An Analysis of the Goals and Requirements of IoT Systems

David I. Chong
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
chong050@morris.umn.edu

ABSTRACT

In this paper I detail information needed in order to understand and analyze an Internet of Things (IoT) system. To do this, I describe two applications of IoT systems. The first examines researchers efforts in designing systems to detect potentially problematic mosquito populations. The second addresses security concerns that arise from cloud disconnection from the perspective of designing a resilient smart home.

Keywords

Internet of Things, System analysis

1. INTRODUCTION

The Internet of Things, abbreviated as IoT, can be defined as an interconnection via the Internet of computing devices embedded in objects which enables them to send and receive data. We can think of the IoT as a way of classifying a system. Due to the definition's broad nature, IoT has a wide variety of possible applications. I will be detailing: what an IoT system is, what goals IoT systems have, how those goals can be met, and some issues that may arise when working with IoT systems.

I will be analyzing two use cases of IoT systems. The first being researchers efforts in designing a system to prevent potentially problematic mosquito populations. The second being researchers efforts in identifying and handling potential problems that may arise when a smart home disconnects from the network or when cloud computing services become unavailable. I will discuss the differences both in design and goals between the systems as well as why certain requirements are needed in one system while not in the others. This could help future designers identify important aspects of their own system or give a framework for thinking about an IoT system.

2. BACKGROUND

2.1 Layers

IoT systems are generally described in terms of several interconnected layers. During my research I ran into many IoT models, so it is worth noting that this may vary from topic to topic. For my purposes I will be identifying three distinct layers, which together form a complete IoT system (see Figure 1).

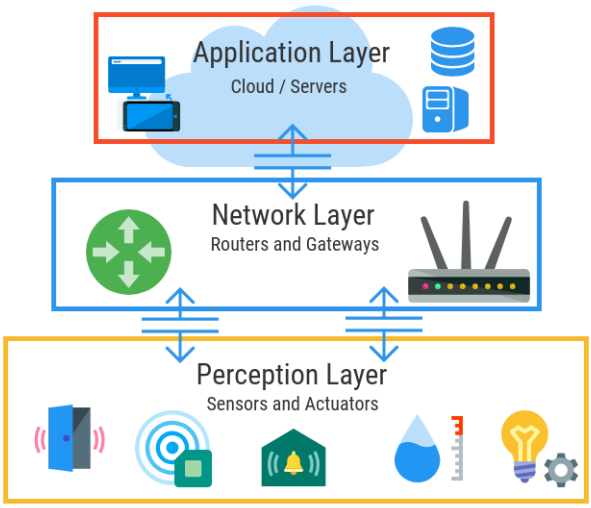

Figure 1: The three layers of IoT [3]

These layers are:

- the perception layer
- the network layer
- the application layer

2.1.1 *Perception layer*

The perception layer takes in physical properties of the things around us collected from sensors that are part of the IoT system. These devices generate the information which is later processed by the application layer. These devices typically have the ability to be interacted with remotely over the network layer. Actuators are devices that can impact the state of the system, and also typically reside in the perception layer. The perception layer can be thought of as the "Things" in IoT. I will talk about some of the devices which could be a part of the perception layer in a later section.

UMM CSci Senior Seminar Conference, April 2019 Morris, MN.

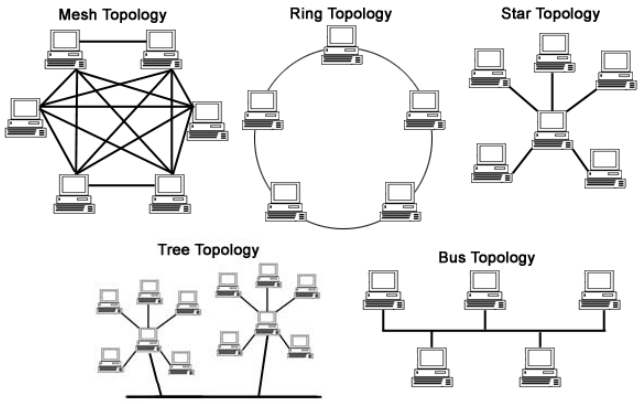

Figure 2: Examples of various network topologies [2]

