Timing Distribution

Timing types

Timing is not simply data that can be transported over a network

When we say ***timing***
we can mean (at least) one of three different things

1. **Frequency**
2. **Uncalibrated time** (often *ambiguously* called phase)
3. **Time of Day** (ToD) (i.e., time locked to an International standard)

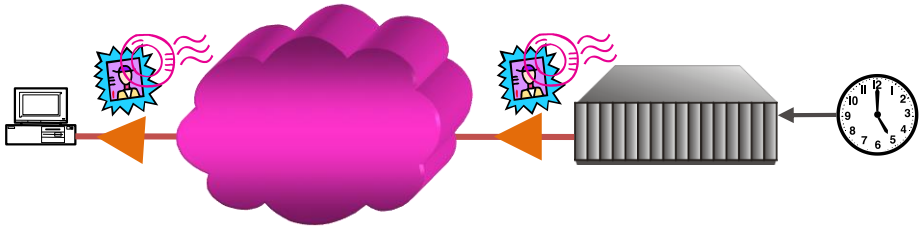
Each of these
- fulfills a distinct **need** (different applications)
- requires *additional* **means** to distribute

For example,
- *all* cellular base-stations need an accurate frequency reference
  in order not to interfere with neighboring base-stations
- cellular base-stations can benefit from a phase reference
  in order to enable smooth hand-overs between base stations
- TDD or CoMP base-stations need highly accurate ToD

Why is ToD distribution hard?

Time Protocol (RFC 868) teaches how to send ToD over IP as follows:

- ToD is the (32 bit) number of seconds since 00:00GMT on 1 January, 1900
  (roll-over about every 136 years, 1st on 7 February 2036)
- send this integer in a packet over TCP on port 37
  - user initiates TCP connection with server
  - servers sends ToD to user
  - both close TCP connection
- this protocol is the basis of UNIX rdate facility



This is great for time distribution to within accuracies of seconds

But not if we want millisecond, or microsecond, or picosecond accuracy
since the time is off by an unknown/nondeterministic travel time

Measuring this time is called ***ranging***

Why is frequency distribution hard?

Frequency is a physical phenomenon
the number of times a periodic phenomenon repeats in one second

So, accurate frequency depends on knowing the *precise* duration of a second

Unfortunately, no two clocks run at precisely the same rate
We need to *synchronize* two *plesiochronous* clocks
to make them truly *synchronous*

Frequency distribution is the means of synchronizing a remote clock
connected via a network

This could be accomplished by periodically sending packets at a precise *rate*

Unfortunately, packets are lost and received at different rates

Basic definition – the **second**

The second was once defined to be 1 / 86,400 of an *average* day

Unfortunately, the earth's rotation rate varies during the year

In 1956 the second was redefined to be 1 / 315,569,259,747 of the year 1900

This is the basis of **UT1** (Universal Time)

Unfortunately, the earth's rotation rate varies from year to year

So in 1967 the second was redefined to be

9,192,631,770 cycles of the Cs-133 hyperfine transition

In order to obtain the best accuracy

many lab measurements are combined to form **TAI** (International Atomic Time)

UTC (Coordinated Universal Time) is a compromise

It differs from TAI by an integral number of seconds

and is kept within 0.9 seconds of UT1

by introducing *leap seconds*

Basic definition – **frequency**

Frequency is defined to be the number of times
a periodic phenomena repeats in one second

It is expressed in inverse seconds = **Hertz** (Hz)

- AC current is supplied at 50 or 60 Hz
- we hear sounds up to about 20 or 25 KHz
- AM broadcast radio is transmitted between 0.52 and 1.61 MHz
- WiFi, bluetooth, and μwave ovens operate at about 2.4 GHz

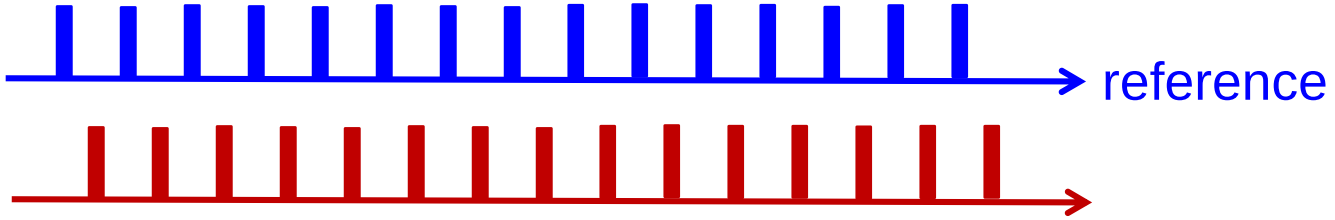
Deviation of a frequency from its nominal value is expressed
as a FFO (Fractional Frequency Offset) in

- parts per million (**ppm**) or
- parts per billion (**ppb**) or
- 10^{-n}

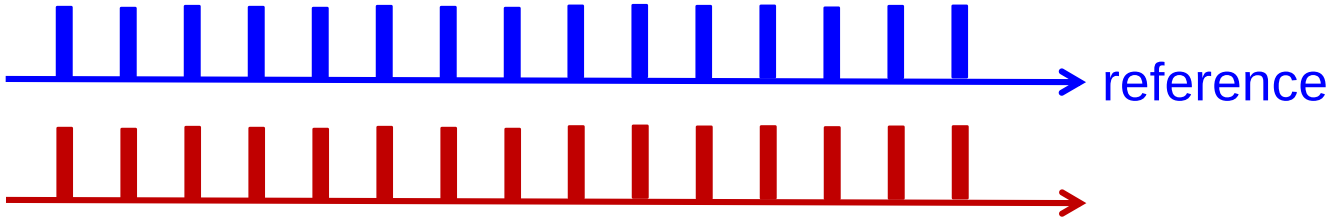
Basic definition – **phase lock**

To ensure that a periodic phenomenon has the same frequency as another phenomenon (the *reference* frequency)

It is enough to ensure that an event occurs for every reference event
This is called ***frequency lock***

