International Journal of Multimedia and Ubiquitous Engineering
Vol.11, No.11 (2016), pp.81-96
http://dx.doi.org/10.14257/ijmue.2016.11.11.07

Development of Crowd Monitoring and Informing System using Fog Computing Model

Eun-Kyu Lee

Dept. of Information and Telecommunication Engineering,
Incheon National University, Incheon, Korea
eklee@inu.ac.kr

Abstract

The concept of future hyper-connected city will include ubiquitous, frequent connections among people, objects, and our environment. In this sense, crowd information is re-interpreted as the level of interactions in which people communicate with physical devices at rooms and stores, and even on streets in various ways. Therefore, it becomes one of the most meaningful and dynamic data that our future data analysis technology will consume. In order to measure and inform crowd information to people, we propose to use an emerging fog computing model, instead of a conventional cloud. A fog moves portions of computing capability of the cloud down to the network edge and form a local computing environment. By distributing computing and networking services closer to where user data is generated, the fog best meet the emerging demands. We develop a testbed of fog computing architecture and implement a crowd monitoring service on top of it. Then, we deploy all the testbed on a public street and run experiments in a real world environment.

Keywords: *Fog computing, Internet of things, Smart street, Fog platform, Smart sensor, Physical computing*

1. Introduction

With advancement of Internet of Things (IoT) technology, a variety of sensors are collecting information about our environment in real time. It is not difficult to see temperature and humidity sensors at rooms that inform us of the status of where we sit. IoT also enables to monitor people's activities as well as our environments. For example, a smart phone today is with many sensors such as Touch ID, GPS, 3-axis gyro, accelerometers, ambient light sensors, and barometers. These sensors together are able to record how the owner uses her phone and where she goes in detail. It is expected that the number of these sensing devices will increase to 50 billion in 2020 [6]. We will get informed of many new information that we have never known without advanced sensors. Research on the IoT is to find what kind of new data we collect and to understand how to process it.

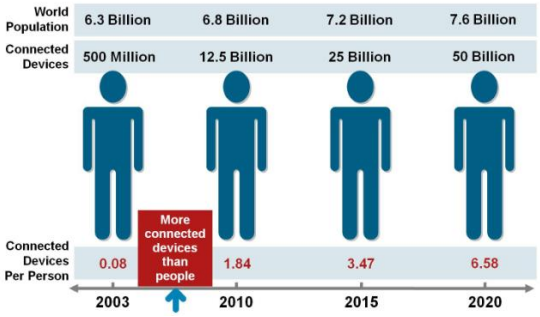

Figure 1. The Number of End Points Grows Exponentially (Source: Cisco IBSG [6])

ISSN: 1975-0080 IJMUE

This paper is interested in collecting and analyzing data about people around us. In particular, we collect and analyze crowd information either in a room or on a street since crowd information can make our lives more efficient and give us clue about people's mobility in physical spaces. For instance, we see long waiting-lines in front of café, restaurants, and banks during lunch break, and people often walk around to find another place. The walking tour may give us a chance to learn the street that we move around. But, employee and students having short break time definitely do not want to waste their time. Instead, they would rather go to a restaurant where they do not need to wait even for a minute. Our approach to resolve the issue is to inform people of crowd information of stores (indoor environment) or even of streets (outdoor environment) in real time. Upon receiving the information, people can make their own time plan more effectively. Moreover, a store owner may take advantage of such crowd information as statistical data for planning future business plan. Interestingly, the information can be interpreted to show a list of popularity among stores to potential customers.

In order to measure and inform crowd information to people, we propose to use an emerging fog computing model [2-4], instead of a conventional cloud. Our fog computing system consists of three parts: sensor nodes, a fog platform, and a local server. To measure crowd information, we count people passing by using infrared (IR) sensors. The count information detecting floating population is stored and analyzed in the fog platform, instead of being sent to the Internet cloud. Note that the platform is usually installed in a place close to the sensors or people; that is, it sits on an edge network covering local areas. Due to this locality, the platform is capable of calculating the degree of congestion of the relevant region. The crowd information is further analyzed so as to enumerate the popularity of specific areas. The goal of this work is to demonstrate the feasibility of the fog computing model by developing a crowd measuring system and running it on a real-world fog environment. Contribution of this work is listed as below.

1) We design a fog computing system that supports two core services: computing and networking. Our design also includes an important issue of fog interoperation with heterogeneous sensor nodes and the Internet cloud.

2) We build a testbed of a crowd measuring system in a real-world experiment and run experiments. The testbed includes sensor nodes that are able to count people by using IR sensors and a wireless fog platform.

The rest of the paper is organized as follows. Section 2 gives introduction about the emergence of fog computing over a conventional cloud. In Section 3, we design a system architecture of fog computing and networking. Section 4 describes our application scenario that measures crowd in a place using IR sensors. Section 5 gives details about our development and testbed. The next section illustrates our deployment of testbed in a real-world environment. Section 7 concludes this paper.

2. Fog Computing: Beyond the Cloud

Over the past decades, we have observed that computing services such as storage, data processing, and control have been pushed into the Cloud. The ability of unlimited computing in the cloud allows end users to access to ample information easily. We also have seen that mobile/sensing devices like smartphones and sensors have become powerful and pervasive, which leads to the emergence of new systems and applications. These systems and applications introduce new functional demands in computing and networking that the cloud alone cannot meet. Instead, local computing at a network edge is often necessary; say, to process data in real time, to create location-aware contexts from local sensors, and to maximize the efficiency of last mile wireless communications. However, the cloud is too far from the devices to satisfy strict latency requirements and is too centralized to cope with heterogeneity and contextual diversity at a local area. It is also too costly to upload every minutia data from individual sensors to the cloud.

