Echtzeitsysteme
Lehrstuhl Informatik VI - Robotics and Embedded Systems

Embedded Distributed Systems

Wintersemester 2012/2013

Prof. Dr. Alois Knoll

TU München

Lehrstuhl VI Robotics and Embedded Systems

Team

Prof. Dr. Alois Knoll

Philipp Heise
(E-Mail: heise@in.tum.de)

Steffen Wittmeier
(E-Mail: steffen.wittmeier@in.tum.de)

Michael Jäntsch
(E-Mail: michael.jaentsch@in.tum.de)

Lecture Overview

1. Introduction to realtime systems
2. Time and Digital Clocks
3. Embedded Hardware
4. Communication
5. Realtime-Scheduling

Embedded Distributed Systems

Part 1: Introduction

Definition

„Ein Echtzeit-Computersystem ist ein Computersystem, in dem die Korrektheit des Systems nicht nur vom logischen Ergebnis der Berechnung abhängt, sondern auch vom physikalischen Moment, in dem das Ergebnis produziert wird.“

„Ein Echtzeit-Computer-System ist immer nur ein Teil eines größeren Systems, dieses größere System wird Echtzeit-System genannt.“

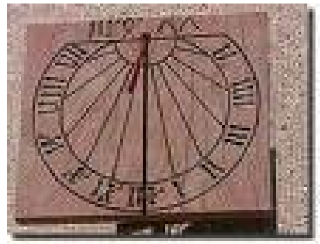
Hermann Kopetz, TU Wien

Properties of Realtime-Systems

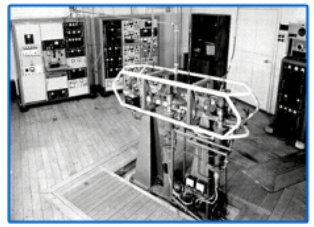
- Integration of physical processes and computers
- Capture and change physical processes over time
- Machine time not separated from real time of physical processes
- Correctness of results not only depends on logical result but also on the instant of time at which it is available

What is time?

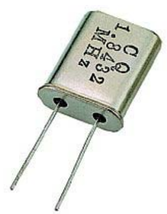
- Historically:
  - Solar time: The interval between two successive returns of the Sun to the local meridian.
  - A solar second is 1/86400 of this interval.
- Today:
  - Atomic clock: Measurement of time based on microwave signals emitted by electrons in atoms when changing energy levels (typically Cesium atoms)
- Computer:
  - Measurement of time based on Quartz oscillations
  - Not as accurate as an atomic clock
  - Examples:
    - http://www.ptb.de/cms/presseaktuelles/uhrzeitapplikation.html
    - http://www.time.gov/timezone.cgi?UTC/s/0/java



Sun Clock
German Museum



First Cesium atom clock



Quartz

Time in realtime systems

- Problems in measuring and processing time:
  - Discretization
  - Drift
  - Synchronization of clocks
- Problems in software execution (over time):
  - Determinism
  - Parallelism
  - Synchronization of processes
- Time is the only valid reference which is available to all systems

Properties of realtime systems

- Timing requirements
  - Timeliness (not too early, not too late)
  - Guaranteed response times
  - Synchronization of events and data
  - **Important:** Overall execution time of system unimportant (e.g. railroad crossing)
- Properties derived from „embedded" nature
  - Realtime systems typically have high I/O load
  - Realtime systems often have to be fault tolerant as they interact with physical processes and system failures could have severe consequences
  - Realtime systems are often distributed

Timeliness vs. Performance

- Consequence of timeliness requirement:
  Mechanisms that increase performance but might affect determinsm of the system are typically avoided:
  - Virtual Memory
  - Garbage Collection
  - Asynchronous I/O
  - Recursion

Classification of Realtime Systems

- Realtime systems can be divided in different classes:
  - Based on the consequences in case of deadline violations:
    Hard- vs. soft realtime systems
  - Based on the execution model:
    time-triggered vs. event-triggered

Hard and Soft Realtime Systems

- **Soft Realtime Systems:**

  Computations have a deadline but a violation of this deadline does **not** have severe consequences. Potentially, the results can still be used affecting only the *quality of* service.

  Example for a soft realtime system: Video streaming
  Consequences of deadline violations: individual frames are lost, the video has artifacts

- **Hard Realtime Systems:**

  A deadline violation can have severe consequences (either material or personal damage).