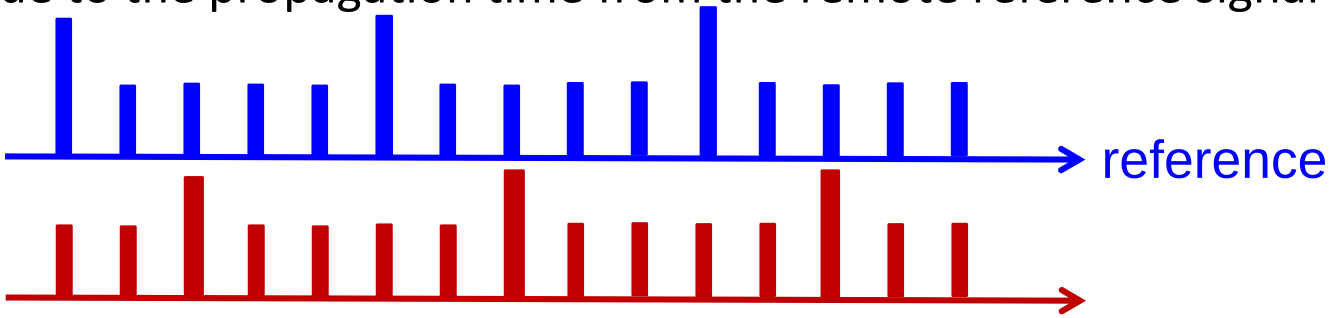


A stronger condition is ***phase lock***
where the events occur at exactly the same times

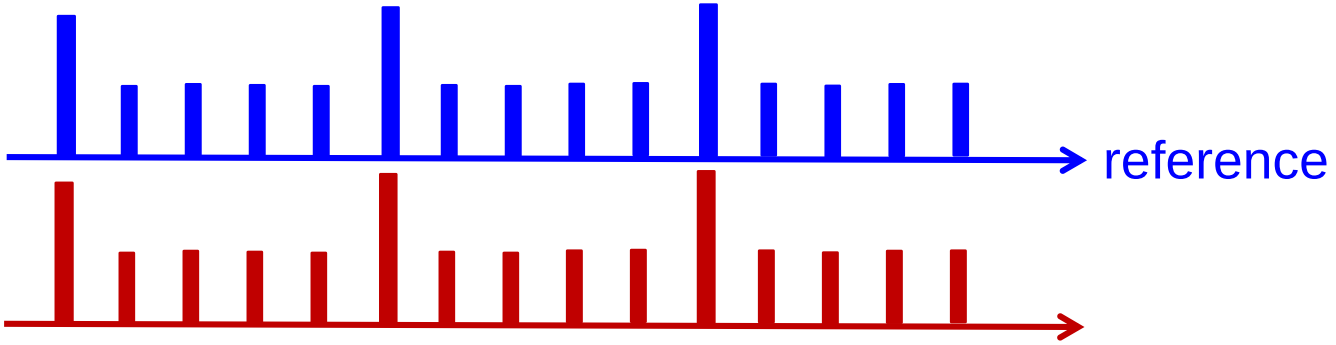


Basic definition – **uncalibrated time**

When a phenomena is phase locked to a *remote* reference
the "beginning of second" markers may not be aligned
due to the propagation time from the remote reference signal



When they *are* (although we still don't know which second is starting)
we say that we have locked **uncalibrated time**



and we can output a **1pps** uncalibrated time signal

Basic definition – **Time of Day**

Uncalibrated time (1 pps) identifies second markers
Time of Day additionally assigns time values to these markers

Once, each country defined a local time of day (hh:mm:ss)

In 1884 **GMT** was introduced as a world standard
and the world was divided into time-zones
In 1972 GMT was replaced by **UTC**, and each country :

- decides on offset from UTC (usually full hours)
- decides when/whether to use daylight savings time (summer time)

Unfortunately, several organizations claim to dictate UTC

Two important ones are :

- NIST (Boulder, Colorado) – used for GPS system
- UTC Observatoire de Paris

The difference between these two is usually < 20 nanoseconds

The **needs** (non-exhaustive list)

Frequency is needed for applications that need to :

- recover periodic phenomena (e.g. bit streams)
- transmit with spectral compatibility
- accurately measure :
  - time durations (differences)
  - periodicities
  - distances (using the constant speed of light)
- perform actions at a constant rate

Uncalibrated time is needed for applications that need to :

- perform an action *just in time*
- transmit in bursts w/o interfering with others
- tightly coordinate execution with multiple neighbors
- triangulate to find location

Time of Day is needed for applications that need to :

- precisely schedule events
- timestamp events
- prove an event tool place before/after another event

Example applications

Frequency

- synchronous (TDM) networking – bit recovery
- delivery of frequency to lock RF of GSM base-stations
- calibration (police radars, parking meters, …) and metrology
- FDD, FDMA arbitration

Uncalibrated time

- factory equipment coordination
- TDD, PON, TDMA arbitration
- optimization of networking resources/paths
- transmit in bursts w/o interfering with others
- low power sensor networks
- GPS