Additionally, we report the risk in the increasing number of interactions in the future IoT network. It is obvious that the increasing number of sensors will increase the amount of data from everywhere, which enables to collect big data. The emerging big data analysis technology, then, will be expected to devise solutions that overcome numerous discomfort and existing problems. This big data architecture, however, assumes the existence of a centralized cloud system that is responsible for communicating with a myriad of sensor nodes. Sensor data shows the following properties. (i) Each data packet is small in size, (ii) data are from different sources and heterogeneous in its format, and (iii) numerous data traffic concentrate to the cloud. These properties increase the number of short, concurrent connections to the cloud and complicates database design.

To evaluate the effect of increasing numbers of interconnections, we carry out an experiment that measures overhead and performance on a server with increasing numbers of concurrent connections. In the experiment, a server runs a web server on Intel Atom N2600 dual core (1.6GHz) CPU and 2GB DDR3 memory. It accepts connections from sensor nodes via HTTP GET method. We prepare two connection sets: one in which 3,000 concurrent sensors generate connection requests 1,200 times in a second (type 1) and the other which we reduce the number of current sensors from 3,000 to 300 without changing other variables (type 2). Then, we measure response time and error rate in handling the requests at the server using Apache JMeter [14].

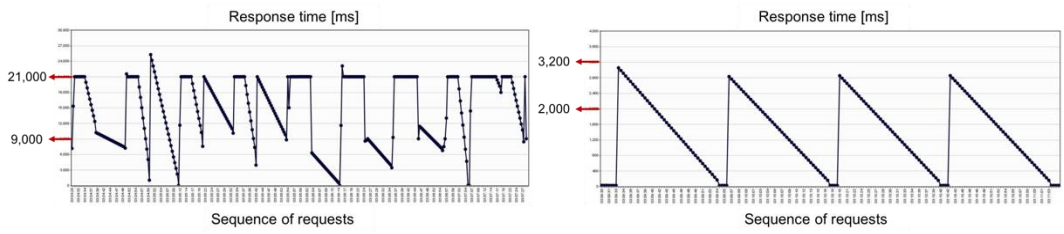

Figure 2. A Server Response Time [ms] Measured in the Experiment: Left with Type 1 and Right with Type 2

On the left in Figure 2 is a graph drawing response time with type 1 connection request. It shows that the response time fluctuates with maximum value of 25,500 ms. An average is recorded at 21,000 ms. We also measure that the error rate goes up to 4.30%. The result of another experiment with type 2 connection request is shown on the right of the same figure. It takes around 3,000 ms with maximum of 3,200 ms. The error rate is 0.92% on average. The experimental results indicate that the increasing number of interconnections significantly degrades network and service performance although the packet size is small. One solution to resolve the performance problem is to distribute traffic load or filter unnecessary connections.

3. Design of Fog Computing and Networking System

To overcome the limitations, portions of computing capability of the cloud move down to the network edge and form a local computing environment, a "fog". By distributing computing and networking services closer to where user data is generated, the fog best meet the emerging demands. The fog presents a new architecture that "takes processing to the data" while the cloud "takes data to the processing." Edge devices and mobile devices are inter-connected together within a local fog network, and they collaboratively carry out storage, data processing, networking, and control tasks.

The fog will resolve many problems in the Internet of Things (IoT) and 5G mobile networks. For instance, fog services are capable of overcoming the bandwidth and cost constraints for long-haul communications; performing data fusion and streaming analytics

in real time; managing a large volume of devices and cyber-physical interactions; coping with network reliability and resiliency; and securing untethered, resource-constrained end devices. However, the fog computing and networking still remains many research challenges - how to model a system architecture for the fog; how to deploy, arrange, and manage fog devices; how does the fog interact with the cloud and with user devices; how to manage physical/logical connectivity in the fog.

3.1. Fog Computing

As an emerging paradigm, there are a few definitions of "fog computing". Below are some examples.

"*The fog extends the cloud to be closer to the things that produce and act on IoT data. These devices, called fog nodes, can be deployed anywhere with a network connection: on a factory floor, on top of a power pole, alongside a railway track, in a vehicle, or on an oil rig. Any device with computing, storage, and network connectivity can be a fog node. Examples include industrial controllers, switches, routers, embedded servers, and video surveillance cameras.*" – Cisco Systems

"*Fog computing or fog networking, also known as fogging, is an architecture that uses one or a collaborative multitude of end-user clients or near-user edge devices to carry out a substantial amount of storage (rather than stored primarily in cloud data centers), communication (rather than routed over the internet backbone), and control, configuration, measurement and management (rather than controlled primarily by network gateways such as those in the LTE core network).*" – Wikipedia [15]

"*Fog computing is a geographically distributed computing architecture with a resource pool consists of one or more ubiquitously connected heterogeneous devices (including edge devices) at the edge of network and not exclusively seamlessly backed by cloud services, to collaboratively provide elastic computation, storage and communication (and many other new services and tasks) in isolated environments to a large scale of clients in proximity.*" – Reference [16]

Fog computing is an extension of the cloud (geographically) to an edge network. It is a platform that provides computing and networking services both to end devices and to cloud data centers. It is named "fog" since computing services are performed in a fog under a cloud. A fog computing model does not send all the sensor data to a data server on the Internet cloud. Instead, a gateway platform (e.g., network router, smartphones, set-top boxes, PCs, and micro servers) that are close to data sources collects and processes sensor data immediately. This reduces service latency so as to enable to interact with end devices quickly and autonomously and to provide convenient service interfaces to users. Fog computing is distributed over geographically different regions unlike the Internet cloud. Thus, it can use proxies and access points to provide services to mobile end devices such as vehicles. Such a distributed property also contributes to distribute data traffic. Its distributed computing nature can also distribute data traffic, avoiding concentration to the Internet cloud servers.