  Example for hard realtime system: Rocket control
  Consequences of deadline violations: crash or self-destruction of rocket

Time-Triggered vs. Event-Triggered

- **Time-triggered applications:**
  - Sequence of process execution defined at compile time
  - Requires a global time base among processing units with high precision (low error of internal synchronization)
  - Each computation uses a single *time slot.* Estimation of worst case execution times required.
  - **Advantage**: Static scheduling → predictable, deterministic behavior
- **Event-triggered applications:**
  - Event trigger execution of computations (processes)
  - Dynamic scheduling → chronology of events not known at compile time

Definition Embedded System

- Technical system which is controlled by an integrated computer. Most of the time, the computer is not visible from the outside and can not be (re-)programmed. For the control, the system is normally equipped with very specific I/O interfaces.
- Typically, microcontrollers are used that are limited in terms of processing power and memory size (compared to standard desktop computers).

Definition Cyber-Physical System

"A cyber-physical system (CPS) is an integration of computation with physical processes. Embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice versa. As an intellectual challenge, CPS is about the intersection, not the union, of the physical and the cyber. It is not sufficient to separately understand the physical components and the computational components. We must instead understand their interaction. (Lee)"

Examples

Realtime Systems are everywhere!!!









Examples

- Car
  - Engine management:
    - Fuel injection and ignition depends on engine speed
    - High engine speeds required reaction times < 1 ms
  - Airbag:
    - Continuous monitoring of sensors
    - Typically, redundant sensors → to prevent erroneous airbag triggering
    - Reaction times about 1ms

Further Examples

- Elevator control
- Traffic light control
- Air conditioning system
- Mobile phones
- ABS system