2.1.2 Network layer

The network layer is the connection between the perception and application layers. It allows the transfer of information from one layer to the other. Various network technologies and communication protocols can be used based on the system needs. We must also consider the physical and network connection's design. This is known as a network topology. There are a variety of different network topologies (see Figure 2). Each benefit the network in different ways. I will talk about these technologies, protocols, and topologies in a later section.

2.1.3 Application layer

The application layer in these examples can be classified as one of two types: local computing or cloud computing. It is important where the data is being computed. In recent years we have seen a rise in cloud computing. This has also led to researchers finding potential problems related to the loss of access to cloud computing and what sort of reaction should happen when this access is lost to avoid security concerns as well as maintaining adequate service. With local computing we are able to avoid some of the issues that have been identified by researchers in relation to cloud computing. I will go into more detail about the advantages and disadvantages of both of these computation methods and how they can be used potentially in conjunction to improve system resilience.

2.2 Optimization

Considering costs in an IoT system tends to relate to computation and communication costs. Decisions about how to optimize a system depend on factors such as maintenance, power consumption, and available computational power.

In general, we can describe these optimizations in terms of the following two categories:

- Software optimization
- Hardware optimization

2.2.1 Software optimization

In the context of this paper I will be focusing on parallel computing and automatic vectorization and their contributions to the more efficient operation of a piece of software.

Parallel computing is a type of computation in which many calculations or the execution of processes are carried out simultaneously. Large problems can often be divided into smaller ones, which can then be solved at the same time [5].

Automatic vectorization, in parallel computing, is a special case of automatic parallelization, where operations are applied to whole arrays of variables instead of individual element variables.

Software can also be optimized in other ways such as taking advantage of trading off precision of computational accuracy in favor of quicker computing leading to a reduction in resource usage. Different decisions about where the computation should happen may depend upon what limitations are most important in the system.

2.2.2 Hardware optimization

Hardware optimization focuses on improving efficiency by reducing power consumption and increasing performance and scalability in the memory system through modification related to system hardware. It is worth noting that hardware optimization is different than hardware acceleration.

Hardware acceleration is the process by which an application will use other hardware components of a system in order to perform certain tasks more efficiently. An example of this would be the use of the GPU to process information rather than the CPU in Bitcoin mining. The computations are more efficiently computed by the GPU due to the type of information needing to be processed.

3. MOSQUITO DETECTION

3.1 Description

The study conducted by Ravi et al. [6] identified the issue that current mosquito identification methods are ineffective. It points out that our current methods rely on human acknowledgement of the issue and physical reporting. This is a ineffective method in some cases as locations where mosquitoes could breed are inaccessible to humans in spite of being close enough to impact human populations near and vulnerable to mosquito-borne illnesses. This could be due to the area in question being remote, but still in range of humans, or dangerous to monitor. Locations which are particularly at risk to mosquitoes are also typically have poorer populations meaning that the solution must be cost effective.

These reasons point to a need for a system which can satisfy the following goals:

- Low cost per device
- Able to be deployed to remote locations
- Able to run for extended period of time - highly optimized by:
  - High computational efficiency
  - High energy efficiency

3.2 System design

In order to identify mosquito populations the researchers designed a system that uses microphones to record audio samples and processes that information locally using a single-board computer such as the Raspberry Pi 3 or Intel Edison.

From there the information is passed through what the researchers describe as a highly optimized algorithm which identifies the fundamental frequency of the audio sample. The fundamental frequency data is then compared to the known data in order to identify whether or not the audio sample holds mosquito like sounds. I will later detail how this computation happens.

