# Team 4 - Precision Time Protocol (PTP)

Completed by: Jachob Dolak

## Plan

The goal of this portion of the project was to learn about Precision Time Protocol (PTP) and implement it on NDFEX.
PTP is a clock synchronization protocol similar to NTP, but with much greater precision.
This is achieved by using hardware timestamping in NICs, allowing for possible sub-100 ns precision between clocks on a network.

This is useful for NDFEX because it allows timestamps between trading machines and the exchange to mirror each other as tightly as possible.
Because HFT operates in the scale of nano and micro seconds, NTP is too imprecise. Integrating PTP will benefit logging, performance evaluation, and debugging.

## Completed

I was able to successfully configure PTP on NDFEX using the standard Linux implementation (ptp4l + phc2sys), with the exchange as grandmaster and hftt0 as an ordinary clock.

However, after setting it up, I stumbled across a solarflare specific implementation of PTP called sfptp.
I successfully enabled it and decided to evaluate each implementation in terms of its self reported offset between the PHC clock and the system clock, and the offset between the grandmaster and the ordinary clock.

The data was collected by analyzing the logs that contain the offset value and plotting them in a Jupyter notebook.
The data was collected after letting each implementation stabilize for 30 mins, and then collecting data for 10 mins.
However, a window of 6 mins of data was used for the graphs to allow for legibility.

## Challenges

There's no fun without challenges, and this was no exception.

The first challenge was the lack of a highly accurate and precise GM clock, like a GPS clock.
This meant that I had to make the decision of syncing the GMs PHC to its system clock (granting accuracy but less precision) or vice versa (precision, less accuracy).
The obvious choice was to sync the system clock to the PHC, because "time is relative" and as long as all the clocks in the lab agree, then accuracy doesn't matter for our performance evaluations.
However, the PHC thought it was currently 1970.
So, the solution was to initially sync the PHC to the system (to at least get the date right) and then swap it to get the benefits of precision.

Secondly, ptp4l and phc2sys needed to run in the background, but the apt package for them didn't come with systemd services.
So, I had to make them.
I surprisingly had never manually made a systemd service before, so this was pretty cool to learn.
It was straight forward after doing some research.

# Takeaways

The results from my data analysis were surprising.
It appeared that the clock offset of the standard Linux implementation for PTP had a lower root mean square than the solarflare one.
This was contrary to my expectation that the vendor specific program would perform better.
sfptp had an average PHC RMS of 25.595 ns while linuxptp had 18.22 ns, and sfptp had a GM offset of 33.89 ns while linuxptp had 17.79 ns.

However, there are some possibilities for the results:

- sfptp actually performs the same or better, but sfptp has access to more precise timestamps than linuxptp, so it paints a more accurate picture with its logs
- while linuxptp has a lower RMS, it swings more frequently and more violently. sfptp could be prioritizing a more stable clock with the sacrifice of some nanoseconds of accuracy.
- A difference of how often the implementations check and report their offset makes linuxptp appear better

Additionally, despite appearing to be less precise, I would still choose sfptp to implement in my lab. This is because sfptp was a single executable, had no necessary config file, and handled more of the setup than linuxptp. To me, it was "good enough" and was simpler to use.

In all, this was a fun project and I am glad I got to learn about PTP.