3.2. Fog Architecture

A fog computing model includes a "fog layer" in which there are a variety of fog platforms. A platform could be a normal desktop PC equipped with conventional hardware components such as CPU, memory, and network cards allowing it to communicate with the Internet. In addition, it is with other communication modules such WiFi, ZigBee, and Bluetooth that allows the platform to interact with end devices. Or, a

wireless access point can play the platform role. The platform is not defined as hardware specification, but its role in the overall system architecture. Figure 3 presents the idealized information and computing architecture supporting the future IoT applications, and illustrates the role of fog computing. One may say that a fog platform is literally a cloud server that performs its computing processing in the middle of fog layer. Often, the fog layer is tightly coupled with the end devices.

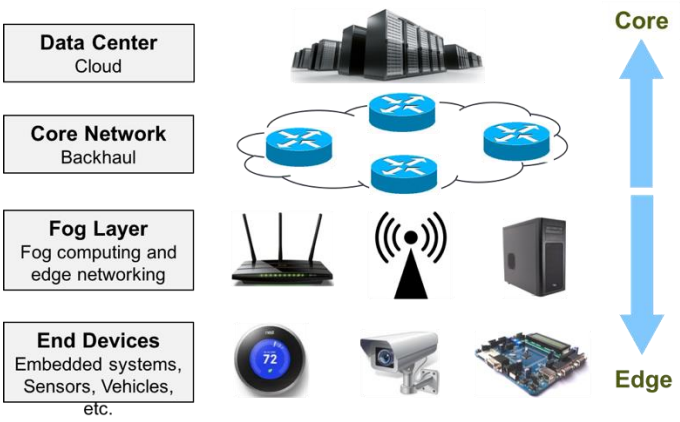

Figure 3. A System Architecture of Fog Computing Model

The fog computing model has three distinguishable features as follows. First, it provides a localized and virtualized computing platform that supports load balancing and responses to end devices in real time. This is very effective in IoT applications that require an immediate and accurate response. Next, the fog layer can play a manager role for unreliable wireless communications. Most of end devices are connected via various wireless communication technologies such as WiFi, ZigBee, and Bluetooth. But, wireless media is known for unreliability and unpredictability, and the Cloud alone has failed to manage such long-haul connections. Using mobile networks such 3G or LTE increases cost, which is not affordable in emerging IoT scenarios. The fog platform is at the right place and has enough intelligence to improve communication quality. Last, the fog layer is able to manage mobility. The fog model is basically a distributed computing system where each platform is now intelligent enough to cope with mobility issues of connected end devices.

4. Measuring Crowd in a Place using IR Sensors

4.1. Scenario

It is not difficult to find long waiting-lines at restaurants, banks, and café during lunch time. Employees and students have less than one hour for their lunch break, and thus they often waste time by walking around looking for another place. A clear solution to reduce the waste is to inform them of congestion information at stores in advance. If an employee is given such information, she can make an efficient plan for her lunch break. She may have a lunch, go to a bank, and then meet a friend at a café even within such a short break. Congestion information (more specifically people counts at a store) can be collected and further analyzed so as to create new knowledge. We evaluate the popularity of stores from such big data and this new knowledge can be provided to future customers. In this experiment, we design a system that counts the bypassing people and then measures and displays the number of people at nearby stores. We build its testbed by using the proposed fog system.

People counting has been an interesting research issue in academia, and we also see many off-the-shelf products in a market. In a real-world, a manual counter is still widely

used. An employee counts people on her eyes and presses a button on the counter. Not to mention intrinsic human errors, this method does not work well (i.e., inaccurate in count) in a crowd environment in a large place. The most popular approach in people counting uses cameras or image sensors to capture still images from which a vision technology recognizes individuals [8]. Figure 4 shows a vision program that counts people from an image captured by a ceiling camera. While this approach provides high accuracy in counting, it often requires an advanced vision algorithm that slows down the counting process and increases costs.

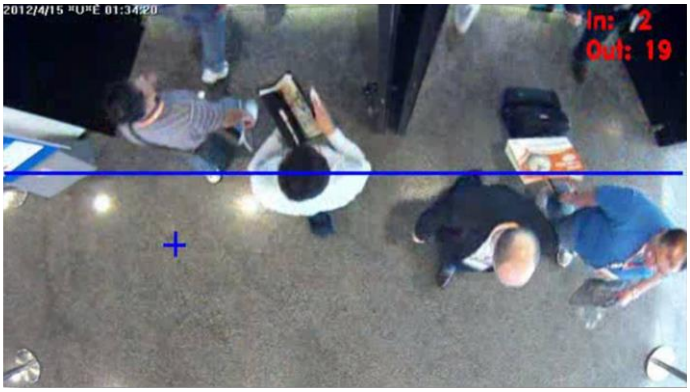

Figure 4. A Vision Program Counts the Number of People from an Image

4.2. Design Approach

The goal of our system is to develop a people counting system that costs reasonable enough to be deployed in as many stores as possible in a real world.

In order to measure congestion information, we must be able to count the number of people who come into or go out of a store. To this end, our system uses infrared (IR) sensors that measure infrared light radiating from objects in its field of view. IR sensors have a huge advantage in its low cost. When counting people, there is a need to distinguish between people going out of the store (out count) and those who come into the store (in count). In order to solve this problem, we place two IR sensors in row as shown in the left part of Figure 5.

