PTP time sync notes

Emil Fresk
https://github.com/korken89

Here are the notes for the PTP time sync.

1. Clock modeling

A clock can be represented as an autonomous system with a tick count T and tick rate $\dot{T}$. The tick rate of the clock, $\dot{T}$, is bounded to a nominal value that is scaled with an unknown error.

$$\dot{T} = (1 + \varepsilon)\dot{T}_{\text{nom}}$$

While ε is not known, it has known bounds from the clock's datasheet and is generally specified in parts per million (ppm). The characteristics of ε is generally temperature and voltage dependent, however we can model ε as a bounded random walk process for simulation purposes.

$$\dot{\varepsilon} = -\alpha\varepsilon + \mathcal{N}(0, \sigma^2)$$

This will give a clock that has a nominal tick rate but that slowly drifts around the nominal value. The nominal tick rate for the PTP peripheral in STM32 microcontrollers is 2^{32} which is accumulated in a 64-bit register, this means that overflows can be neglected as this would overflow about once every 136 years.

Combining the above gives the system's transfer function is as follows:

$$\begin{pmatrix} \dot{T} \\ \dot{\varepsilon} \end{pmatrix} = \begin{pmatrix} 0 & \dot{T}_{\text{nom}} \\ 0 & -\alpha \end{pmatrix} \begin{pmatrix} T \\ \varepsilon \end{pmatrix} + \begin{pmatrix} \dot{T}_{\text{nom}} \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \mathcal{N}(0, \sigma^2) \end{pmatrix}$$

For the clock on the micro processor we can control the deviation from nominal tick rate of the PTP clock, giving the following:

$$\begin{pmatrix} \dot{T} \\ \dot{\varepsilon} \end{pmatrix} = \begin{pmatrix} 0 & \dot{T}_{\text{nom}} \\ 0 & -\alpha \end{pmatrix} \begin{pmatrix} T \\ \varepsilon \end{pmatrix} + \begin{pmatrix} \dot{T}_{\text{nom}} + u \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \mathcal{N}(0, \sigma^2) \end{pmatrix}$$

It's important to understand that $\dot{T}_{\text{nom}}$ can differ between two clocks using the same crystal due to manufacturing tolerances. As an example, below is a simulation over the Brownian motion of two clocks that includes these manufacturing tolerances.

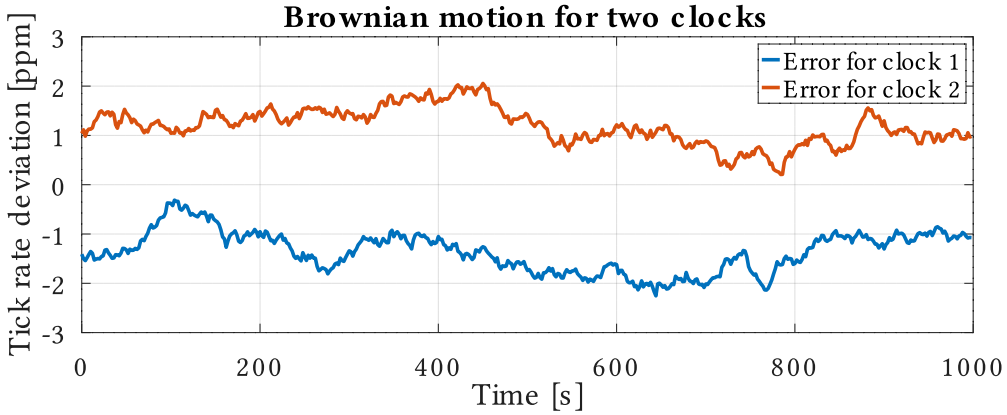

Figure 1: The simulated frequency error of two clocks.

2. Measurements

In PTP we have two clocks that we want to synchronize, one that we can control T^C, and some other autonomous clock T^A. While we can observe a timestamp from either clock we can't make an observation at the same time due to time delays to send timestamps and reading them out.

This is where the PTP peripheral in the MCU comes into play, through a set of network packet transfers we can measure the offset between the clocks as

$$y_m = T^C - T^A + w$$

where w is some measurement noise. This means that we need to find the dynamics model of the clock offset instead of individual clocks.

3. Offset modeling

From the offset

$$T^O = T^C - T^A$$

we can derive the corresponding dynamics by differentiating both sides

$$\dot{T}^O = \dot{T}^C - \dot{T}^A$$

and then simplifying based on the tick rate equation

$$\begin{aligned}\dot{T}^O &= \left(1 + \varepsilon^C\right)\dot{T}_{\text{nom}} + u - \left(1 + \varepsilon^A\right)\dot{T}_{\text{nom}} \\ \dot{T}^O &= \dot{T}_{\text{nom}}\left(\varepsilon^C - \varepsilon^A\right) + u.\end{aligned}$$

This shows, as both ε^C and ε^A are random walk processes, that the offset will drift over time if no control action is supplied from u. Moreover, the bound on the drift is double that of each individual clock.

4. Control aim

The main reason for offset drift is that the effects from the random walk processes cannot be eliminated, hence a controller is needed. Given that we can estimate T^O and $\dot{T}^O$, we can formulate a state feedback regulator that drives these states to 0.

That is, find a state feedback controller

$$u = -g \cdot \begin{pmatrix} T^O \\ \dot{T}^O \end{pmatrix}$$

such that

$$\begin{pmatrix} T^O \\ \dot{T}^O \end{pmatrix} \to \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

5. Estimating offset

We can estimate the current offset and its derivative using a Kalman filter, however the question is what model should we use for the estimator? As the underlying drivers for the random walk is mostly temperature, this means that changes in temperature will cause an increase or decrease in tick rate. Moreover we can assume that the change in tick rate is continuous. This indicates that either a velocity model or acceleration model should be a good fit for the problem.

Lets start with an acceleration model using the following state space model:

$$\begin{pmatrix} T^O \\ \dot{T}^O \\ \ddot{T}^O \end{pmatrix}_{k+1} = \begin{pmatrix} 1 & \Delta t & \Delta\frac{t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} T^O \\ \dot{T}^O \\ \ddot{T}^O \end{pmatrix}_{k} + \begin{pmatrix} 0 \\ u_k \\ 0 \end{pmatrix} + \mathcal{N}(0, \boldsymbol{Q})$$

$$y_k = (1 \;\; 0 \;\; 0) \begin{pmatrix} T^O \\ \dot{T}^O \\ \ddot{T}^O \end{pmatrix}_{k} + \mathcal{N}(0, R)$$

5.1. Simulation results (acceleration model)