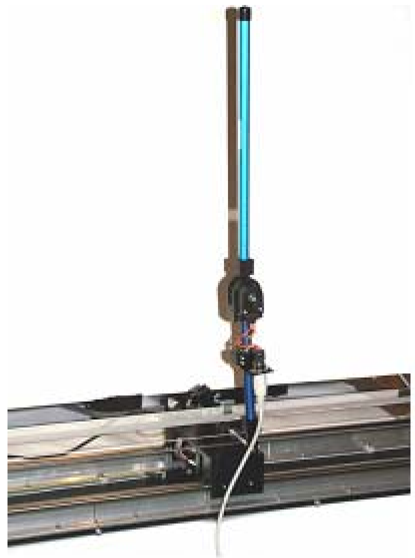
…

Examples at I6

Control exercises

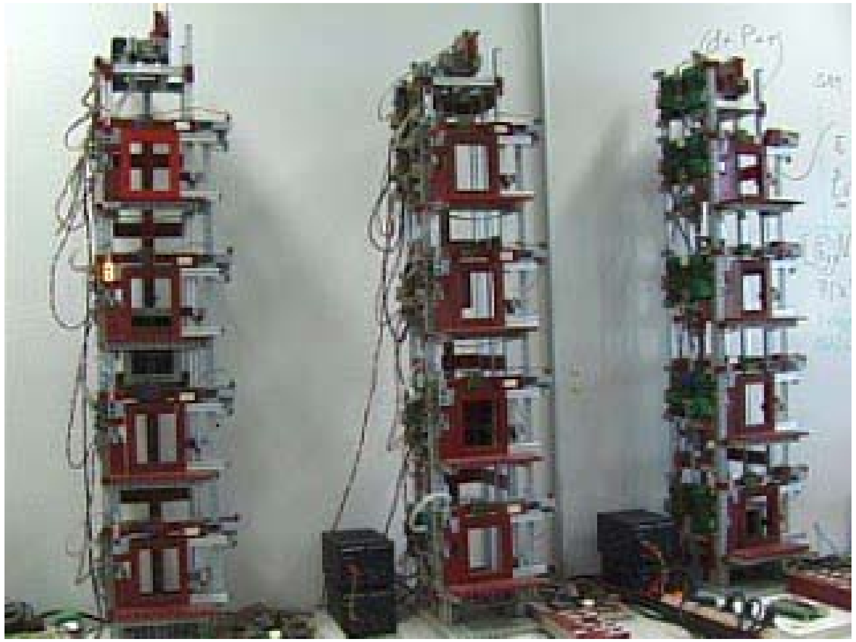
Hovering Bar

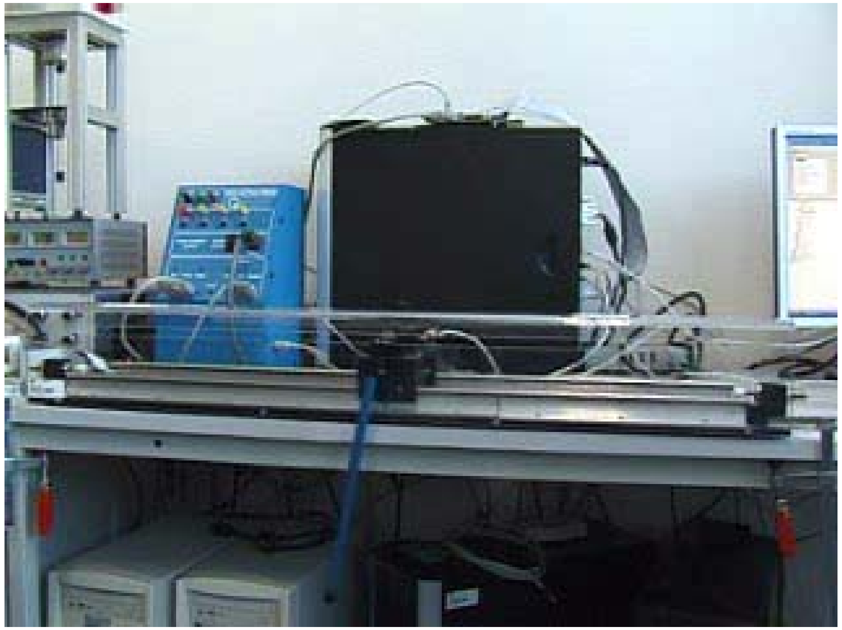
Production technique



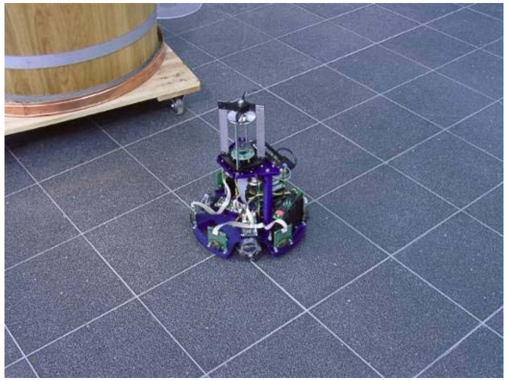
Inverted Pendulum

Videos

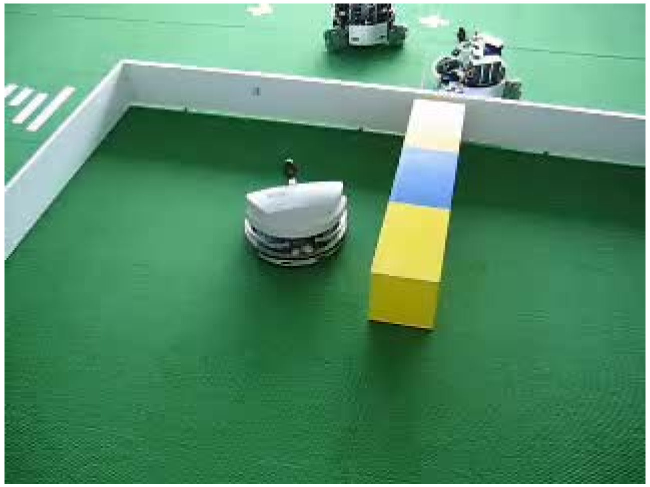


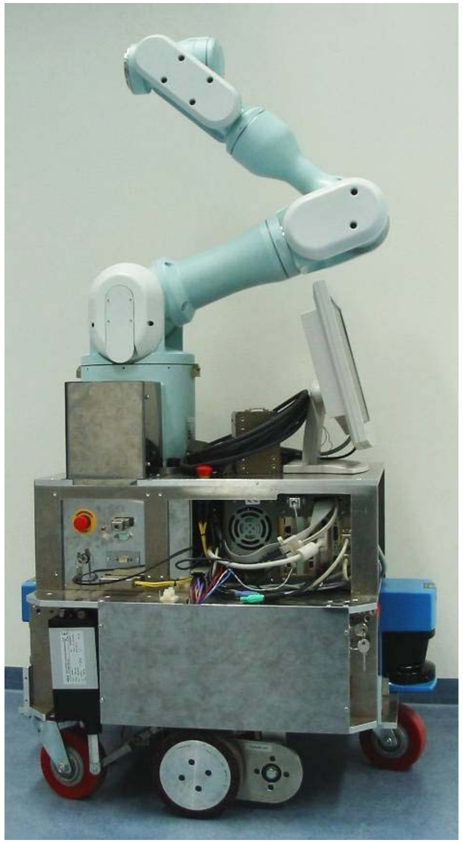


Robot Control



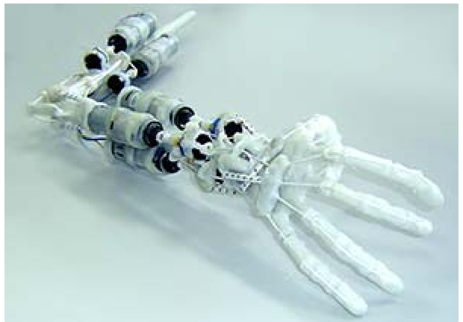
Robotino





Leonardo

Stäubli



Tumanoid

Literature

- Lee, Seshia: Introduction to Embedded Systems, http://leeseshia.org/
- H. Kopetz, Real-Time Systems, 2nd Edition, Chapter 3, 2011, Springer
- Jane W. S. Liu: Real-Time Systems (Überblick, Schwerpunkt Scheduling)

- Stuart Bennet: Real-Time Computer Control: An Introduction (Überblick, Hardware)

- Alan Burns, Andy Wellings: Real-Time Systems and Programming
Languages (Schwerpunkt: Programmiersprachen)

Embedded Distributed Systems

Part 2: Time and Clocks

Partly taken from: H. Kopetz, Real-Time Systems, 2nd Edition, Chapter 3, 2011, Springer

Introduction: Time and Order

1. The constants of physics are defined in relation to the standard of time: the physical second (e.g., speed: m/s)

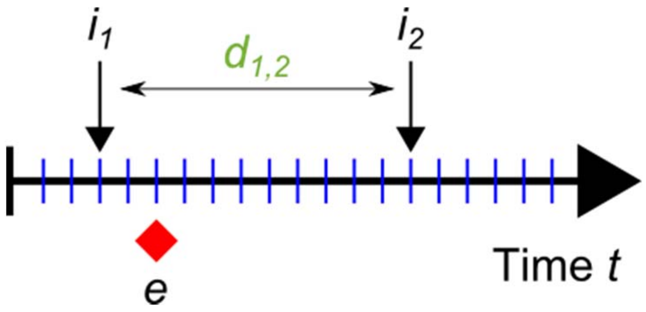
   The global time in cyber-physical real-time systems should be also based on the metric of the physical second.

2. In distributed systems, the nodes must ensure that the events are processed in the same consistent order (preferably in the temporal order in which the events occured).

   A global time base helps to establish such a consistent temporal order on the basis of the time-stamps of the events.

Temporal Order

- The continuum of Newtonian real time can be modeled by a directed timeline consisting of an infinite, dense and ordered set {T} of **instants *i*** (points in time).
- The section on the time line between two instants is called **duration *d***.
- **Events *e*** take place at an instant of time (but have no duration).
- Events that occur at the same instant are said to occur ***simultaneously***.
- Instants are totally ordered.