Now we will identify the technologies involved in the system and how those fit in our description of the three layers of IoT.

- Perception
  - Microphones installed on single board computers
- Network
  - Various methods were considered like RFID, NFC, Bluetooth, and WiFi
    - The researchers had to consider the frequency and range capability of each of these technologies
    - WiFi with mesh topology (see figure 2.)
- Application
  - Single board computers, central server node
    - Raspberry Pi 3 - majority of computation
    - Intel Edison - classification

Data is collected by the microphones installed on the single board computers. This design helps reach the goal of having low cost per device. Once the information is collected, the majority of computation happens on the single board computer.

In order to more efficiently process this data the researchers took advantage of software optimizations. For example instead of sending the information in a continuous stream they used techniques known as batching and dallying. Batching is when a computer program collects lots of work to do at once. Doing this shares computational overhead over a larger set of work. This works in two ways, the devices are able to sleep until they have enough data to compute and they don't have to send data as often. They instead send data in chunks when not asleep. The other technique, dallying, is similar to this principle. Dallying is when a computer program waits to do something. In this case dallying lets us then use batching to more efficiently run this system. Use of both of these techniques allows the system to have a better throughput per energy unit. Throughput is the number of requests or computations that can be processed in a fixed amount of time.

After the computation occurs, that data is then passed over a WiFi mesh network to a central server node where the classification and storage of finally computed data would then occur. The researchers chose to do this as it helps reach the goals of remote deployment, as the devices are able to communicate while being spread out, and somewhat resilient in the case of single unit failure. The reason for this is the use of a mesh network topology. If we refer back to the topology figure, Figure 2, we can imagine that each device shown could be a single single board computer. The interconnections between all or most devices provide a wider range for communication and help alleviate the burden of

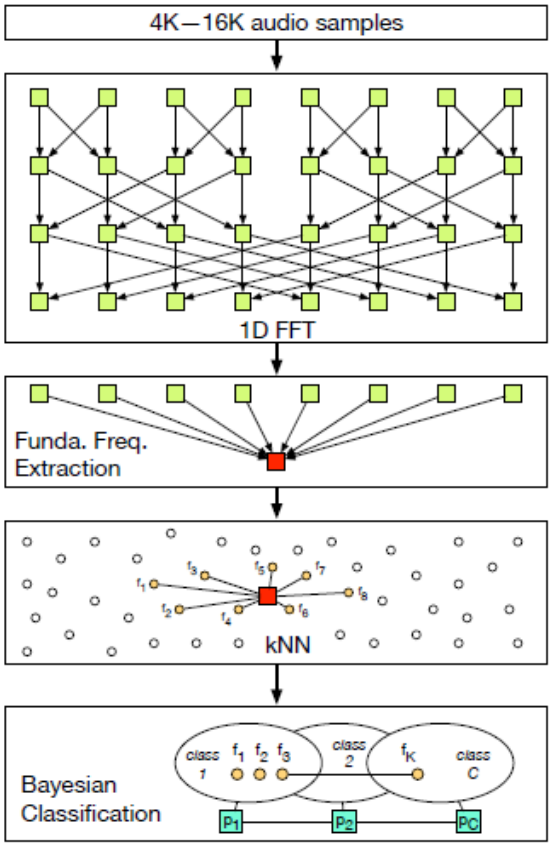

Figure 3: A high level view of computation [6]

single device failure. If a single device fails, it is likely that other devices will still be available to act as the channel for communication.

3.3 How the information is processed

While the researchers explain the process in detail, I will only be going in depth with one of the sections of the computation as that is where almost 90% of the time is spent when the system is used.

We can think of this from a high level view as seen in Figure 3. The process of computation runs the steps as follows:

- Samples are recorded
- Samples are passed through a 1D FFT (Fast Fourier Transform) and the fundamental frequency is then extracted
- That information is taken and compared to training data using k-Nearest Neighbor (kNN) algorithm
- Finally a Bayesian classifier is used to categorize the information

While I would like to talk about this process in depth the only part that I will be going in depth with will be the FFT and fundamental frequency extraction as I said before around 90% of computation time is in these steps.