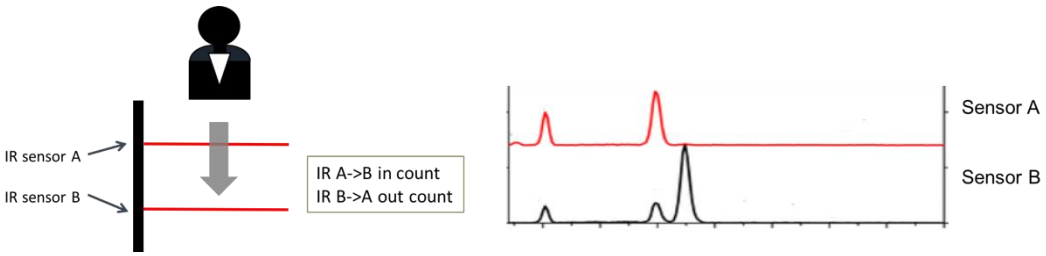

Figure 5. Two IR Sensors Distinguishes People Coming in or Going Out. The Curve Represents the Case where one Goes through the Sensor A and then B

Upon receiving signals from the sensors, our system distinguishes the out count from the in count by detecting which signal comes first. For example, when a signal from sensor A is recorded first followed by a signal from sensor B, two signals are detected just as shown in the right part of the same figure. The curve indicates that a person comes in to the store (in count). Likewise, in case a signal from sensor A follows that from sensor B, our system recognizes that a person goes out of the store (out count).

5. Testbed Development

Our testbed consists of three parts: a sensor device counting people, a fog platform processing the counting data, and a server storing data permanently for future use.

5.1. Sensor Device

We develop a sensor device that is responsible for counting people and forwarding the counting record to the fog platform.

The sensor device is composed of three components: an embedded platform, a sensor, and a communication module. For the platform, we use an Arduino [9] that is an open-source prototyping platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. We first test with an Arduino Nano, the smallest board in the Arduino family. But, the Nano does not work with two IR sensors (it failed to measure maximum sensor values) mainly due to its limited computing power. Instead, we use an Arduino Uno whose specification is illustrated in Figure 6.

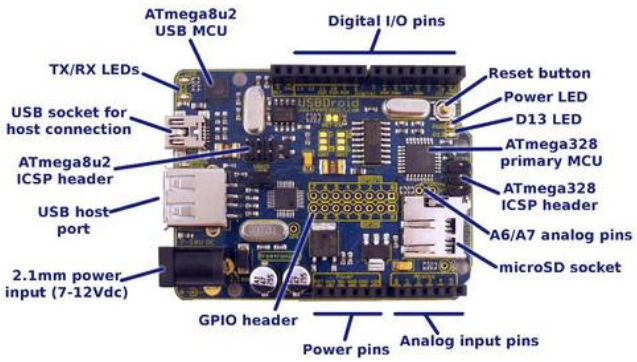

Figure 6. Our Testbed Uses an Arduino Uno as the Hardware Platform of a Sensor Device

For communications, the sensor device uses a WiFi technology after considering a list of wireless communication options such as WiFi, Bluetooth, and ZigBee. There are a few reasons for our choice. First, WiFi has a longer radio range than competitors, which is critical in our testbed since we plan to deploy our system mainly to outdoor environment or at a large indoor place such as a shopping mall. In such environment, each sensor node must be able to communicate with our fog platform. Next, the sensor device is installed at a front door of a store. This means that it can be easily powered, and thus focus only on communication performance. Last, our fog platform now supports WiFi communications only at this version. We use Wifly Shield [10] for our Arduino device, and Figure 7 illustrates the shield and its features.

- Qualified 2.4GHz IEEE 802.11b/g transceiver
- High throughput, 1Mbps sustained data rate with TCP/IP and WPA2
- Ultra-low power - 4uA sleep, 40mA Rx, 210mA Tx (max)
- Small, compact surface mount module
- On board ceramic chip antenna and U.FL connector for external antenna
- 8 Mbit flash memory and 128 KB RAM
- UART hardware interface
- 10 general purpose digital I/O
- 8 analog sensor interfaces
- Real-time clock for wakeup and time stamping
- Accepts 3.3V regulated or 2-3V battery
- Supports Adhoc connections
- On board ECOS -OS, TCP/IP stacks
- Wi-Fi Alliance certified for WPA2-PSK
- FCC / CE/ ICS certified and RoHS compliant.
- Industrial (RN-131G) and commercial (RN-131C) grade temperature options

Figure 7. The WiFly Shield Equips our Arduino the Ability to Connect to 802.11b/g Wireless Networks

For the sensor, we choose an IR sensor that has a measurement range of 1.8 meter. But, a preliminary test that we carry out in our laboratory reveals that the sensor generates incorrect sensor records because a reliable range is much shorter than that in data sheet. We change our sensor to Sharp GP2Y0A710K0F that has variable distance measuring range from 100 cm to 550 cm. Figure 8 demonstrates characteristics of distance measurement (output) of the sensor [11].

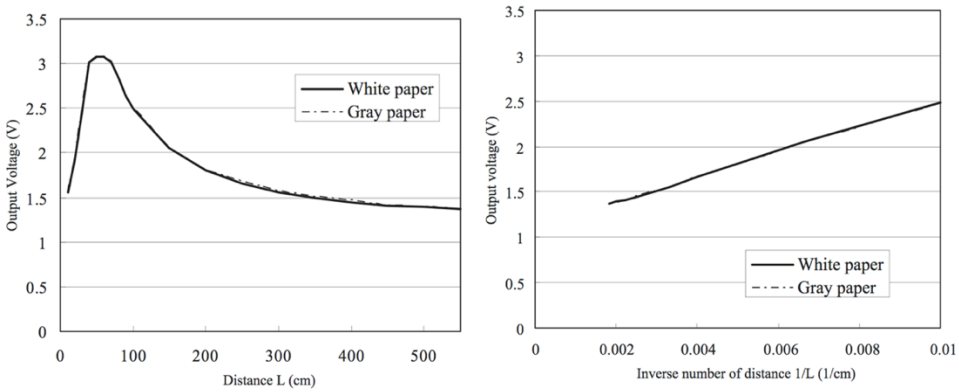

Figure 8. The IR Sensor Supports Distance Measuring Range from 1m to 5.5m. The Reflection Ratios in the White and Gray Papers are 90% and 15%, Respectively