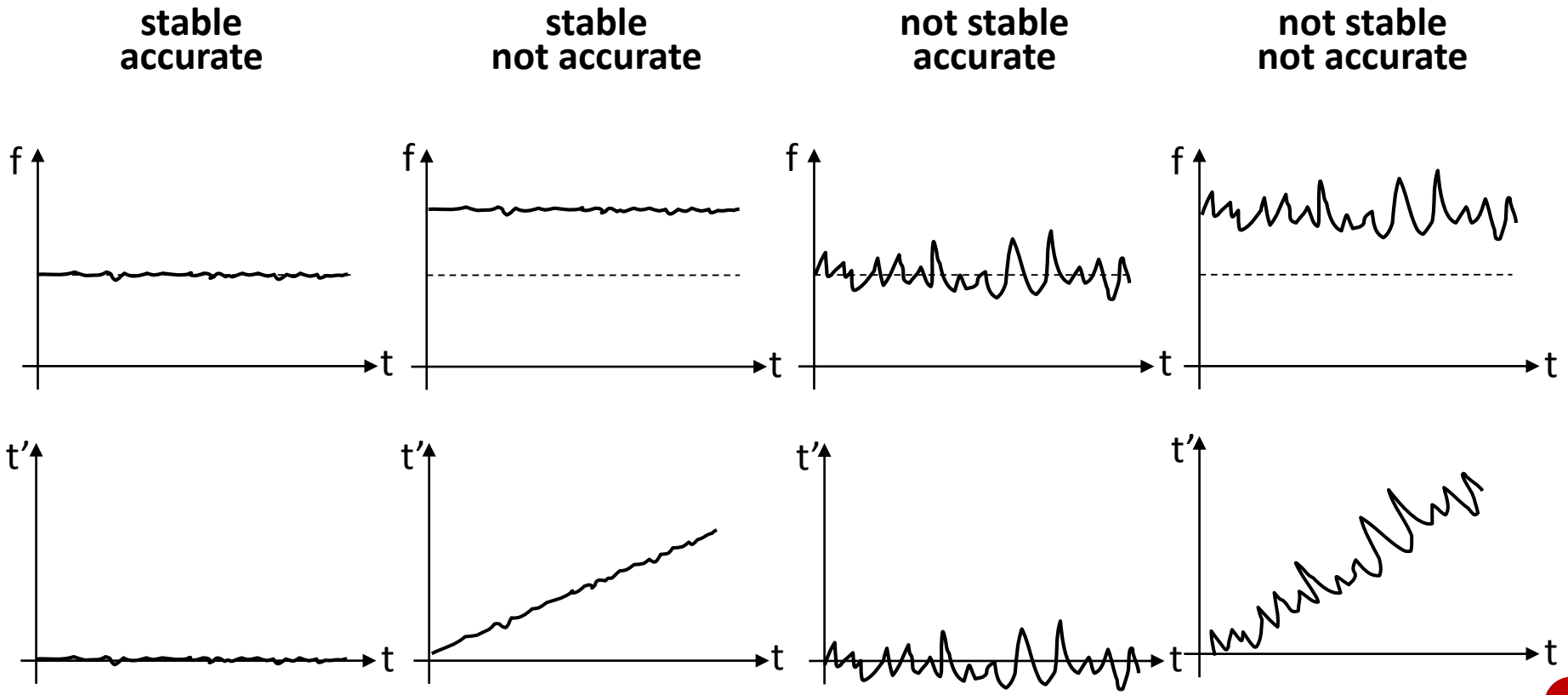
Time of Day

- delivery of time to set clocks
- time-stamping financial transactions
- smart grids (time-based metering)
- legal uses of ToD

Frequency - stability and accuracy

The performance of a system may depend on its frequency

stability and/or **accuracy**



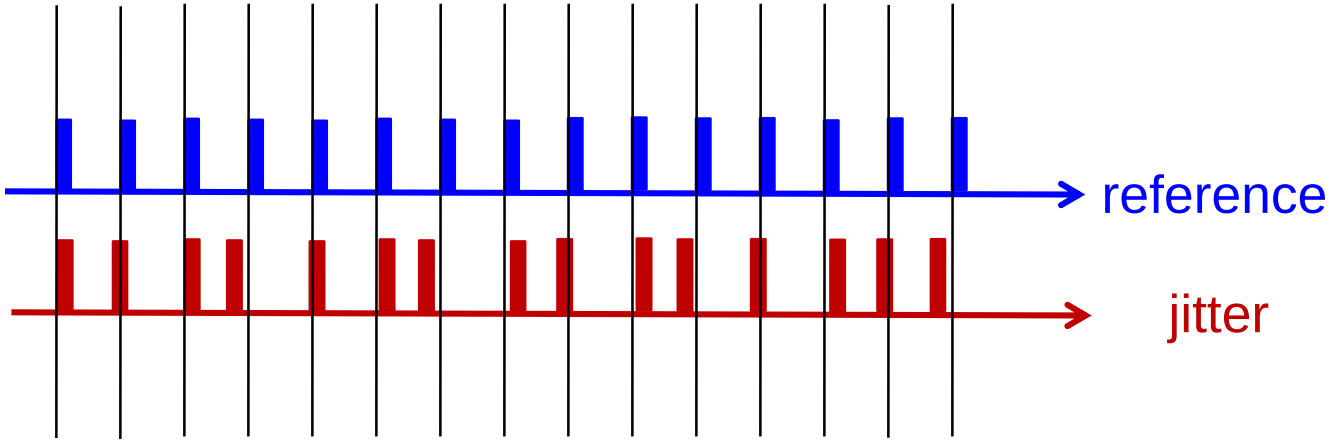
Frequency distribution **jitter**

We need metrics that enable accurate performance predictions
FFO is too blunt a tool for highly accurate frequency distribution

It is conventional to distinguish between **jitter** and **wander**

We start by measuring **TIE** (Time Interval Error)

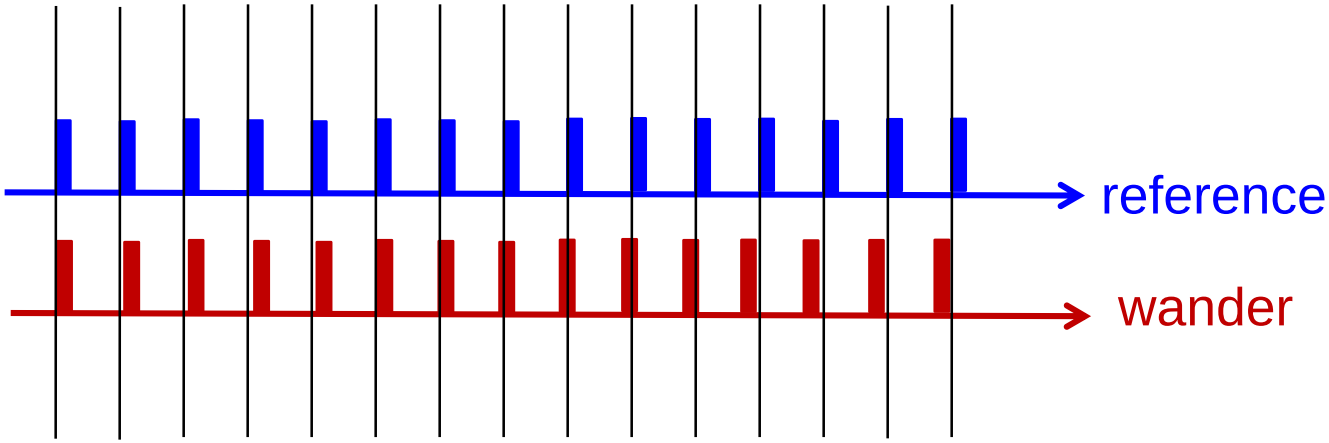
Jitter is event-event (high frequency) fluctuations in TIE



