Fog computing: Helping the Internet of

Things realize its potential." Computer 49.8 (2016): 112-116.
15. Explaining 5G. Spectrum IEEE Online. Video. Web. 15 June 2017.
16. Goetz, Christopher G., et al. "Movement Disorder Society-sponsored revision of the

Unified Parkinson's Disease Rating Scale (MDS-UPDRS): Scale presentation and

clinimetric testing results." Movement disorders 23.15 (2008): 2129-2170.
17. Gubbi, Jayavardhana, et al. "Internet of Things (IoT): A vision, architectural elements,

and future directions." Future generation computer systems 29.7 (2013): 1645-1660.
18. Guerra, Maria. "What Exactly Is A Smart City?". Electronic Design 65.2 (2017): 20-

22,35. Print.
19. Ha, Kiryong, et al. "Towards wearable cognitive assistance." Proceedings of the 12th

annual international conference on Mobile systems, applications, and services. ACM,

2014.
20. Hahm, Oliver, et al. "Operating systems for low-end devices in the internet of things:

a survey." IEEE Internet of Things Journal 3.5 (2016): 720-734.
21. IEEE 1471-2000, “IEEE Recommended Practice for Architectural Description of

Software-Intensive Systems”
22. Information processing systems -- Open Systems Interconnection -- Basic Reference

Model -- Part 2: Security Architecture. Online at:

https://www.iso.org/standard/14256.html
23. Internet of Things Guide: Cloud Definition Detail | Things Guide. Web. 15 June 2017.

24. ISO/IEC 18384-1:2016 Information technology. Online at:

https://www.iso.org/obp/ui/#iso:std:iso-iec:18384:-1:ed-1:v1:en
25. ISO/IEC 27001 Information Technology. Online at: https://www.iso.org/isoiec-

27001-information-security.html
26. ISO/IEC 27040:2015 Information technology. Online at:

https://www.iso.org/obp/ui/#iso:std:iso-iec:27040:ed-1:v1:en
27. Kamburugamuve, Supun, Leif Christiansen, and Geoffrey Fox. "A framework for real

time processing of sensor data in the cloud." Journal of Sensors 2015 (2015).
28. Lane, Nicholas D., et al. "An early resource characterization of deep learning on

wearables, smartphones and internet-of-things devices." Proceedings of the 2015

International Workshop on Internet of Things towards Applications. ACM, 2015.
29. Li, Jianhua, et al. "EHOPES: Data-centered Fog platform for smart living."

Telecommunication Networks and Applications Conference (ITNAC), 2015

International. IEEE, 2015.
30. Nexus IoT Glossary. Online at: https://www.nexusgroup.com/en/glossary/?letter=C
31. Okafor, K. C., et al. "Leveraging Fog Computing for Scalable IoT Datacenter Using

Spine-Leaf Network Topology." Journal of Electrical and Computer Engineering

2017 (2017).
32. OpenFog Architecture Overview White Paper. 1st ed. OpenFog Consortium, 2016.

Web. 6 June 2017.
33. OpenFog Consortium. Openfog Consortium Depiction Of The Pillars For Fog. 2017.

Web. 7 June 2017. Web.
34. OpenSSL: https://www.openssl.org/ (2015)
35. Roman, Rodrigo, Javier Lopez, and Masahiro Mambo. "Mobile edge computing, fog

et al.: A survey and analysis of security threats and challenges." Future Generation

Computer Systems (2016).
36. Sclater, N., Mechanisms and Mechanical Devices Sourcebook, 4th Edition (2007),

25, McGraw-Hill

37. Sheng, Zhengguo, et al. "A survey on the ietf protocol suite for the internet of things: Standards, challenges, and opportunities." IEEE Wireless Communications 20.6 (2013): 91-98.
38. Skouby, Knud Erik, et al. "Smart cities and the ageing population." The 32nd Meeting of WWRF. 2014.
39. Sobell, M.G.: A Practical Guide to Fedora and Redhat Enterprise Linux. Pearson Education (2013)
40. Wikipedia, Middle. Online at: https://en.wikipedia.org/wiki/Middleware. Web. June 2017
41. Wikipedia, Multi. Online at: https://en.wikipedia.org/wiki/Multitenancy. Web June 2017
42. YiYi, Shanhe, Cheng Li, and Qun Li. "A survey of fog computing: concepts, applications and issues." Proceedings of the 2015 Workshop on Mobile Big Data. ACM, 2015.
43. Zao, John K., et al. "Augmented brain computer interaction based on fog computing and linked data." Intelligent Environments (IE), 2014 International Conference on. IEEE, 2014.
44. Zdravković, Milan, et al. "Survey of Internet-of-Things platforms." 6th International Conference on Information Society and Techology, ICIST 2016. Vol. 1. 2016.