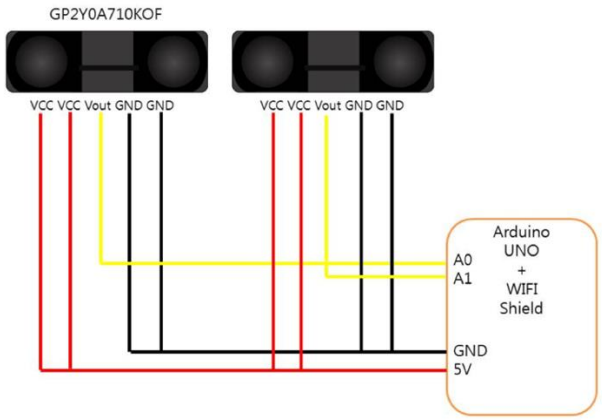

Figure 9. A Circuit Diagram that Connects two IR Sensors to Arduino

Figure 9 shows that two IR sensors are connected to the Arduino in our testbed. We connect Ch 1 and Ch 2 to two IR sensors respectively. If Ch 1 senses and generates a signal first followed by Ch 2's, then our system interprets "in count". The system records

"out count" upon receiving the signals in a reverse order. Our preliminary test also finds small measurement error in the sensor; the sensor generates twenty values to represent one signal and it sometimes generates 1~2 values going out of normal range of specification. Thus, we calculate a signal by averaging 18 values after dropping 2 minimum values. We also consider three types of shield cases as shown in Figure 10. An IR sensor can be installed on the floor, to the ceiling, and/or on the wall with proper shield cases, where a case contains two sensors. Our testbed tests and uses all three types of installation.

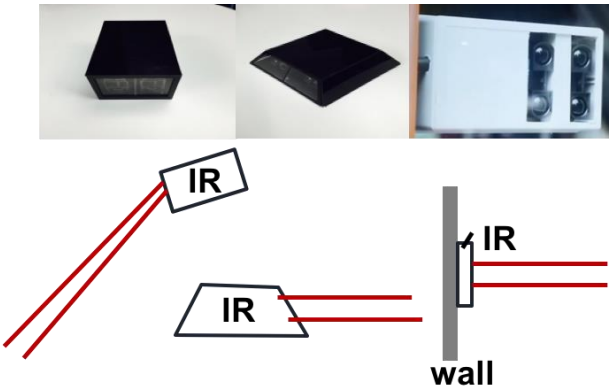

Figure 10. An IR Sensor can be Installed on the Floor (left), to the Ceiling (middle), and on the Wall (right) with Proper Shield Cases

5.2. Measuring Sensing Accuracy

We conduct two experiments to evaluate how accurately our sensors can detect a person passing by. The first experiment is to evaluate the accuracy of an IR sensor itself. Our sensor is designed to detect an object passing by measuring distance. More specifically, the sensor records distance of 180 cm ~ 190 cm when there is no object passing by. When an object passes, the distance decreases below 180 cm; that is, the sensor records 153 cm, say. It is critical to evaluate how accurately the sensor measures the distance since its performance impacts significantly on the accuracy of people counting. To this end, we place an IR sensor having 500 cm of distance measuring range without any object and check to see if the sensor measures 180 cm ~ 190 cm consistently.

Figure 11. An IR Sensor Measures a List of Distance Values when no Object Passes by

Figure 11 shows a list of distance values measured by the sensor in the experiment. In particular, the red boxes in the figure are the cases where distance values go below 180

cm, meaning that the sensor mistakenly detects an object. In order to get more details, we repeat our test 500 times and 1000 times. Error (distance values going below 180 cm) occurs 5 and 9 times in the 500-times and the 1000-times experiments, respectively, representing 1% and 0.9% of error rate. In order to fix the error, we propose three engineering solutions. i) A partition board is inserted between two IR sensors, which helps avoid interference between signals from the sensors. ii) We added a capacitor of 0.1 uF into the sensor circuit, which improves physical errors and stabilizes a voltage value. iii) We average 20 values, which helps dropping outliers. After advancing the sensor with our solutions, it shows 0% of error (approximately).

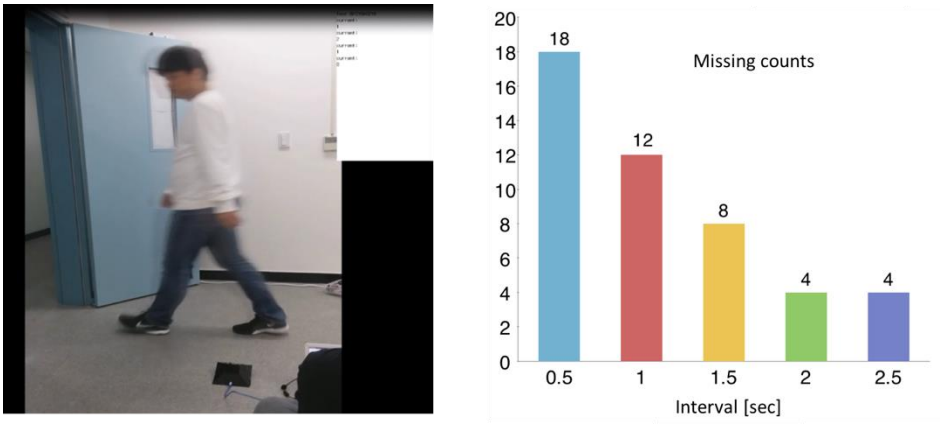

Figure 12. We Run an Experiment to Evaluate how the Sensor Accurately Detects People Passing by and Draw Results in Bars