- Events are partially ordered (additional criteria are required to totally order events, such as the node at which the event occured).



Causal Order

- For real-time applications, the ***causal dependencies*** among events *e* are of interest.
- The ***temporal order*** of two events is ***necessary***, but ***not sufficient***, for their causal order.
- ***Causal order*** is ***more*** than ***temporal order***.

Digital Physical Clocks

- In digital physical clocks, a physical oscillation mechanism that periodically increases a counter is used to measure time.
- The periodic event is called a **microtick**.
- The duration **between two consecutive** microticks is called a **granule** of the clock.
- The granularity of a digital clock leads to a **digitization error** in time measurement.

Digital Physical Clocks: Phased-Locked Loop (PLL)

- Typical frequencies of crystal oscillators: kHz … MHz
- CPUs, mobile phones, etc. require clock signals with frequencies in the GHz range
- Precise multiplication of the frequency of crystal oscillators is required

  → Phase-Locked Loop (PLL)

  „A PLL is a circuit which synchronizes the frequency of the output signal generated by an oscillator with the frequency of a reference signal by means of the phase difference of the two signals.“

  (J. Encinas)

Digital Physical Clocks: Reference Clock & Absolute Time-Stamp

- A reference clock is a ***clock z*** that runs at frequency $\boldsymbol{f^z}$ and which is in perfect sync with the international standard of time.
- $\boldsymbol{1/f^z}$ is the granularity $\boldsymbol{g^z}$ of clock *z.*
- The ***granularity*** of a clock ***k*** is given by the number of microticks of the reference clock *z* between two subsequent microticks of the clock *k.*
- An **absolute time-stamp** of an event is the time of its occurence measured by the reference clock.
- The duration between two events *e* is measured counting the microticks of the reference clock.
- The temporal order of events that occur between two consecutive events of the reference clock cannot be reestablished from their absolute time-stamps.