Jitter amplitude is measured in Ui_{pp} (Unit Interval peak-to-peak)
For example – for an E1 : 1 UI_{pp} = 1/2MHz = 488 ns

Frequency distribution **wander**

Slow meandering of TIE is called wander



For TDM systems, the jitter-wander dividing line is defined to be 10 Hz

Two widely used wander measures are **MTIE** and **TDEV**

MTIE is the maximum peak-to-peak variation of TIE
in all observation intervals of duration τ during the measurement

TDEV is a measure of the expected time variation TIE
as a function of integration time, after removing FFO effects

In order to reach a required performance level
both are required to obey **masks**

Master and other clocks

A **master** or **reference** (frequency) clock
outputs some periodic function of time

For example $\sin(2\pi\nu_0 t)$ or $PULSE(2\pi\nu_0 t) = \begin{cases} 1 \text{ if } 2\pi\nu_0 t \text{ is an integer} \\ 0 \quad else \end{cases}$

The *phase* Φ(t) of a clock is the argument of the periodic function
so, for a master clock, the phase $\Phi(t) = 2\pi\nu_0 t$

All other clocks have some other phase Φ(t)
For example, sin(Φ(t)) or PULSE(Φ(t))

The time of a clock is defined to be $T(t) = \frac{1}{2\pi\nu_0}\Phi(t)$
so, for a master clock, T(t) = t
and for other clocks, the **Time Error** TE(t) = T(t) – t

The frequency of a clock is the derivative of its phase (divided by 2 π)
so, for a master clock, the frequency is ν_0

The clock model

We assume a non-master clock's phase changes over time as follows

$$\Phi(t) = \Phi_0 + 2\pi(\nu_0 + \Delta\nu)t + \pi D \nu_0 t^2 + \varphi(t)$$

where

Φ_0 is the initial phase

ν_0 is the nominal frequency

$\Delta\nu$ is the offset from the nominal frequency

D is due to constant frequency drift

$\varphi(t)$ is a random component (*phase noise*)

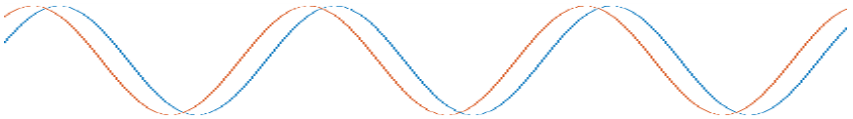
We further define $x(t) = \frac{1}{2\pi\nu_0}\varphi(t)$ and $y(t) = \frac{1}{2\pi\nu_0}\frac{d\varphi(t)}{dt}$

note that x is in units of time, and y is the derivative of x

Clock Model (illustrative examples)

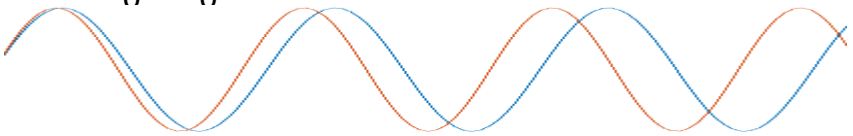
If the clock has exact frequency but a time offset $x = t_0 + t$

$$\Phi(t) = 2\pi t_0 + 2\pi \nu_0 t$$



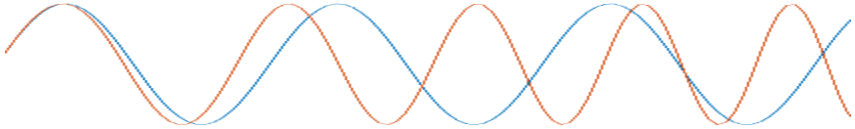
If the clock has a constant frequency offset $\nu_0 = \nu_0 + \Delta\nu$

$$\Phi(t) = 2\pi(\nu_0 + \Delta\nu)t$$



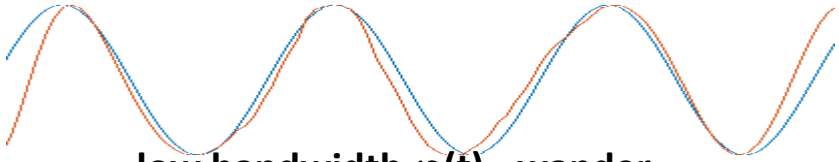
If the clock has frequency drift $\nu_0 = \nu_0 + \frac{1}{2} D t$