Next experiment is to evaluate how accurately the sensor can detect people passing by. To this end, we place the sensor on the floor and 50 people passes by the sensor in the 2.5 second interval. Figure 12 captures the experiment (left) and draws the results (people counting) in bars (right). The sensor fails to detect 2 people out of 50 (4 missing counts in the result bars), which accounts for 4% of error rate. We then decrease the interval to 2 second; that is, the first person passes by the sensor followed by the second person after 2 second. The sensor still fails to detect 2 people (the same error rate). We gradually decrease the interval to 1.5, 1, and 0.5 seconds and repeat the experiment. The missing count increases to 4, 6, and 9 people out of 50, corresponding to 9, 11, and 18% of error rates, with decreasing intervals, respectively. We highlight that the error rate almost reaches 20% with 0.5 second of interval. This implies that the sensor may fail to count people correctly when two or more people pass by together. In fact, this is mainly attributed to intrinsic limitation of the IR sensor. As a potential solution to this problem, we propose to use a piezo sensor that measures changes in pressure, acceleration, temperature, strain, or force. An outdoor carpet may be equipped with the piezo sensor and help count people more accurately. We leave this idea as one of our future research topics.

5.3. Fog Platform

Fog computing extends the cloud computing paradigm to the edge of the network. In this emerging architecture, a fog platform places close to smart sensors and meters and host many applications using data generated from these sensors and meters.

With close collaboration with Cisco Systems, Inc., we use Cisco's Connected Grid Router series (CGR 1240) [12] as our fog platform that is responsible for connecting to our sensor nodes and for running our application of crowd analysis. It also connects to our edge server throughout Ethernet connection. CGR is designed for use in field area

networks and offers a reliable communications platform for smart metering, distribution automation, and remote workforce automation. It is modular and supports a variety of communications interfaces, such as WiMAX, 2G, 3G, and 4G LTE, 900 MHz RF Mesh, Ethernet, and Wi-Fi. In our testbed, it communicates with the sensor node via WiFi communications. Below is a list of its features [12].

- Rugged industrial design and compliance with IEC-61850-3 and IEEE 1613 for utility substation environments
- Multi-service features including IPv6, advanced routing, IP multicast, and quality of service
- Comprehensive security capabilities based on open standards
- A highly resilient design that optimizes communications network uptime and availability
- Network and device management tools for easy deployment, upgrades, and remote monitoring

Figure 13. Our Fog Platform uses CGR 1240 Router that has a Hypervisor Architecture

CGR implements a hypervisor architecture in which Cisco's IOS software runs as a virtual machine (VM) as shown in Figure 13. The hypervisor can support other VMs that users can choose. Currently, a stock Linux OS is packaged with the CGR image. When this image is installed, a Guest OS instance is automatically created on the CGR. The Guest OS on the CGR is like a regular Linux host, and can be used to run utility applications. The following sections give details on how to start and manage the Linux Guest OS. Details of using the Guest OS and IOS to manage applications are described separately.

CGR allows to configure IOS to provide network connectivity to guest OS. The internal connectivity between Guest OS and IOS is described in Figure 14. Network connectivity to GOS is provided by one of the external interfaces managed by IOS. Traffic from Guest is forwarded by IOS through regular IP forwarding mechanisms.

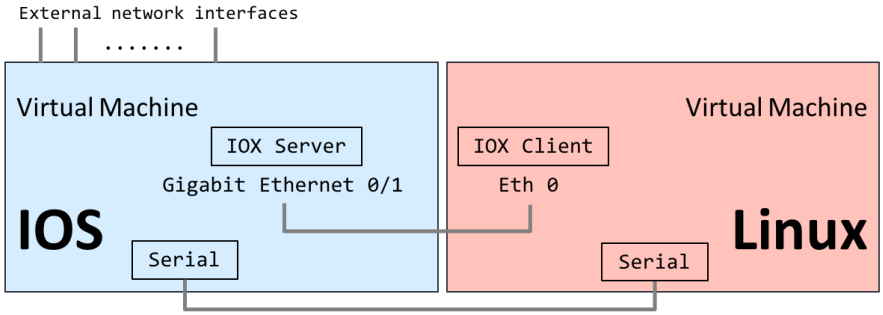

Figure 14. CGR Allows to Configure IOS to Provide Network Connectivity to Guest OS

5.3. Local Server and Display

The testbed also includes a server that is responsible for storing sensor data permanently and provide crowd information to users within the fog area via a display monitor. We implement our server using Cisco's Edge 340 that is a Linux-based open platform with powerful computing and multimedia capabilities [13]. It integrates rich connectivity to enable all the essential components of a digital connected room experience with Ethernet LAN uplink, wireless access, rich media, and application computing. It is also an open application platform that allows us to customize it to enable vertical solutions (Figure 15).