Digital Physical Clocks: Clock Drift

- The **drift rate** ρ of a physical clock *k* with respect to a reference clock *z* is defined as:
  $\rho = | f^k / f^z - 1 |$
- A perfect clock has a **drift rate** ρ of 0
- drift rates vary due to changes in ambient temperature or ageing of crystal
- The data sheet of a resonator defines a maximum drift rate ρ_{max}.
- Due to the drift rate, clocks deviate from the reference clock over time if not resynchronized.

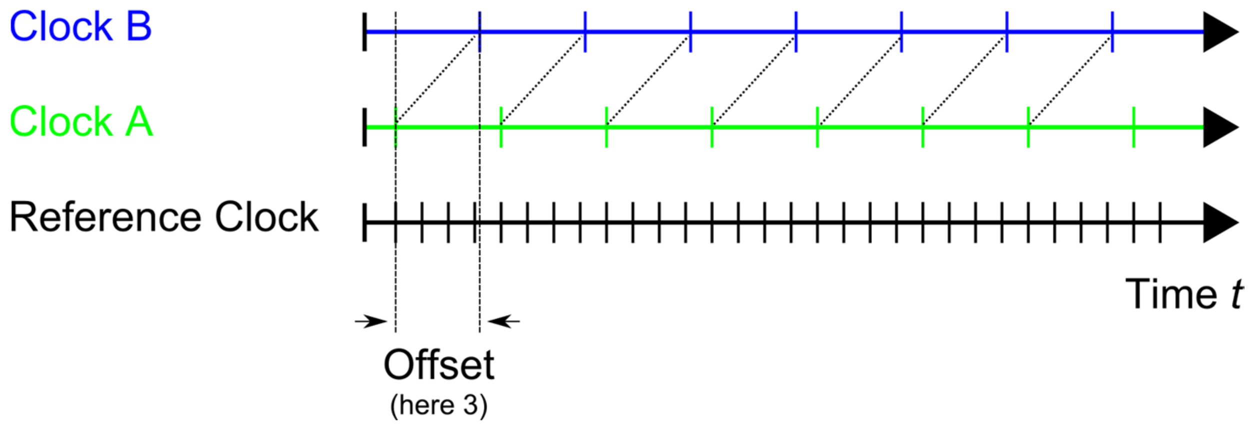
Clock Type	Drift Rate [s/s]
Quartz	10^{-5}
Pendulum	10^{-6}
Atom	$1.5 * 10^{-14}$
Atom (laser-cooled)	10^{-15}

Digital Physical Clocks: Failure Modes

- A digital physical clock can exhibit two types of failures:
  - The counter value could become erroneous (e.g. due to a overflow)
  - The drift rate could depart from the specified drift rate

Digital Physical Clocks: Offset

- Offset: The offset of two clocks is the time difference between the respective microticks of the two clocks – measured in the number of microticks of the reference clock.



Digital Physical Clocks: Precision & Internal Synchronization

- **Precision**: The precision Π denotes the maximum *offset* of respective microticks of an ensemble of clocks in a duration of interest and measured in microticks of the reference clock.
- Because of the drift rate ρ, an ensemble of clocks will drift apart if not resynchronized periodically. The process of mutual resynchronization is called **internal synchronization**.

Digital Physical Clocks: Accuracy

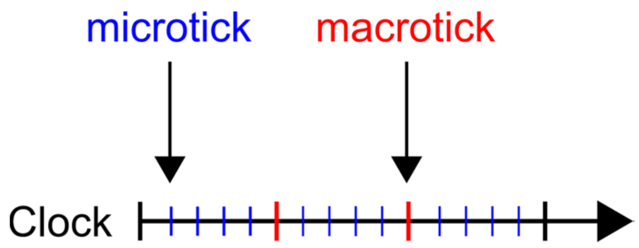
- **Accuracy**: The accuracy denotes the maximum *offset* of a given clock from the external time reference during a duration of interest.
- To keep a clock within a bounded accuracy it must be periodically resynchronized. This process is called **external synchronization**.
- Note: If all clocks of an ensemble are externally synchronized with an accuracy A, then the ensemble is also internally synchronized with a precision of $\leq 2A$
  $\rightarrow$ the converse is not true

