CSCI 572: Computer Networks II (Fall 2021)

Final Project: Network Plant Moisture Sensor

Kyle Hansen
Michael Humphrey

December 8, 2021

1 Introduction

Internet of Things (IoT) devices connect wirelessly to a network and send data over wireless protocols. There has recently been an increase in research on IoT devices regarding their efficacy and how they compare to traditional wired devices that communicate over a cable to the internet. This research will implement a trivial IoT device and analyze how the device interacts with the internet. The IoT device that will be implemented is a plant moisture sensor that communicates wirelessly to a server which will notify a user when their plant needs to be watered via android smartphone. This project was simple enough that it was feasible to build given the time constraints of this research, but is complex enough that valid analytics will be possible to obtain from constant usage of the device through different networks.

The project consisted of three major components. The first component being the sensor system, which consisted of a series of sensors which transmit data continuously to a microcomputer, which was used to interface to the internet. The second component was a server, which received requests from the microcomputer and established a wireless connection to an android smartphone in order to display updates received from the system. The final component was the android mobile application which received the data from the server, the packets from the server were decoded and information such as plant status, packet latency and packet loss were displayed.

Each component was developed separately with the intent to connect to one another as development progressed. As such, each component was tested thoroughly before attempting to interface with any other components. Using this development style, the final product needed minimal unit testing as each component was confirmed to be correct before being combined into the final product.

The materials used throughout the project are listed below:

- Raspberry Pi 3
- Raspberry Pi Moisture Sensor
- Android Smartphone

The metrics used to test within IoT devices are:

- Latency between plant moisture sensor and smartphone
- Packet loss and its effects on viable data
- Whether adding multiple sensors fixes the problem of packet loss

Each metric was measured on a variety of different networks which were used to induce real-world difficulties that a typical end-user may experience when using similar IoT devices.

To test the latency and packet loss, the product was tested on:

- Very fast network (Symmetric Gigabit)
- Very slow network (8 Mbps download, 2Mbps upload)
- Passive internet (few users, no major data being transferred)
- Active internet (many users, major data being transferred)

This project was implemented first on a local network in order to ensure all functionality and assert that the IoT device was working properly. this local network was the control for the data that is obtained in the methods described above. It was the data that each other test will be compared to in order to determine whether or not each factor largely contributes to the communication of IoT devices to end-users.

2 Related work

Some different projects related to plant moisture sensor have been completed and shared. These include basic hard wired notifications [1] as well as moisture sensors that send emails at specific moisture levels [2].

A basic hard wired moisture sensors consists simply of the moisture sensor, a Raspberry Pi and a computer. The Raspberry Pi is used to interface with both the computer and the moisture sensor. It sends a voltage to the moisture sensor and if the moisture level is below a specific level then the moisture sensor will return a voltage to the Raspberry Pi, if the moisture level is above the specified threshold then the sensor does not return a voltage. Based on the output from the moisture sensor, the Raspberry Pi then reports either the moisture level or that the moisture level is above or below a certain threshold.

A more in depth approach to this project that has been done before is one that includes email notifications based on moisture levels. Once again, the Raspberry Pi is configured to power the moisture sensor and receive a signal back from the moisture sensor letting it know whether the moisture sensor is reading high enough moisture levels or not. The Raspberry Pi is then loaded with a script that allows it to send an email when triggered. This script sets up an email account that it will be sending the email from as well as an email account that it will be sending the email to and the message to be sent. It then uses the host and port of a server to set up and login to the email followed by sending the message from the sender to the receiver. Finally the script contains a callback that is called whenever the moisture level threshold is crossed which is used to call the function that sends the email.

Our solution draws some information off of these related works such as the hardware implementation, however, in terms of networking and user interface it is much more in depth. Our solution will work in real time, immediately updating a user over the internet whenever the moisture levels cross the threshold. We will be creating a server that allows for multiple sensors to notify a user via an application that a plant does or does not have enough water. We will also be performing an analysis on our IoT devices to better understand their performance on different networks. Our goal is to learn how latency and packet loss can affect a system of devices and if these hurdles can cause issues in a simple IoT setup.