$$\Phi(t) = 2\pi \nu_0 t + \pi D \nu_0 t^2$$

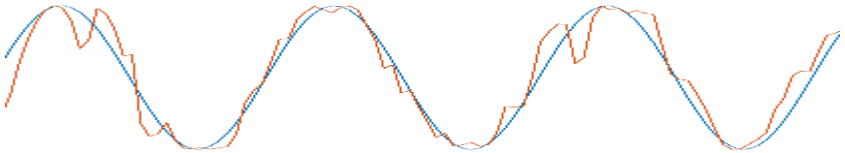


If the clock only has random phase noise

$$\Phi(t) = 2\pi \nu_0 t + \varphi(t)$$



low bandwidth φ(t) - wander



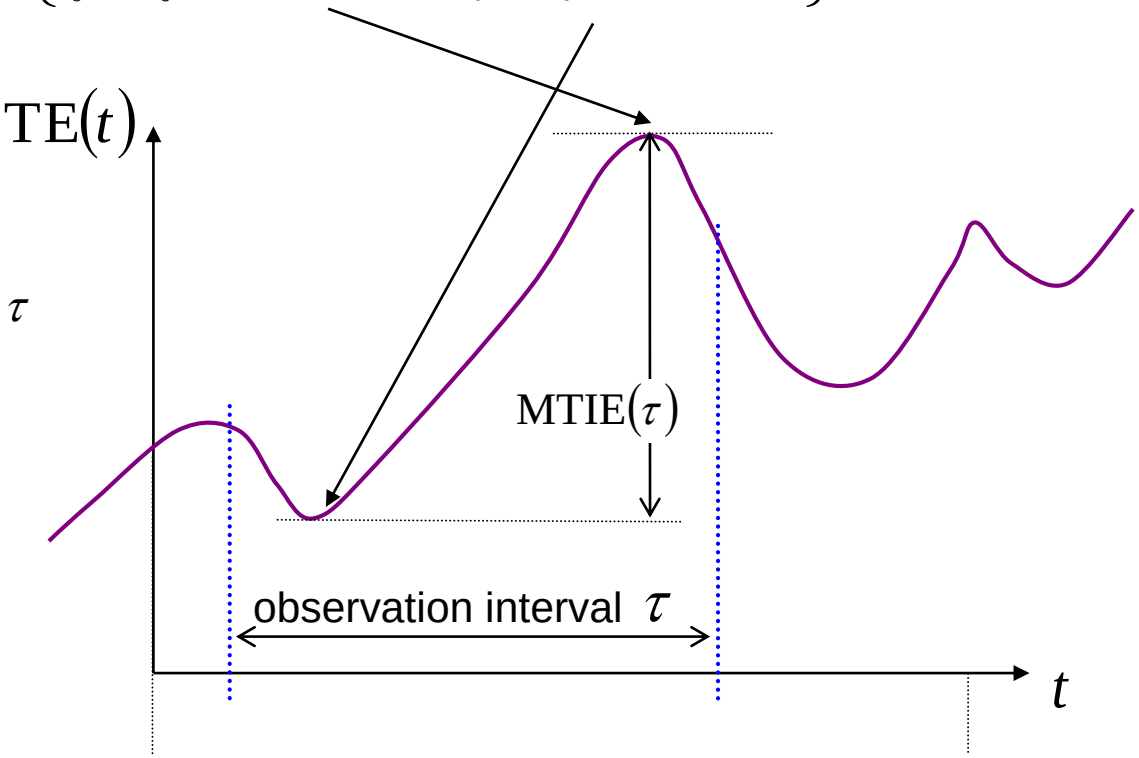
high bandwidth φ(t) - jitter

Maximum Time Interval Error (MTIE)

MTIE(τ,T) is defined as the maximum peak-to-peak variation of the Time Error as a function of observation interval τ

$$\mathrm{MTIE}(\tau) = \max\left\{ \max_{t_0 \le t \le t_0+\tau} \{\mathrm{TE}(t)\} - \min_{t_0 \le t \le t_0+\tau} \{\mathrm{TE}(t)\} \right\}$$

the larger the interval τ *the larger* MTIE is



Time deviation (TDEV)

TDEV $\sigma_x(\tau)$ is a measure of the time *stability* of a clock's phase
as a function of observation interval τ

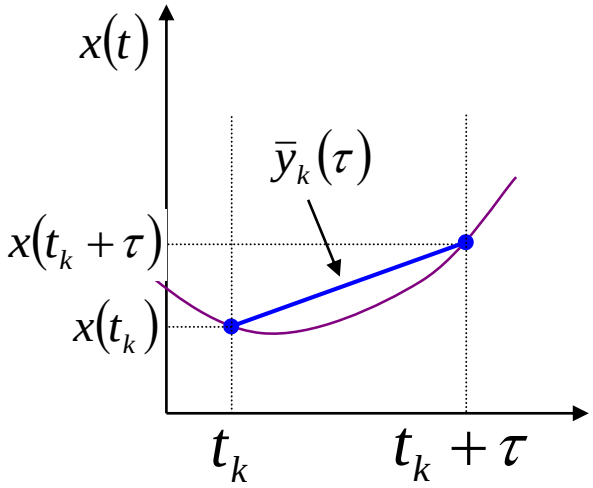
It is based on the **Allan Variance** $\sigma_y^2(\tau)$
which is a measure of frequency stability

The TDEV is the conversion of the Allan variance back into the time domain

$$\text{TDEV } \sigma_x(\tau) = \frac{\tau}{\sqrt{3}} \sqrt{\text{Mod } \sigma_y^2(\tau)}$$

First we find the frequency averaged over τ
at sampled times t_k

$$\bar{y}_k(\tau) = \frac{x(t_k + \tau) - x(t_k)}{\tau}$$



We are interested in the statistical variance $\sigma_y(\tau)$
but calculating this is tricky because the phase noise is not *white*

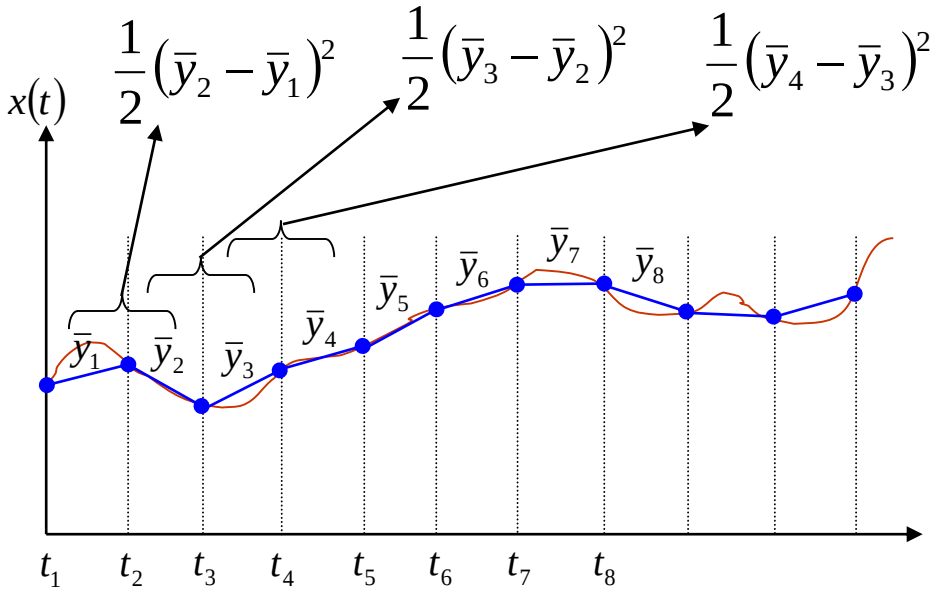
Allan variance

The *regular* Allan variance and deviation are defined

as the expectation of the difference between two successive y_k

$$\text{AVAR}\ \sigma_y^2(\tau) = \frac{1}{2}\left\langle (\bar{y}_{k+1} - \bar{y}_k)^2 \right\rangle = \frac{1}{2\tau^2}\left\langle \left[x(t_k + 2\tau) - 2x(t_k + \tau) + x(t_k)\right]^2 \right\rangle$$

$$\text{ADEV}\ \sigma_y(\tau) = \sqrt{\sigma_y^2(\tau)}$$



The *modified* Allan variance is similar

but doesn't require the integration time and sampling interval to be equal

Obtaining **frequency**

There are many ways to obtain stable and accurate frequency

The most common are :

