Time Synchronization in Cellular IoT Network

Kyle Belanger, Owen Raftery, Sasha Shikhanovich, Thomas Murphy
University of Massachusetts Amherst, College of Engineering

Abstract
In this project, a Sixfab Pico LTE was used in attempt to connect to a Network Time Protocol (NTP) server over a 4G LTE connection. The purpose of this demonstration was to sync the time on the board with the NTP server's time and observe a drift over the course of multiple hours. Attempts to complete this project over 4G LTE were unsuccessful, but by utilizing the built-in Raspberry Pico W's ability to connect to WiFi some significant progress was able to be made. A connection was made to multiple different NTP servers with a successful time synchronization and a proper drift was observed.

Contents

1 Introduction

In this project, the group was provided with a Sixfab Pico LTE. This microcontroller is a Raspberry Pico W that has the ability to connect to a 4G LTE network using an embedded SIM card. The goal of this project was to use this board to establish a connection with a Network Time Protocol (NTP) server over 4G LTE. This connection was then to be used to sync the board's internal clock. This action would allow the group to observe a drift in time between the board's clock and the time that is returned from a NTP server. Throughout this report, various references will be made to files, all of which can be found in the group's Github repository that contains all of the information relating to this project.

2 Getting Started

This section outlines the steps that the group took to get started with the Sixfab Pico LTE.

2.1 Activating the SIM Card

Getting started with the Sixfab Pico LTE is a little bit different than using some other boards with 4G LTE capability. The Pico has an embedded SIM card, which means that in order establish a connection you need to activate the card on the Sixfab Website us-

ing the provided account information. Once you have made sure the board is connected, there is only more step to do before interacting through 4G LTE. Before any connections can be made, the LTE module on the board needs to be activated. This can be done by pressing the `PWRKEY` button, after which a blue light should blink on the board. At this point, the Sixfab Pico LTE is ready to use with a 4G LTE connection.

2.2 Setting Up the Board

The next steps to using the SixFab Pico LTE can be done by following the tutorial provided on the Sixfab Website. These steps include installing the Pico LTE SDK and the recommended Thonny IDE to interact with the board. After setting up the board using these instructions, the group first ran the `boardBlink.py` file to ensure that the board was properly setup and working.

2.3 Connecting to the Internet

At this point in the project, the board is ready to connect to the internet and perform simple requests. The first requests that the group performed were HTTP GET, PUT, and POST requests using webhook.site. These requests can be made by following a Sixfab tutorial and using a built-in PicoLTE micropython module that came in the SDK installed previously.

3 NTP Over a 4G LTE Connection

This section outlines the group's attempts to connect to a NTP server over a 4G LTE connection.

3.1 Using AT Commands

At this point in the project, the group is ready to start connecting to a NTP server over an LTE connection. After doing some research, the group discovered that connection to a NTP server requires a User Datagram Protocol (UDP) connection. Since there is the PicoLTE module does not have functions supporting this protocol, a possible workaround was discovered using AT commands. AT commands can be used to establish and configure a connection by utilizing the `send_at_comm()` function in the PicoLTE module. Examples of the group's attempts to do this can be seen in the networkTesting folder.

But, attempts at using these AT commands were not as fruitful as expected. Creating a UDP socket using AT commands seemed rather impossible, much less trying to connect to a NTP server. Other micropython modules (such as `socket` and `ntptime`) were used in attempt to create a UDP socket and connect to a NTP server, but those also yielded poor results.

While continuing to test AT commands and python modules on the board, it was discovered that the Sixfab Pico LTE was automatically syncing it's time with the network upon booting. From here, the group was able to use AT commands to create and confirm a network connection, as well as pull the time from the board's 4G network. An example of this can be seen in `SyncTimeWithAT.py` in the networkTesting folder.

3.2 Issues

Although the time was able to be gathered from the network using AT commands, there were a large problem. The time that was pulled from the network was only accurate up to a second of precision, which is a larger than what would be needed for this project. The overarching reason for this problem was because time was being pulled directly from the

4G LTE network, not an actual NTP server. NTP servers generally return time with a millisecond precision, whereas the network only provides second precision. Clearly, in the context of this project, receiving the time from a NTP server is much more favorable.