A technical research paper titled "Delay Analysis in IoT Sensor Networks" also looks into the end to end delays and latency of IoT sensor networks in order to better understand how IoT sensors may be latency sensitive as well as to try to make user experiences that may depend on these devices more seamless and optimal. The paper used a two-stage tandem queuing model which provided a way to estimate end-to-end latency and predict latency variation. It also introduced a feedback mechanism to find network utilization, optimal number of computing units in a cluster and other networking performance metrics. It was found that reducing queuing delays can greatly help reduce tail latency, queuing effects increase linearly with utilization and tail latency can be kept low even with a busy server if scheduling is utilized [4].

3 Approach

Our approach consisted of testing the efficiency if IoT devices over a series of networks with the intent of determining whether different types of networks (fast/slow, active/passive) affect the communication of IoT devices with the network. Primarily, this was done through a simple IoT device – a plant moisture sensor, which we will test on each type of network. As the project was broken into three major segments, we will discuss each further below:

- Hardware Sensor/Microcomputer:
  Our hardware sensor was the LM393 Plant Moisture Sensor. The microcomputer was a Raspberry Pi 3, which was used to create a simple client which interfaced with the sensor as well as the server. The client for the microprocessor was written in C, and conforms to best practices of networking in C. The client's primary tasks were to collect data from the sensors attached to it, and send this data to a static server. Once the obtained from the GPIO pins of the Pi, it would create a pseudoo-packet that would be sent to the server. This packet consisted of a timestamp (used to determine latency), a sequence number (used to determine packet loss), a plant number (used to determine which plant the message was for) and a message for the server to decode.

- Server:
  Our server was deployed first on a local network, followed by running on Isengard to be able to access the server over the internet. It maintained data about the sensor system, as well as the latency of all operations and the data it received was sends out to android devices. It was written in C as well, and conforms to best practices of networking in C. The server's primary task is to store all devices connected to it, allow more to connect to it, and forward any data sent from a Raspberry Pi to the Android devices connected to it. The server was built to be multi-threaded to allow an arbitrary number of devices to connect to it. The devices connected to it were stored in a list, each with an identifier indicating whether it was a Raspberry Pi or an Android device.

- Android Application:
  Our front end of the project was an Android Application which interfaced with the server. It was written in Java, and was simple to use, simply connecting to the server, and displaying updates to the user. The application was expanded to be able to handle multiple plants, from multiple different connections. The Android application has three main purposes: decode the information sent to it by the server, perform any necessary computations on this data, and display it to the user. Decoding the data was the largest challenge, as it had to be read in as a byte array, then separated into its multiple different fields. After this, the application would compute the end-to-end latency by obtaining the current time, and subtracting the timestamp from the packet from the current time. It would check for packet loss by comparing its expected sequence number to that sent by the server. If they matched, no packet loss, otherwise this is the

wrong packet, and some packet has been dropped. After these computations, the necessary data is displayed to the user in a simple manner. The user can go between any of the plants registered to it, and view messages from the Raspberry Pi, as well as track the latency.

The following figure illustrates the networking that occurred to get the data from the Raspberry Pis to the Android devices.