- use of local frequency references
  - crystal oscillators
  - atomic clocks
- exploiting synchronous networks
  - TDM, SDH
  - SyncE
- exploiting wireless
  - GPS
- distributing timing over packet networks
  - using periodic packet streams
  - using time distribution protocols

Exploiting **frequency references**

Local frequency references can supply frequency for applications

All local references suffer from
- **drift** depending on environment (temperature, humidity, etc.)
- **aging**

In order of accuracy (low to high *long-term* accuracy)
- LC circuits
- piezoelectric crystal oscillators
- temperature compensated crystal oscillators (TCXO)
- crystal oscillators in temperature controlled environments (OCXO)
- cavity resonators
- Rubidium atomic clocks
- Cesium atomic clocks
- Hydrogen masers

A frequency reference which is stable/accurate enough
(within 10^{-11} of UTC frequency)
is called a **PRC** (Primary Reference Clock)

Exploiting TDM networks

Since highly accurate frequency references are expensive
it is usual to have only one PRC (or a small number of them)
and *distribute* its frequency to all locations where it is needed

Conventional synchronous (TDM) networks

- require accurate frequency for their own use
- automatically distribute frequency in their **physical layer**
- distribute frequency end-to-end by master-slave relationships
- maintain a hierarchy of timing strata (PRC, stratum 1, stratum 2, …)
- each stratum level has well-defined performance parameters

This frequency can be provided as a service for other needs

Since these networks are ubiquitous
frequency distribution services are often free or inexpensive

For example, if an E1 supplies data to a GSM cell-site
its physical layer frequency can be used to lock RF

A TDM alternative - **SyncE**

Asynchronous (Ethernet/MPLS/IP) networks
are rapidly replacing synchronous networks
Free frequency distribution is thus becoming much rarer

But there is a way of having your cake and eating it too

Although modern Ethernet physical layers transmit continuously
the standard Ethernet physical layer is not frequency locked
But it is easy to replace it with a synchronous one

This is the idea behind **SyncE** (Synchronous Ethernet)

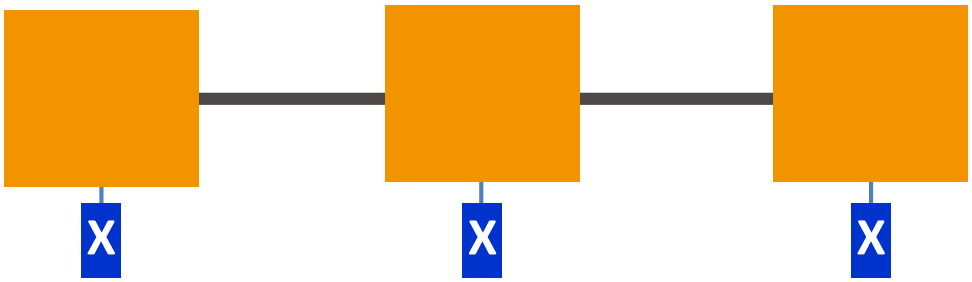
SyncE does not change packet performance

- the physical symbol rate is made synchronous
- but Ethernet frames are still released asynchronously

End-to-end frequency distribution requires end-to-end support
for existing networks this may require *forklift upgrade*

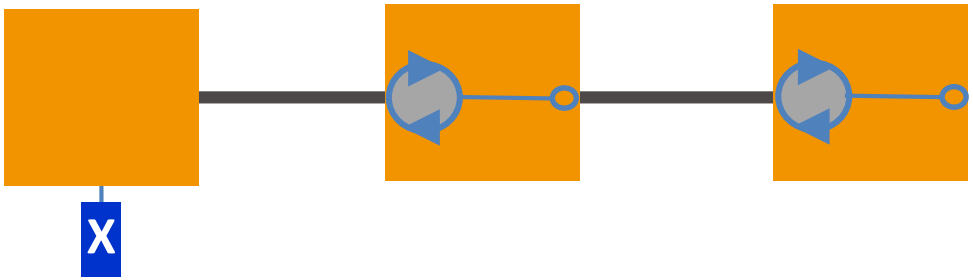
Synchronous Ethernet

In traditional Ethernet, PHYs use free-running oscillators



When using Synchronous Ethernet (SyncE)

the receiving PHY uses a PLL to lock onto the frequency of the reference and uses this frequency as the source for its downstream transmissions



This is precisely how TDM/SDH networks work

And similar to SDH **S**ynchronization **S**tatus **M**essage

the G.8262 ESMC slow protocol distributes SyncE status

Exploiting **GNSS**

Another ubiquitous frequency-carrying physical layer is the **G**lobal **N**avigation **S**atellite **S**ystem

The US **GPS** satellite system

- transmits a highly accurate (*long term*) frequency reference
- covers over majority of the world's surface
- can be received as long as there is a clear view of the sky
- has built-in redundancy
- receiver price is now minimal

Similar to GPS :

- the Russian **GLONASS** system
- the Chinese **COMPASS** (Beidou) system
- the new European **Galileo** system

The **LORAN** low frequency terrestrial Maritime navigation system can be similarly used

Exploiting **packet traffic**

If there is no continuous physical layer carrying periodic events
then we must distribute frequency as *information* (data)

A simple method is to send a *periodic* stream of **timing packets**

However, while these packets are sent at a rate R
that is, at times $T_n = n\,R$

they arrive at times $t_n = T_n + d + V_n$ where

- D = average propagation delay through the network
- V_n = **PDV** (Packet Delay Variation)