At this point, the group was stuck in the project after exhausting all available resources. At this point, we were given the green light to attempt the same project over WiFi instead of a 4G LTE connection.

4 NTP Over a WiFi Connection

This section outlines the group's attempts to connect to a NTP server over a WiFi connection.

4.1 Switching Focus

After switching over to using a WiFi connection, many resources were available for configuring a connection to a NTP server. Due to the popularity of the Raspberry Pico W, which the Sixfab Pico LTE is built around, it was easy to find projects others had done in the past that the group was able to build off.

Utilizing code from aallan on Github, the group was quickly able to connect to an NTP server. aallan's original code was designed to set the Pico's clock the the time returned by a NTP server with millisecond precision, which was perfect for the purposes of this project.

4.2 Getting Results

Re-purposing aallan's code, the group was able to write `ntp_utils.py`. The functions in this file were used to do three things: connect to a NTP server, set the board's time, and request the time from the server. These functions were then used in `ntp_drift.py` to record the drift observed over time.

After all of the code was written and tested on the board, `ntp_drift.py` was run twice using two different NTP servers, each for a little more than 14 hours to gather enough data to show a significant drift.

5 Results

This section outlines the shows and explains the group's results.

5.1 Observing Drift

After gathering data on the drift that happens between the Pico's internal clock and the NTP server, it was seen that the drift can grow to be quite large over time. After running for a little more than 14 hours, the drift between the board and the time1.google.com NTP serer was roughly 51 seconds.

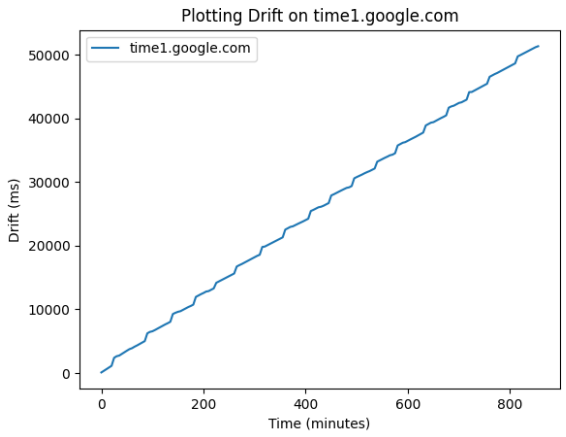

Figure 1: Observed drift from time1.google.com.

The same result was also produced using the pool.ntp.org server.

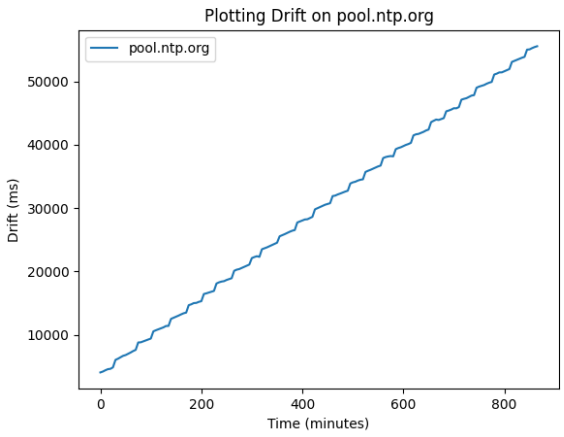

Figure 2: Observed drift from pool.ntp.org.

As seen in both figures, the drift observed over time in these trials appears to follow a linear pattern, growing larger and larger as more time passes.

6 Conclusions

In this project, the group was able to successfully connect to multiple NTP servers and receive information about the time with millisecond precision. This information was used to sync the internal clock of a Sixfab Pico LTE microcontroller and then used to observe the time drift between the Pico's clock and the real time from the NTP server. The observed drift from all tested servers appeared to grow linearly with time. After roughly 14 hours spent calculating the drift, both experiments resulted in a roughly 51 second drift between the Pico and the NTP server.

This project displays the importance of regularly syncing the time between devices and an NTP server to ensure that a device does not become wildly out-of-sync with the real time. Examples of cyber security threats that can occur due to a system not being in-sync can be found in the "NTP Time Errors Leading to Security Threats" report posted in the Github repository.