3.3.1 FFT and fundamental frequency extraction

FFT stands for Fast Fourier Transform. This can be thought of as if we were given a smoothie, we would then be able to identify what ingredients and in what amount they were added to the smoothie. This process happens by passing the information through a series of filters. This is all done in order to help with analyzing, comparing, and modifying the original data. This makes sense in the context of mosquito detection and fundamental frequency extraction as the original sample audio will possibly contain interfering audio. We must break this audio down into its parts to identify the unique sound produced by mosquitoes from the original data. We can also think of FFT as taking a time-based pattern, measuring every possible cycle, and returning the overall cycle recipe (the amplitude, offset, and rotation speed for every cycle that was found) [1]. With this information we are able to pass it to the kNN to find similar information from the training data and then passing that to the Bayesian classifier for final categorization. This computation benefits from the use of parallel computing and automatic vectorization.

If we look at FFT, it can be represented as:

$$X(k) = \sum_{n=0}^{N-1} x(n).e^{-j(2\pi/N)nk}; k = 0, 1, ..., N-1[6] \quad (1)$$

A simple explanation of this equation would be that the left output is the frequency recipe, amplitude and phase, a complex number, which would need to be converted using rounding from being a complex number into an integer in order to be processed by a computer. The summation adds everything up, taking into account each contributing part to of the frequency, in order to have a single output frequency value, the amplitude and phase. This brings us to the difficult part, the right side.

Equation 1:
N = number of time samples we have
n = current sample we're considering (0 ... N-1)
x(n) = value of the signal at time n
k = current frequency we're considering (0 Hertz up to N-1 Hertz) [1]

I will be providing a link to a resource which thoroughly explains this portion as the details are out of the scope of this paper. What we need to understand though is that it determines what contributions to the original sample frequency on a timed interval [1].

3.3.2 Other steps: kNN and Bayesian classifier

The process of classifying frequencies, comparing the computed frequency versus test data, as mosquito or non-mosquito sounds takes place on a central server. The researchers utilized kNN and a Bayesian classifier to process and categorize the data. These processes, together, made up about 10% of the computation time, so I will not be going in depth with my description. The kNN (k Nearest Neighbor) step takes the information which has had the fundamental frequency extracted and compares it to a test dataset. The dataset contains a thousand audio samples per species for a total of three thousand samples. The original audio files provided are used for training the classification models. A subset of these samples is used for testing and evaluating prediction accuracy [6]. Once similar samples are found, it then passes the sample onto the Bayesian classifier. The Bayesian classifier was selected over other probabilistic classification methods as it performs well at minimizing the probability of misclassification. It also is stated as being CPU and memory conservative which both help with reaching the optimization goals of this system.

3.4 Optimizations

The main aspect of this study focuses on optimization, with the goal of being able to deploy the mosquito detection system in remote places for long periods of time. This system takes advantage of hardware acceleration, precision reduction, and compiler optimizations all in an attempt to make the system more efficient both in terms of computation and energy consumption.

Hardware accelerators help most during the bulk of computation time in this system. As I stated before around 80-90% of the runtime of the code occurs in the FFT step. This step benefits from the use of hardware accelerator blocks made available due to the different devices tested. The researchers found that use of hardware acceleration over traditional CPU computing lead to up to a 2 times speedup in computation. The researchers were also able to take advantage of parallel computing given the multiple cores available on the Intel Edison device. This too led to further performance improvements as now multiple audio samples are able to be processed at the same time instead of serially [6].

Computing data using these methods on the single-board computers rather than sending the raw data directly to the central server greatly reduced the size of data needed to transmit cutting the energy cost required to send data.