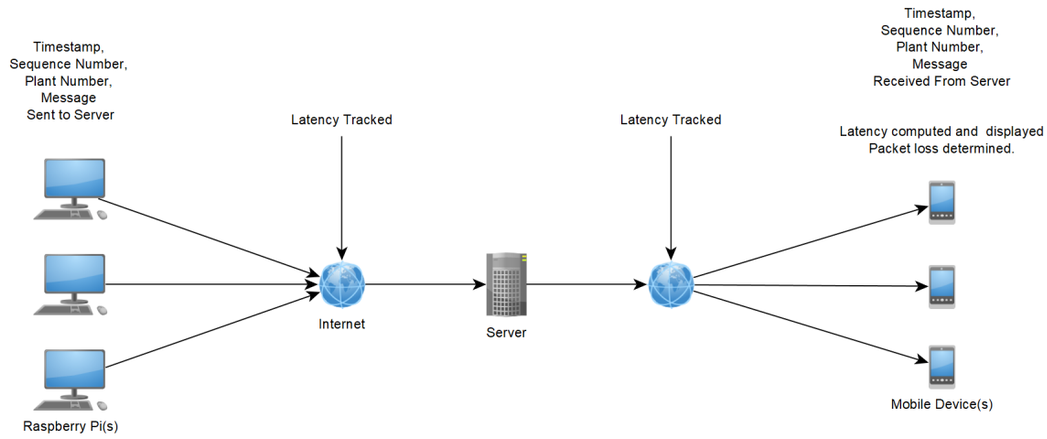

Figure 1: Layout of Networking

Once all systems and applications were in place we proceeded to perform testing. We first tested completion and correctness by observing and evaluating all packets sent from the Pi, received and sent by the server as well as received by the android device. This was followed by testing server robustness to make sure that outside devices, namely the Pi and Android devices, would not interrupt or crash the server. Finally we tested the end-to-end packet latency and packet loss on multiple different networks. The data observed and evaluations made will be discussed in the Performance Evaluation section.

4 Performance Evaluation

In order to test the performance of our setup, we decided to test it on several different networks with different upload/download speeds, as well as differing workloads on each network. Each network type listed is below:

- Fast Network with little to no workload (passive network)
- Slow Network with little to no workload (passive network)
- Fast Network with a high workload (active network)
- Slow Network with a high workload (active network)

For the sake of this project, we consider a fast network as one with a symmetric Gigabit Ethernet fiber-optic connection, and a slow network as one with a 2Mbps download speed and 0.5Mbps upload speed, running on broadband WiFi.

Additionally, in the scope of this project, we consider a network with no workload – or a passive network – as one that is relatively dormant, not transmitting large

amounts of packets, with few devices connected, and a network with a high workload – or an active network – as one that is actively downloading and uploading large files while running the software for the IoT device.

Our testing protocol was relatively simple. The network type that we were trying to test on was created (e.g. for the busy networks, downloads were started just before testing). Once the network type was created, the server program was started which would forward packets to the android devices. After the server was running, the Raspberry Pi was started, and began to send packets to the server. Subsequently, multiple android devices would be connected to the server to test with. The Raspberry Pi sent information every 5 seconds. We recognize that if this were to be implemented in a real-world product, the time between sending information would likely be much longer than every 5 seconds, but for testing, this was sufficient. Once the Raspberry Pi sends data to the server, the server would forward the information to each android mobile device that was also connected to the server. Once the android devices received the packets from the server, they would decode the information contained within them, and display it on screen. The latency and whether or not packet loss occurred was noted, and logged for evaluation.

We do acknowledge that our method of testing is not perfect. The networks that were tested on were partially uncontrolled. Other devices or processes may have been occupying the network during our testing, as it was done on real-world networks, and not on a controlled or simulated environment. Similarly, irregularities may have been introduced within each router that the internet packets travelled to. Again, as this was tested on a real-world network, this is unavoidable, and will always induce some bias into the data.

As such, the results obtained are not perfect; some of them are skewed heavily, and do not have a precise trend. Below is a summary of the latencies obtained throughout our testing.

