What is the Time? Part 1

Call: +44 (0)23 8098 8890
E-mail: sales@itdev.co.uk (mailto:sales@itdev.co.uk)

Posted 4th December 2018, By James H (/company/about/team/james-h)

Introduction

In general, distributed devices need to agree on time in order to effectively coordinate tasks, but getting every one to tell the same time is challenging! This article, the first in a series of three, will introduce you to these challenges by discussing why many distributed clocks may not tell the same time.

What is the Time?

When someone asks you the time, you normally look at your watch and say something like "*It's twenty past twelve*". Unknown to you, the person who asked you that question, let's call him Bob, is trying to decide whether they should run to the bus stop to get the last bus home.

What you also didn't know was that at the exact same time that you read your watch and it said that the time was 12:20, the bus driver's watch read 12:25. From the bus driver's point of view, he is right on time and is already at Bob's bus stop.

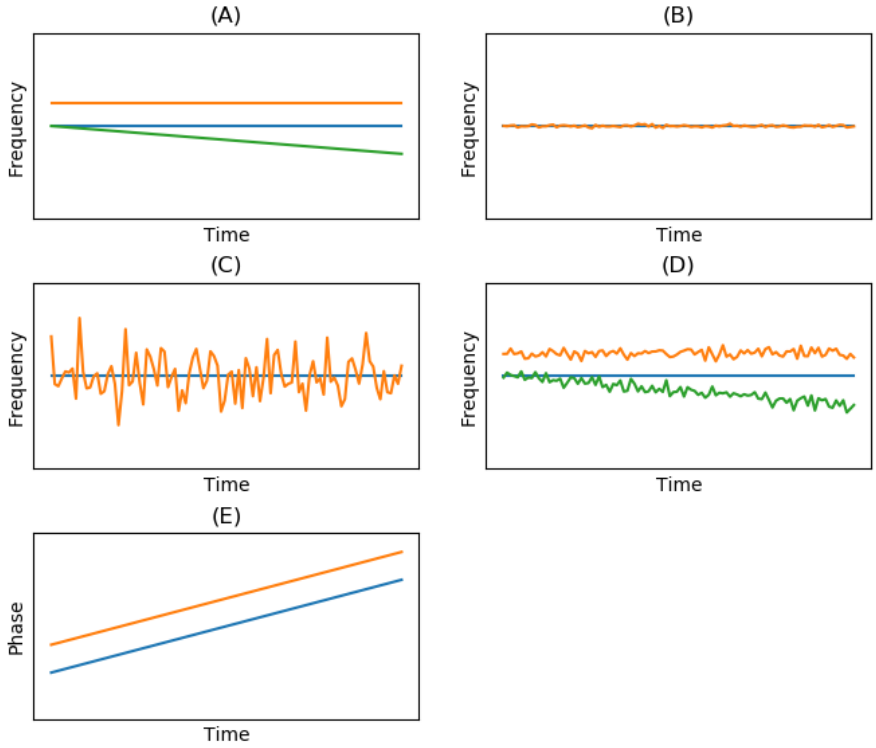
Because you told Bob that it is currently 12:20, he figures he has enough time and just casually strolls to the bus stop. He estimates that it will take him 4 minutes, so he'll get there in time for the bus that arrives at 12:25. But by the time Bob gets there, the bus driver thinks the time is 12:29 and has already left that stop. Poor Bob!

Why do Bob and the bus driver disagree on the time? They might both have set their quartz watches from two clocks that also disagreed on the time and anyway, when they set their watches, they do so only to a 1-minute accuracy at best, so they could already have been 2 minutes out from each other. This is a setting problem: to instantaneously set two different and distant clocks to the same time is hard.

The next problem is that even if they had managed to set their watches to the same time, it is highly likely that their individual watches will "tick" at slightly different rates and may have skipped the occasional "tick". Thus, as time passes their watches will slowly start to disagree on the time. This is a discipline problem: as time passes the individual oscillators need to be continually controlled so that they remain in agreement.

Why is this? It has to do with the properties of the quartz crystal oscillators driving their watches. The frequency of the oscillator is affected by temperature and age, to mention a couple of factors. These introduce something called drift. It can also be made "noisier" by things like vibrations or noise in the drive current etc. This can be referred to as "jitter". Every oscillator has what is called a "nominal frequency". This is the frequency at which the oscillator is designed to run at. Naturally there will be some deviations between units. This is extenuated by the conditions described.

The figure below shows some common oscillator behaviours.



In all the figures above, the blue line represents a perfect oscillator. It has a constant frequency that neither jitters nor drifts over time.

In (A) we see two new oscillators. The orange oscillator has a constant frequency offset. It is stable but not accurate. The green oscillator exhibits drift. Its frequency is slowly changing over time. It is neither stable nor accurate. The drift could be due to ageing, for example, or perhaps a steady temperature change.

In (B) we see a more realistically stable oscillator. The frequency has some noise, but it is minimal: we would be happy to call it stable. It is also accurate because its frequency, minus the noise, matches the reference oscillator, in blue.

In (C) we see an oscillator that is accurate on average but is not particularly stable in the short term, as it exhibits extreme amounts of noise: the frequency is not stable over the short term and only averages out over the long term.

In (D) we see the oscillators in (A) with noise, as most oscillators will include some noise.

In (E) we see a setting error, or phase offset. Both clocks are running at the same frequency, they just have a phase offset.

Going back to poor Bob and the bus driver, if they both carried a mini atomic clock on their wrists they might suffer from situation (E): a setting error where a phase error between their clocks was created as they were set. As neither the driver or Bob will have mini atomic clocks for wrist watches (too expensive and not portable), they will have a setting error combined with some frequency drift and jitter i.e. charactistic from (D).

To demonstrate the effects of a setting error and drift, imagine the following. If Bob and the driver suffered solely from a setting error, they could both look at their watches and read out 12:20 and 12:25 respectively. Then a day later, at the same time, they would make the same reading. If the effects described in (D) apply, reading their watches at the exact same time of day on the next day might see them read out 12:21 and 12:25, for example due to oscillator drift.

So, without going into the different types of oscillators out there, one can see a need to keep oscillators continually synchronised i.e. it is not just enough to synchronise each watch and then let them run freely. They must be constantly adjusted to make sure they are continually in sync over time.

One question arises: who has the correct time, the bus driver or Bob? This is where the idea of a reference or (grand) master clock comes into play. We need a very stable and accurate time source which both Bob and the bus driver could synchronise their watches to. We also need a way to send the time, as measured by our reference, to Bob and the driver such that they have a minimised setting error and can continually adjust their clocks to compensate for the effects of drift and jitter. This is what a time server will give us, as we will discover in the next article.

How ITDev Can Help

As a provider of software and electronics design services, we are often working with the Linux and Android operating systems. We have extensive experience in developing, modifying, building, debugging and bring up of these operating systems be it for new developments or working with existing kernels to demonstrate new device technologies and features.

We offer advice and assistance to companies considering to use or already using Linux or Android on their embedded system. Initial discussions are always free of charge, so if you have any questions, or would like to find out more, we would be delighted to speak to you. If you are interested in attending a workshop on embedded Linux/Android, or receiving more information on this topic, please sign up to our Embedded Linux Interest Group (http://eepurl.com/dCms-P).