3.5 Research findings

Ravi et al. decided against sending data to a central server as sending raw data over radio communication using Bluetooth or WiFi on a radio-capable board was costly to energy usage. The devices were powered by a 2000 mAh AA battery and under these conditions of:

- 8K samples of audio data
- Particle Photon board (Single-board computer)
- Broadcom BCM43362 WiFi chip

resulted in only 20 hours of battery life under continuous use with 300 - 400 mJ of electricity expended per transmission.

Based on these findings they decided to try a different approach. Instead of using cloud computing they found performing lightweight computation on the embedded board locally reduced the communication bandwidth needs and lowered overall energy use.

Using an optimized implementation of the algorithm [6] on an Intel Edison platform requires 5 ms of compute time and under 5 mJ of energy. Approximately 80x reduction in best-case energy use stretching the battery life to around 2 months. The devices now could be powered for long periods of time over a battery and be deployed to remote, hard-to-service locations such as swamps, gutters, and large construction sites.

The researchers were able to design a possible solution to detect mosquito populations more effectively than current methods. They were able to detect the presence of mosquitoes in a particular area where the system is deployed. They were able to detect the presence of mosquitoes with about an 80% accuracy.

4. RESILIENT SMART HOME

4.1 Description

The second study [4] focuses on providing stability for essential services in a smart home which is controlled and powered by cloud computing services. Researchers identified which services a smart home provides and which they consider essential. Based on the identified services the researchers designed a solution called RES-Hub. RES-Hub was designed to provide a cloud neutral, meaning it is able to function without cloud access, solution for smart homes. It was designed to be able to detect the loss of cloud computing service. At that point it should be able to notify the user or system in order to allow a reaction to prevent further issues caused by service loss. It should also be able to provide an alternative interaction channel in order to allow the user to continue to interact with smart home devices when cloud services are down. It should change the security settings of devices as they may now be vulnerable to attacks. It should transfer the computational service from the cloud to a local or alternative unit. All of these means that the device should be able to run a smart home independent of cloud service connection, in a reasonable but with limited feature capacity.

Now we will identify the technologies involved in the system and how those fit in our description of the three layers of IoT. It is worth keeping in mind that this study is more generalized than the first so it will be more loosely defined.

- Perception
  - Various: speakers, home hubs, light bulbs, thermostats, security cameras, switches and plugs, locks, smoke detectors, and many more
- Network
  - Various: typically Ethernet, WiFi, or Bluetooth
- Application
  - Cloud computing
  - Local computing

With the wide variety of perception layer devices we must consider the priority these devices hold. Since loss of connection to cloud services may occur it is necessary to maintain essential functionality with some of these devices prioritized both by necessity and by the computational ease of data produced by the device. For some devices, it is more costly to maintain full service when cloud computing is not available. This can be due to a variety of reasons, such as the type of data which the device collects could be difficult to compute, to the point that RES-Hub may not be able to take over these computations. We also must consider the risks involved when certain parts of the system are unavailable when prioritizing. For instance, we would like our locks to function in case of a fire. If cloud services are down and the doors are locked, we would like RES-Hub to be able to automatically unlock the doors or, at least not prevent manual unlocking.

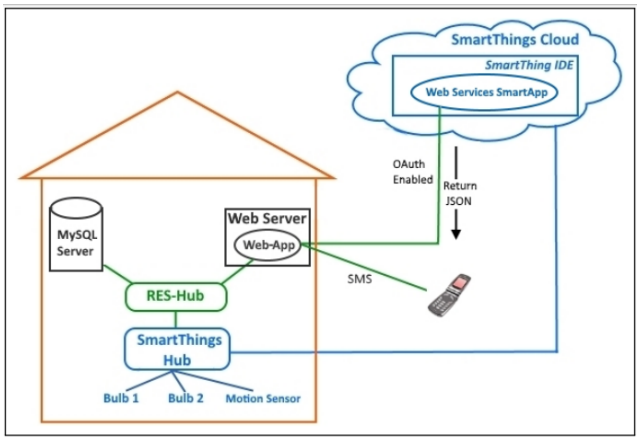

Figure 4: Resource state synchronization and Notification modules [4]