- CPU
  - Intel Atom N2600 / Dual Core, 1.6GHz / Integrated GPU
- Memory
  - 2GB DDR3 800MHz
- Storage
  - 32G built-in SSD
  - 1 SD slot for extension
- Size
  - L×W×H: 244×186×35mm

- Network Interfaces
  - 1*GE uplink
  - For Wi-Fi SKU: Wi-Fi: 802.11a/b/g/n, dual-band, AP or Client mode
  - Bluetooth (v4.0)
- Other interfaces
  - 4*USB2.0 ports (Type A)
  - IR Remote and extender
  - RS-232(DB-9)
  - Console(RJ45)

Figure 15. Our Local Server uses Edge 340 to Enable the Crowd Solution

The server runs Linux Fedora v16 kernel 3.1 with Cisco firmware 1.1. It also runs APM (Apache, PHP, MySQL) that allows to communicate with the fog platform and eventually with sensor nodes. The database stores all the raw data collected from the sensor nodes. The server provides crowd information through a few channels to end users (i.e., web services interface and display). Support for enhanced media capability is another feature of the server. It supports various formats of graphics, flash play, and HTML 5 functionalities of canvas, audio, and video. Figure 16 shows a list of software components configured in the server (left) and depicts a process in which sensor data is collected, stored, analyzed, and displayed.

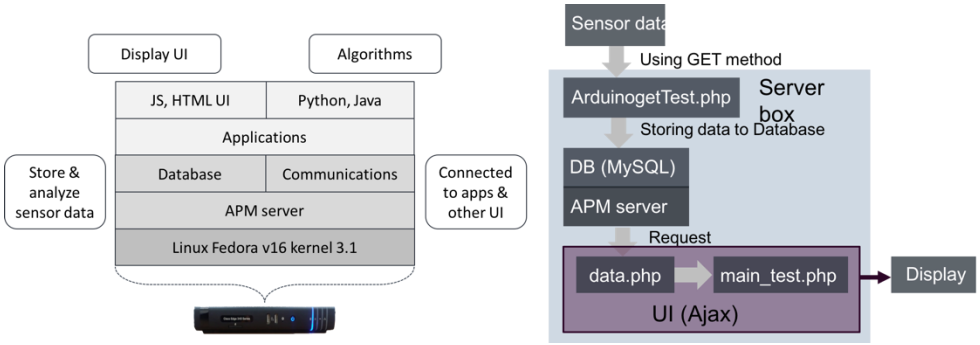

Figure 16. Our Local Server Consists of a Few Software Modules to Collect, Store, Analyze, and Display Sensor Data

Sensor data is transmitted to the server using the GET method via HTTP protocol. It comes with three values. The first value is the unique identification of a sensor node installed that also indicates the node's physical location. The second and third values are in count and out count values from the node, respectively. Sensor data is collected every 3 minutes. Collected data is stored in database and displayed at a large screen installed on a street. We use Ajax technology to develop a UI module that displays sensor data on a

screen. The module consists of two main components. “data.php” obtains current crowd information and floating population by accessing database. “main_test.php” calls an Ajax function to obtain the output data generated from the “data.php” component in JSON format. Figure 17 illustrates the overall system architecture of our testbed for monitoring crowd, which follows the concept of fog computing model.

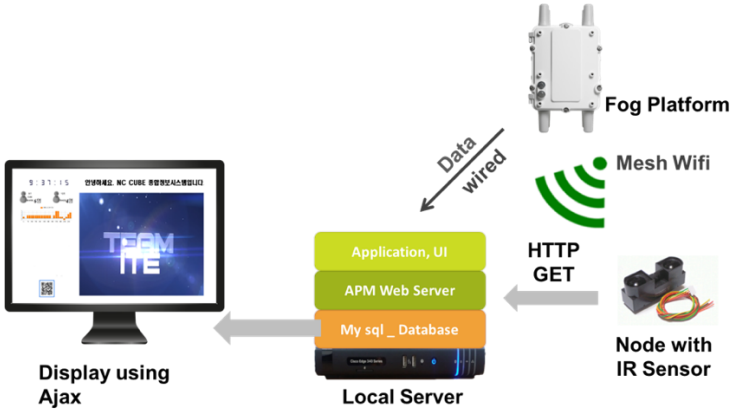

Figure 17. A System Architecture of the Developed Testbed for Monitoring Crowd

6. Deployment on a Real World Environment

We deploy our crowd monitoring system in a real world environment and run experiments to demonstrate the feasibility of a fog computing model. Our system is deployed at NC Cube that is an outdoor shopping mall having four streets: Spring, Summer, Autumn, and Winter. In particular, our mother project installs a kiosk, named IoT Cube, on the entrance of the Winder street as shown in Figure 18. The crowd monitoring system installed on the Cube counts both people visiting the Cube and people visiting the street.

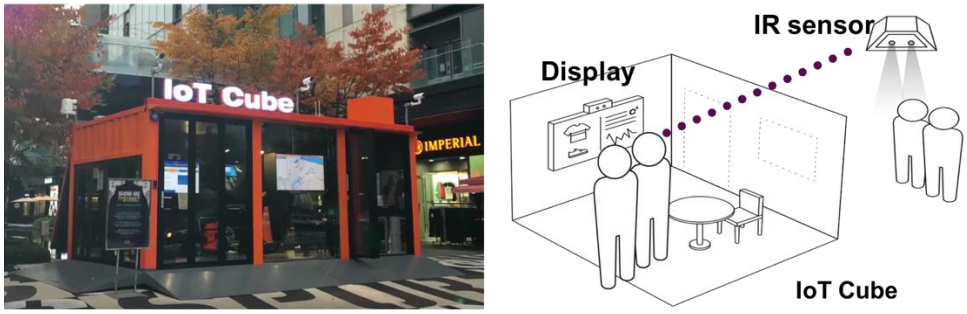

Figure 18. The Crowd Monitoring is Installed on the Cube and Counts People

We install three IR sensor nodes on the Cube as shown on the left in Figure 19 that also illustrates the floor plan of the Cube. IR [3] is to count people coming into the Cube while IR [1] and IR [2] are installed to count people entering the Winter street. The red arrows indicate the directions of infrared signals generated from three sensor nodes. On the right of the figure is IR [3] installed inside the Cube. It is attached to the ceiling, but tilt in a diagonal direction so as to detect people in a correct way. IR [1] and IR [2] are also installed inside the Cube but directed to outside, and sends out signals throughout the Cube’s windows.