Digital Physical Clocks: Time Standards

- A time base origin is called the ***epoch***.
- Three time standards are relevant for (distributed) real-time computer systems:
  1. **The International Atomic Time (TAI)**
     Defines the second as the duration of 9,192,631,770 periods of the radiaton of a specified transition of the cesium atom 133.
     Epoch: January 1, 1958 at 00:00 h (GMT). TAI is a *chronoscopic* timescale – a timescale without discontinuities)
  2. **The Universal Time Coordinated (UTC)**
     Replaced GMT (Greenwich Mean Time) in 1972. Not chronoscopic (**leap** seconds – one-second adjustment to keep the UTC close to the mean solar time).
  3. **UNIX (or POSIX) Time**
     Seconds since January, 1st 1970 (UTC) **not** counting leap seconds.

Global Time

- If all clocks of a distributed system are internally synchronized with precision Π, each **n-th** microtick of a clock can be interpreted as a *macrotick* to approximate a *global time*.



- The global time is called *reasonable* when the internal synchronization error is less than the duration between two consecutive macroticks $\Pi < g$ (i.e. the global time-stamps for a single event can differ by at most 1 tick).

Interval Measurement

- An *interval* is delimited by two events (e_{start} and e_{stop}).
- Interval measurement can be affected by:
  - the synchronization error
  - the digitalization error
- If the global time is reasonable, the interval error is always less than 2*g*, where *g* is the granularity of the global time.

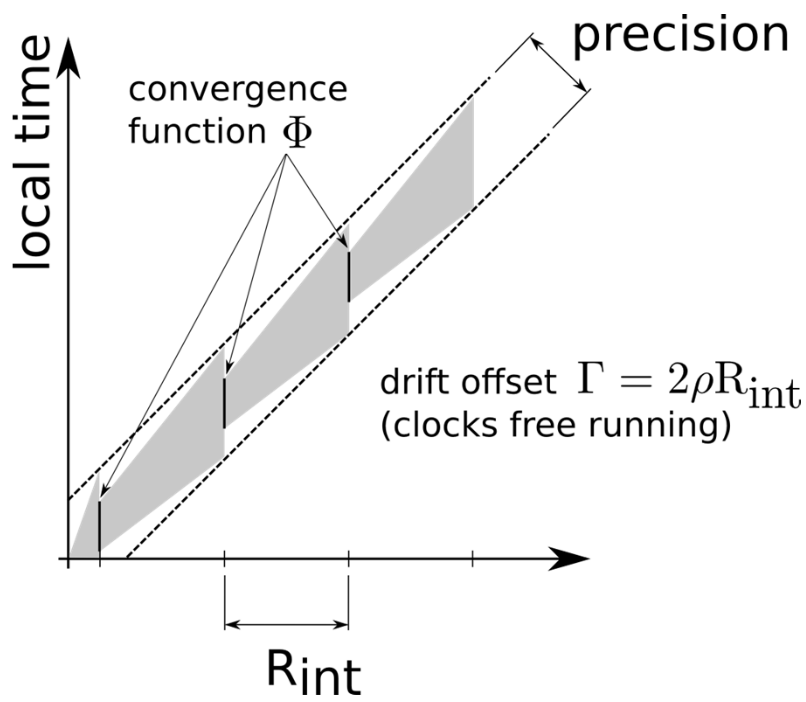
Summary: Fundamental Limits of Time Measurement

- In a distributed real-time system **with** a **global time base** (of granularity *g*), the following fundamental limits of measurements can be defined:

  1. The time-stamp of an event observed by two nodes can differ by one tick. This, however, is not sufficient to recover the temporal order of the events.
  2. The true duration *d* of an observed interval is bounded by +/- 2**g*.
  3. The temporal order of events can be recovered from their time-stamps if the difference between their time-stamps is equal or greater 2**g*

Internal Clock Synchronization

- Internal synchronization ensures that the global ticks of all nodes occur within a specified precision Π (despite the drift rate of each node).
- Resynchronization interval is called R_{int}
- The convergence function Φ denotes the offset after synchronization.
- The drift offset Γ indicates the maximum offset before synchronization.
- Synchronization condition: $\Phi + \Gamma \leq \Pi$



Internal Clock Synchronization

- Central Master Synchronization
  1. Master sends synchronization message with value of its time counter to all other nodes
  2. Slave records time-stamp when receiving synchronization message
  3. Slave computes deviation of its clock by taking the message transport latency into account and corrects its clock.
- Φ is determined by the fastest and slowest message transmission times (the latency jitter ε)
- The precision of the central master synchronization is: $\Pi_{central} = \varepsilon + \Gamma$
- Not fault tolerant: Failing master ends synchronization