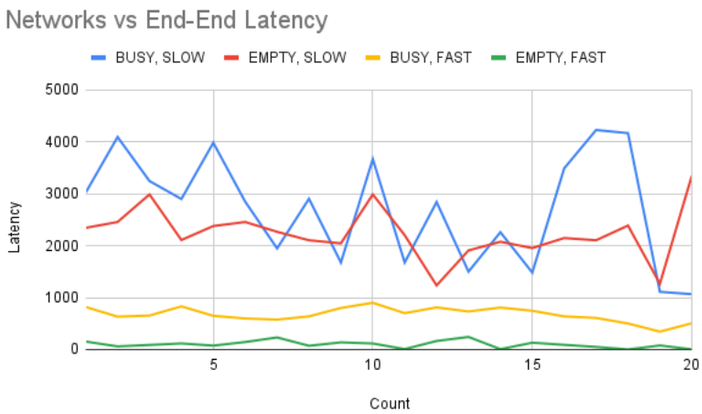

Figure 2: Latencies Per Network Type

There are a few conclusive evaluations that can be made from the above figure. First, the latencies for each network matches what would logically be expected. That is, the fast passive network had the lowest latency, followed by the fast, active network, and then the slow, passive network, and finally, the highest latency network was the slow, active network.

Beyond this, however, there are a few distinctions that can be made for each network. First, the latencies for the fast network were much more consistent, and linear on the graph. this could be because the packets were able to travel directly over the network from sender to receiver with no queueing delay. This could also be because the fast

network was much more in speed and did not have the unpredictable behavior that one can find with broadband internet.

Additionally, For the slow network, whether passive or active, the latency of the data being transmitted varied wildly from one transmission to another. This could be due to the fact that the slow network introduces queueing delays for packets, or simply because a broadband network is less predictable and more prone to irregularities.

Throughout all testing conducted, we did not come across packet loss in any transmissions between Raspberry Pi and server or between Server and Androd device. This is likely because the network protocol used to transmit the data was TCP, with its guaranteed and in-order delivery. this means that there was no way that the data would not be transmitted. Future work could include transmitting the data using a lossy protocol such as UDP to test how packet loss is affected. The software is configured to handle this, but it has not yet been tested with UDP packets. This will be discussed more in the future work section of this paper.

5 Future Work

As this project was implemented in a relatively short timespan, there is a large amount of work that could potentially be done to build upon and improve this project.

First, work could be done to make this project more user-friendly and easy to use for a typical end-user. This could include spending time on the front-end of the application, and make it nicer to use, as well as removing the need to explicitly connect to the server every time a user opens the application.

Additionally, work could be done to turn this project into a product that real users could use. This would involve creating a database in which certain microcomputers are mapped to certain android devices, allowing for many disjoint users.

Furthermore, this project could test its implementation on other network protocols, and determine which protocol performs the best in terms of latency as well as packet loss for protocols prone to packet loss.

Finally, work could be done to test how this product would perform if data was encrypted and sent over a secure connection. It could be determined how much this affects the performance as well as the packet loss of transmissions.

6 Conclusion

The results and evaluation that was collected with this project was highly conclusive. We managed to build an Internet of Things device that may be used in real-world scenarios, and could be a marketable product given a few tweaks to make it more user-friendly. We found that the network speed and current activity played a huge role in the latency that packets took to travel end-to-end between Raspberry Pi to the server and finally to android devices. We found that a fast network is less latent than a slow network, and that a passive network will perform better than a busy one. This means that any future IoT devices can reasonably predict the latency of end-to-end transmissions involving microcomputers.

References

[1] Kumar, R., Doe, J., & Nguyen, B. (2018, June 24). Measuring soil moisture with the Raspberry Pi. Raspberry Pi Tutorials. Retrieved October 11, 2021, from https://tutorials-raspberrypi.com/measuring-soil-moisture-with-raspberry-pi/

[2] Raspberry pi plant pot moisture sensor with email notification tutorial. The Pi Hut. (2017, March 8). Retrieved October 11, 2021, from https://thepihut.com/blogs/raspberry-pi-tutorials/raspberry-pi-plant-pot-moisture-sensor-with-email-notification-tutorial

[3] Tutorial - using capacitive soil moisture sensors on the Raspberry Pi. SwitchDoc Labs Blog. (2020, July 22). Retrieved October 11, 2021, from https://www.switchdoc.com/2020/06/tutorial-capacitive-moisture-sensor-grove/

[4] A. Althoubi, R. Alshahrani, and H. Peyravi, "Delay Analysis in IoT Sensor Networks," Sensors, vol. 21, no. 11, p. 3876, 2021.