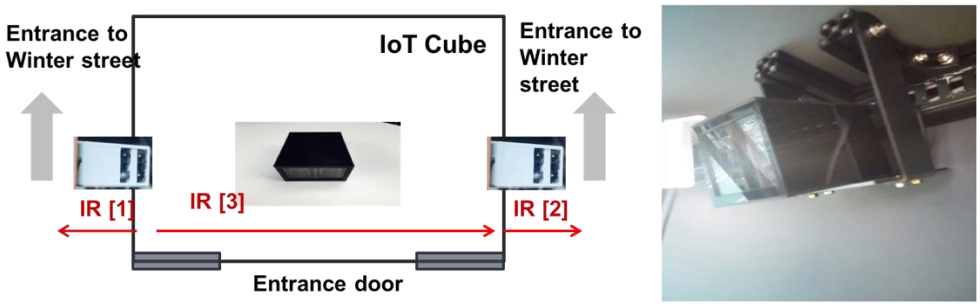

Figure 19. Three IR Sensor Nodes are Installed on the Cube and Detects People

Three sensor nodes record counting data and sends it to the fog platform and eventually to the local server. Then, the server computes the current number of people in the Winter street and in the Cube. This calculated values are expressed as a numerical value and the bar graph in real time via a display. Figure 20 captures the display screen showing current congestion information. The bars represent statistical information by showing the number of people coming into the Cube at different times during the day. The screen also displays commercials and ads of the stores in the mall. This is quite effective because most people coming into the street are interested in the stores and the owners of stores may consider them as potential buyers with high probability.

Figure 20. Our Server also Provides all the Sensor Information Including Crowd via a Display Screen Installed Inside the Cube

7. Conclusion

This paper has discussed a new paradigm in a computing model, fog computing. The fog is expected to resolve many problems in the Internet of Things (IoT) and 5G mobile networks. For instance, fog services are capable of overcoming the bandwidth and cost constraints for long-haul communications; performing data fusion and streaming analytics in real time; managing a large volume of devices and cyber-physical interactions; coping with network reliability and resiliency; and securing untethered, resource-constrained end devices. However, the fog computing and networking still remains many research challenges. To study more on fog computing, we have developed a testbed of fog system and run a crowd measuring system in a real world environment.

Acknowledgments

This work has extended previous research results [1]. This research was supported by Incheon National University research grant in 2016.

References

[1] E-K. Lee, Introduction to Fog Architecture: Beyond the Cloud System, International Conference on Information Technology and Computer Science, (2016), July.
[2] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog Computing and Its Role in the Internet of Things, Workshop on Mobile cloud computing, (2012).
[3] S. Yi, Z. Qin, and Q. Li, Security and Privacy Issues of Fog Computing: A Survey, Wireless Algorithms, Systems, and Applications, (2015), August.
[4] M. Chiang, Fog Networking: An Overview on Research Opportunities, Technical Report, Princeton University, (2015), December.
[5] L. M Vaquero, L. Rodero-Merino, "Finding your Way in the Fog: Towards a Comprehensive Definition of Fog Computing", pp.27-32. ACM SIGCOMM Computer Communication Review, Vol.44, Issue 5, (2014), October.
[6] D. Evans, "The Internet of Things How the Next Evolution of the internet is Changing Everything", Technical report, CISCO IBSG, (2011), April.
[7] F. Bonomi, R. Milifo, P. Natarajan, and J. Zhu, "Fog Computing: A Platform for Internet of Things and Analytics" In Big Data and Internet of Things: A Roadmap for Smart Environments, Springer International Publishing, (2014), pp.169-186.
[8] D. Yang and H. Gonzalez-Banos and L. Guibas, Counting People in Crowds with a Real-Time Network of Simple Image Sensors, IEEE International Conference on Computer Vision (ICCV), (2013).
[9] Arduino, open-source hardware, at https://www.arduino.cc/.
[10] SparkFun WiFly Shield - WRL-09954 - SparkFun Electronics, at https://www.sparkfun.com/short/9954
[11] Sharp GP2Y0A710K0F IR Range Sensor, at http://www.sharp.co.jp/products/device/doc/opto/gp2y0a710k_e.pdf
[12] Cisco Connected Grid Router (CGR) 1240, at http://www.cisco.com/c/en/us/support/routers/1240-connected-grid-router/model.html
[13] Cisco Edge 340, at http://www.cisco.com/c/en/us/products/switches/edge-340-connected-platform/index.html
[14] Apache JMeter tool, at http://jmeter.apache.org/
[15] Fog computing, Wikipedia, at https://en.wikipedia.org/wiki/Fog_computing
[16] S. Yi, Z. Hao, Z. Qin, and Q. Li, Fog Computing: Platform and Applications, IEEE Workshop on Hot Topics in Web Systems and Technologies, (2015), December

Author

Eun-Kyu Lee, Dr. Lee is an assistant professor in the Department of Information and Telecomm. Engineering of Incheon National University (INU). Before joining INU, he had worked for Symantec Research Labs (LA, USA) and Electronics and Telecommunications Research Institute (Daejeon, Korea) for several years. He has been involved in many research projects in the fields of smart grid, smart building, electric vehicle network, location-based services, and Telematics. He received M.S. and Ph.D. in Computer Science from the University of California, Los Angeles in 2011 and 2014, respectively. His research interest includes Internet of Things, cyber-physical systems, wireless networks, cybersecurity and privacy, and pervasive physical computing.