Internal Clock Synchronization

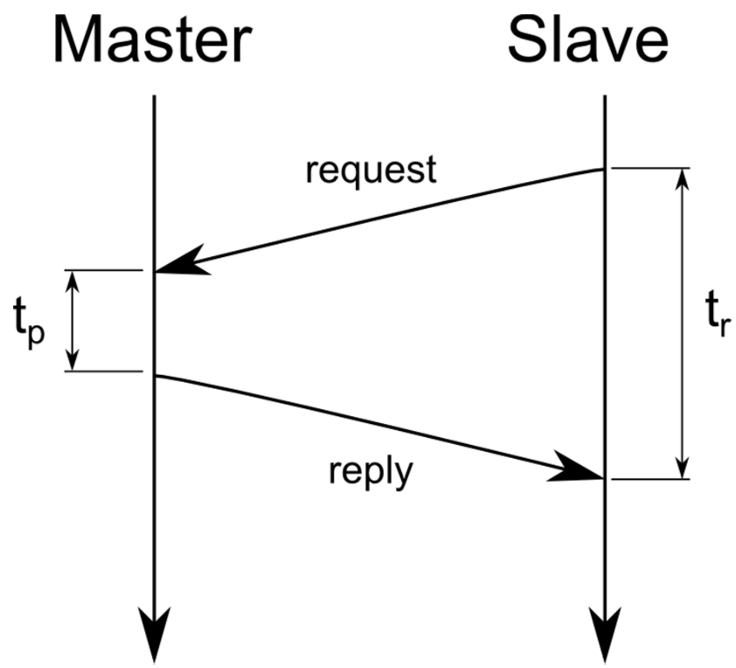
- Main term affecting the synchronization precision is the jitter ε.
- Delay jitter depends on system level of creation and interpretation of time synchronization message:

System Level	Jitter Range
Application	500 μs – 5 ms
Kernel	10 μs – 100 μs
Hardware	< 1 μs

- It is not possible to internally synchronize the clocks of an ensemble of *N* nodes to a better precision than:
  $\Pi = \varepsilon * (1-1/N)$

Internal Clock Synchronization: Cristian‘s Algorithm

- *S* requests the time from *M*
- On reception of the request from *S*, *M* prepares a response containing the time *T* from its own clock
- *S* then sets its time to be $T + t_r/2$
- Possible improvement:
  Take processing time t_p into account



Internal Clock Synchronization: State Correction vs. Rate Correction

- Based on correction term calculated by the convergence function the local time can be adjusted using:

State Correction	Rate Correction
• Correct local time immediately • Problem: Discontinuity in time (e.g. if clock is set backward, the same time value is reached twice)	• Correct the rate (speed) of the clock • Digital implementation: Change number of microticks per macrotick • Analog implementation: Change parameters of the crystal oscillator

External Clock Synchronization

- External clock synchronization **links** the **global time** of a distributed system **to** an **external time reference**
- Typically a designated node of the cluster, the **time gateway**, **receives** the **time** from the external time reference, **computes** the **rate correction** and **forwards** it to the **nodes**.