But, by proper averaging/filtering (actually control loops are needed)

$$\langle t_n \rangle = T_n + d = n\,R + d$$

and the packet rate R has been recovered (d is unimportant for now)

This is called **ACR** (Adaptive Clock Recovery)

Note : the filtering takes (convergence) time (uncertainty theorem)

Use of **timestamps**

Instead of sending a periodic stream of timing packets
we can send a nonperiodic stream
but insert a **timestamp** into the packet
identifying the precise time the packet was sent
If this is hard to do accurately, we can use a *follow-up* packet
that identifies the precise time of the previous packet

This functionality exists in all time distribution protocols

For *frequency distribution*, we only need a one way protocol
Packets sent (broadcast/multicast/unicast)

- from master fed by PRC
- to slave requiring accurate frequency

Time distribution requires a 2-way dialog (ranging)

- frequency must be continuously recovered
- periodic ranging provides uncalibrated time
- timestamps provide time identification

NTP and PTP

There are two well-known time-stamping packet time distribution protocols

Network **T**ime **P**rotocol (NTPv3 = RFC 1305, NTPv4 = RFC 5905)

- over UDP/IP (IPv4 or IPv6)
- client/server model with stateless server
- distributes UTC (with leap seconds)
- pure software implementation
- client can combine multiple servers for higher accuracy
- millisecond accuracies sometimes possible

Precision **T**ime **P**rotocol (IEEE 1588 v1=1588-2002 v2=1588-2008)

- over various transport protocols (Ethernet, IP, ...)
- master/slave model (and hierarchy)
- distributes TAI (no leap seconds) but can inform of offset
- requires hardware timestamping
- client can choose between master on list (**B**est **M**aster **C**lock **A**lgorithm)
- defines various on-path support elements
- White Rabbit extensions for subnanosecond accuracy

Ranging

To ensure an event occurs simultaneously with a reference event
we need to determine how long it took
for the information on the reference event to arrive

This is called **ranging**

For physical links we can use **T**ime **D**omain **R**eflectometry

For packet interchange ranging requires a 2-way protocol

- master – slave master controls slave
- client – server client sends request to time server when it needs

PONs (Passive Optical Networks) distribute uncalibrated time
because upstream traffic is TDMA

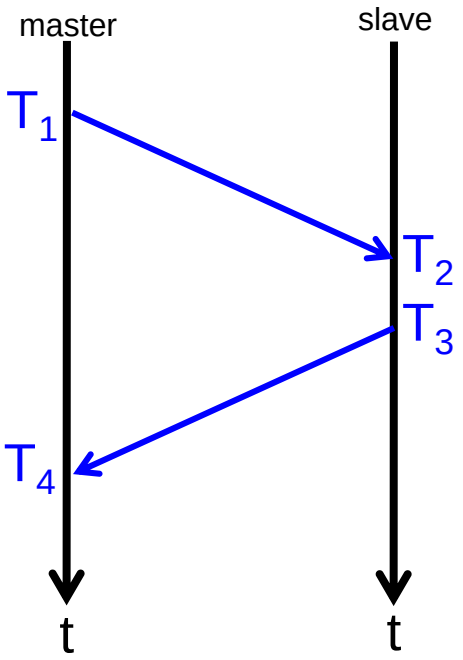
OLT is a master to ONT slaves

- GPON locks ONT frequency and OLT compensates for time offset
- EPON locks ONT time

How is ranging performed ?

The idea behind ranging is simple (demonstrated for master-slave)

1. master sends a packet to slave at time T_1
2. packet is received by slave at time T_2
3. slave replies to master at time T_3
4. master receives reply time T_4



We can not compare T_1 and T_4 with T_2 and T_3

but

- $T_4 - T_1$ = round trip propagation time + slave processing time
- $T_3 - T_2$ = slave processing time

so

- $T = (T_4 - T_1) - (T_3 - T_2)$ = round trip propagation time

If we can assume **symmetry** ($d_{1\rightarrow2} = d_{3\rightarrow4}$)
½ T is the required one-way propagation time

Minimum gating

Even if the two directions are physically symmetric (e.g., co-routed)

the delays in the two directions may be very asymmetric due to loading

We can overcome this by waiting for a lucky packet

that traverses all switches without waiting in any queues

We can identify such a packet by its having the minimum delay T_4-T_1

If such a packet is found, its round-trip will be

- close to physical flight time (unaffected by random delays)
- approximately symmetric

If even 1% of the packets have zero or nearly zero queuing delay

we can still reconstruct time well

since wander is still Nyquist sampled

The frequency – time connection

Technically we do not have to distribute frequency
in order to distribute (uncalibrated) time

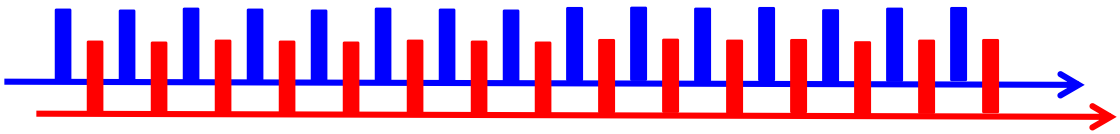
We *could* simply send frequent time updates

However, the following is more efficient :

1. slave first acquires frequency from the timing packets
2. stable and accurate 1 pps train is locally generated
3. ranging is performed
4. 1 pps train is moved to proper position

This reduces timing packet rate (and network load)
but introduces a convergence time

If we have an alternate source of frequency (e.g., SyncE)
then we can reduce the packet rate without convergence time



NTP

Network **T**ime **P**rotocol synchronizes hosts and routers of the Internet

NTP is arguably the longest continuously running protocol in the Internet

There are thousands of NTP servers on the Internet
and every PC, laptop, and smartphone has an NTP client

NTP nominally provides accuracies of
- < 1 millisecond in controlled environments
- tens of milliseconds on LANs
- hundreds of milliseconds over the Internet

NTP runs over UDP

The latest version of NTP (NTPv4) is defined in RFC 5905
and there is complete NTP (C language) reference code at ntp.org

NTP has many robustness mechanisms

NTP has optional security features (presently being updated)

NTP (cont.)