I believe this system best follows a tree network topology. If we refer back to the a previous figure, Figure 2, we can think of the RES-Hub device being the central device with each perception device being the "leaves" of the tree. Each RES-Hub node can represent a different home, which all connect to a "root". This "root" we can think of as the Internet. This information would then be transferred through the RES-Hub before being connected to a cloud computing service. If the connection to the "root" is lost each tree can act self contained, albeit to a lesser extent as computational power would typically be lessened.

4.2 System Design

When designing a smart home it is worth noting that any layer of the IoT system may fail. Sensors can break or disconnect, the network could go down for a variety of reasons, or cloud service may become unavailable. These services can be lost for a variety of reasons like technical problems, natural causes, scheduled maintenance, or even targeted attacks. To solve this issue the researchers designed RES-Hub. It provides users with a local backup service in case of cloud service loss.

RES-Hub is made of two modules which can be seen in Figure 4. The first part is a resource state synchronization module which detects the state of devices from the cloud at regular intervals. If a response is not received it can be assumed that the cloud computing service is not available. The second and final part is the notification module. This module contacts the smart home owner via SMS, also known as text messaging.

The researchers were able to design a system which provides a cloud neutral solution for smart homes in case connection to the cloud computing service is lost. They were able to do this with the use of RES-Hub.

5. CONCLUSION

We now know that each system has different requirements and goals.

- Mosquito case
  - Optimization and energy efficiency to allow remote deployment and less costly maintenance
- Resilient smart home case
  - Secure and resilient to allow for stable services in the home in the event of cloud computing or network outage

Both of these systems needed to be cloud independent for different reasons defined by the other goals and requirements of the system.

We now know that each layer (perception, network, and application) can be modified to better serve the function of an IoT system. These layers are modular. We must consider each when designing around system requirements and goals.

While these are only two examples of IoT systems, the goal and requirement based layer analysis which this paper used can be applied whenever analyzing or designing an IoT system.

Acknowledgments

I would like to thank professors Lamberty and Machkasova for there continuous help and feedback throughout this course. I would like to thank the other professors in the computer science faculty as well for giving me the knowledge needed to complete this project.

I would also like to thank my friends and family who have supported me through the process of my Senior Seminar. I would like to also thank University of Minnesota Morris alumnus Justin Mullin for providing useful feedback on this paper.

6. REFERENCES

[1] An interactive guide to the fourier transform. https://betterexplained.com/articles/an-interactive-guide-to-the-fourier-transform/ (visited on 2019-05-3).

[2] What is a network?, Nov 2018. https://www.computerhope.com/jargon/n/network.htm (visited on 2019-05-3).

[3] Architectures in the IoT civilization, Mar 2019. https://www.netburner.com/learn/architectural-frameworks-in-the-iot-civilization/ (visited on 2019-05-3).

[4] T. T. Doan, R. Safavi-Naini, S. Li, S. Avizheh, M. V. K., and P. W. L. Fong. Towards a resilient smart home. In *Proceedings of the 2018 Workshop on IoT Security and Privacy*, IoT S&P '18, pages 15–21, New York, NY, USA, 2018. ACM.

[5] T. Primya, G. Kanagaraj, V. Suresh, and G. Selvapriya. A survey paper on various computing to emerge cloud computing, Oct 2016.

[6] P. Ravi, U. Syam, and N. Kapre. Preventive detection of mosquito populations using embedded machine learning on low power IoT platforms. In *Proceedings of the 7th Annual Symposium on Computing for Development*, ACM DEV '16, pages 3:1–3:10, New York, NY, USA, 2016. ACM.