NTP is a client/server protocol

- server holds no client state
  - packet contains algorithm-specific variables
- client query rate adapts to accuracy
- a single client can query multiple servers for
  - higher accuracy
  - redundancy
  - diversity

NTP server provides UTC (includes leap seconds)

- client responsible for translation to local time
- NTP usually uses 64-bit timestamps (32bit seconds + 32bit fractional seconds)
  which means it rolls over every 136 years (an *epoch*)
  and its resolution is 232 picoseconds

PTP

Precision **T**ime **P**rotocol synchronizes devices on a network

- PTPv1 was designed for industrial applications
- PTPv2 was designed to improve accuracy and precision
- a new version (including security features) is in progress

PTP operates over arbitrary PSNs, currently defined PSNs include

- Ethernet IEEE 802.3
- UDP/IPv4
- UDP/IPv6
- DeviceNet
- ControlNet
- PROFINET

PTP allows profiles for different applications (e.g., power, telco, enterprise)

PTP operates in domains, in each domain there is

- one grandmaster (a GM can be GM to many domains)
- many slaves

PTP (cont.)

PTP distributes TAI using 10 byte timestamps – 48bit seconds, 32bit nanosec

PTP separates

- ranging procedure (delay messages) from
- frequency distribution (sync messages)

PTP supports

- hardware timestamping
- 2-step timestamping
- on-path support (see clock types below)

PTP defines 10 different message formats

- **Sync** – initiated by the Master clock and carries the timestamps of the master clock.
- **Delay_Req** – initiated by the Slave clock to measure the propagation delay of the sync message
- **Pdelay_Req** – initiated by a clock port that supports link propagation delay measurement.
- **Pdelay_Resp** – sent by a clock port that supports link propagation delay measurement

PTP message types

PTP defines 10 different message formats

Event messages (time critical)

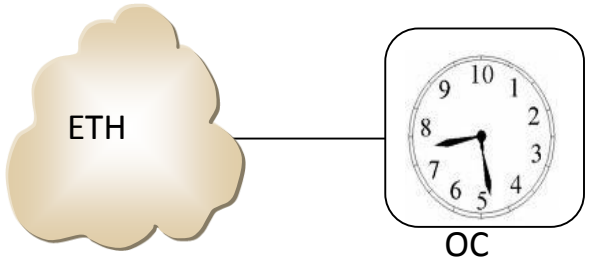
- **Sync** – timestamps from master clock
- **Delay_Req** – ranging request from save clock to master
- **Pdelay_Req** – for link delay measurement
- **Pdelay_Resp** – for link delay measurement

General messages (data messages - not time critical)

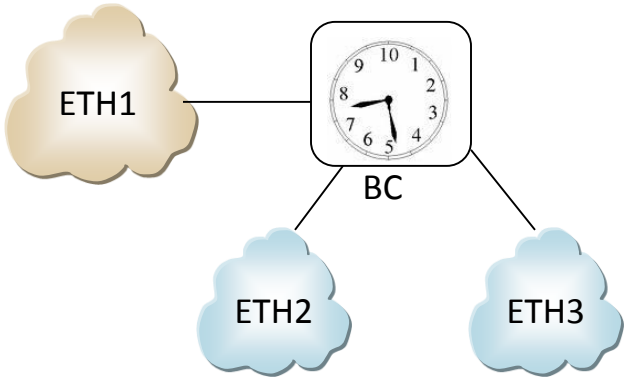
- **Announce** – to establish synchronization hierarchy
- **Follow_Up** – timestamp of previous sync message from master
- **Delay_Resp** – reply of the master to Delay_Req message.
- **Pdelay_Resp_Follow_Up** – timestamp of previous Pdelay_Resp message
- **Management** – query and update datasets of PTP clocks
- **Signaling** – negotiation between PTP clocks

PTP clocks (1)

Ordinary clock – Master or Slave

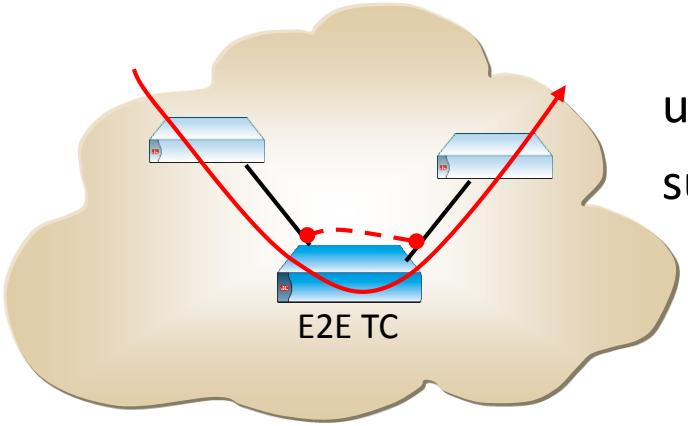


Boundary clock – Slave on one network and Master on another



PTP clocks (2)

End-to-End (E2E) Transparent Clock (TC)



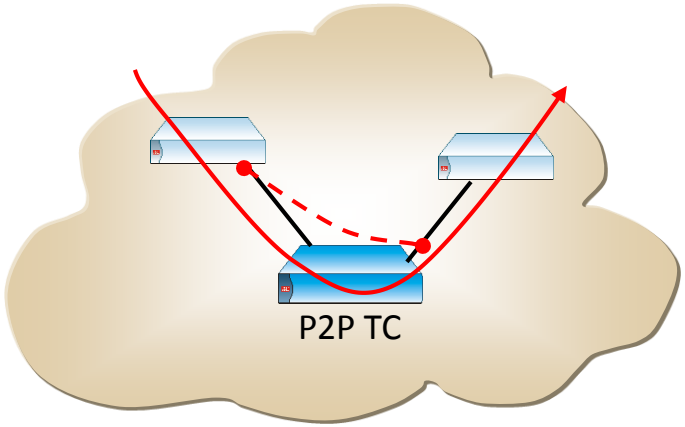
updates **T**ime **C**orrection **F**ield

sums residence times

subtracting TCF

leaves constant delay

Peer-to-Peer (P2P) transparent clock



TCF includes on-the-wire time

subtracting TCF

gives exact time