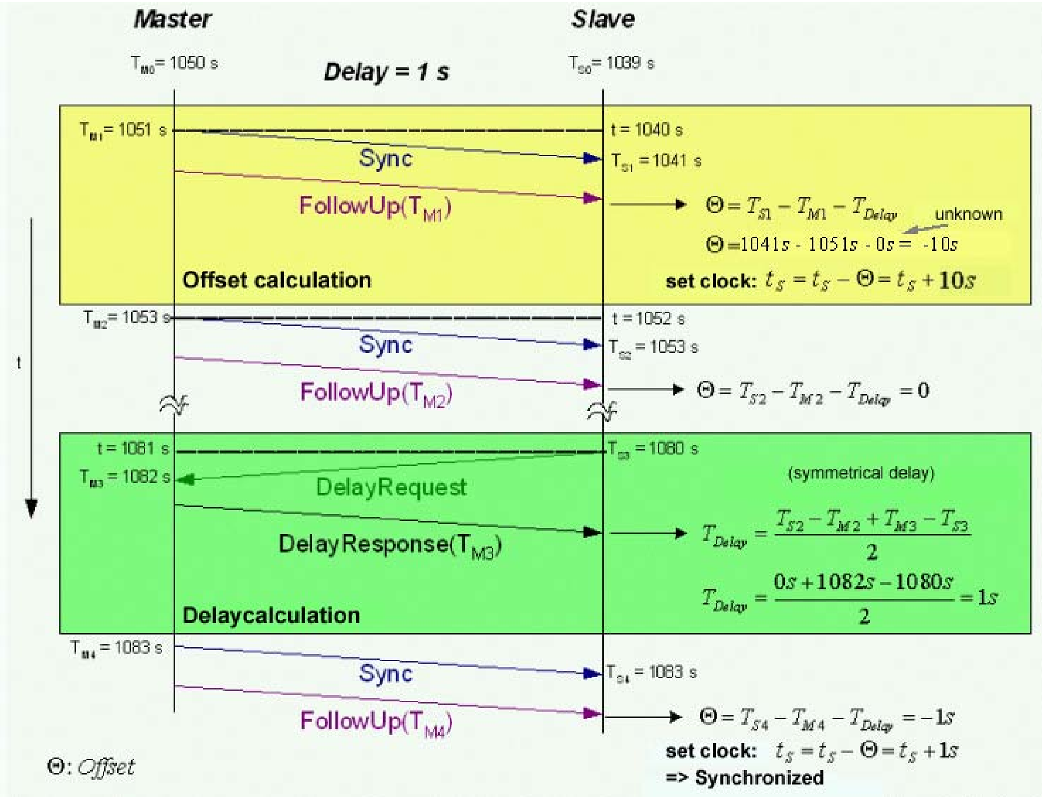
External Clock Synchronization: Time Formats

Protocol	Epoch	Format	Chronoscopic
Network Time Protocol (NTP)	January, 1st 1900, 00:00 h	4 Bytes for seconds 4 Bytes for fraction of seconds	No (based on UTC and therefore on leap seconds)
IEEE 1588	January, 1st 1970, 00:00 h	Seconds based on TAI Fraction of a second in nano seconds	Yes

- Time-Triggered Architecture (TTA) uses a mixture of NTP and IEEE 1588 as time format (full seconds based on TAI and parts of seconds as binary fraction) $\rightarrow$ chronoscopic and fully conformant to the dual system

External Clock Synchronization: IEEE 1588

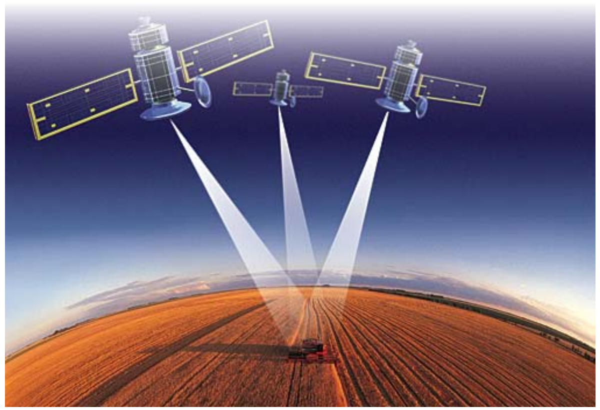
- IEEE 1588-2002 defines the Precision Time Protocol (PTP)
- Accuracy of < 1µs via Ethernet networks

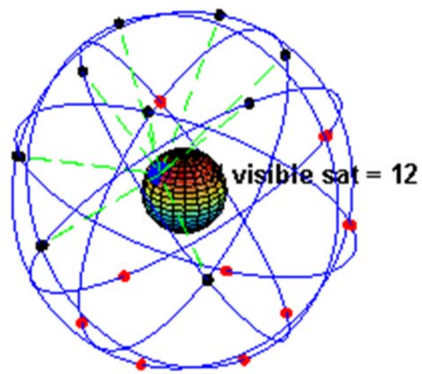


From :Precision Clock Synchronization, White Paper, Hirschmann

Example: GPS

- The Global Positioning System (GPS) was developed by the US Department of Defense
- Two services are provided:
    - Precise Positioning Service (PPS) – for military purposes
    - Standard Positioning Service (SPS) – for civilian purposes. Precision was purposely degraded (Selective Availability SA) before May 2, 2000.
- Accuracies in the range of cm possible with Differential Global Positioning System (DGPS)





Example: High-Speed Printing

- Paper runs at speeds of up to 100 km/h
- All printing stations (for different colours) must be synchronised so that the deviation between individual prints is less than 1μm
- Station rollers can be synchronised by coupling them mechanically by shafts
- Better: precise timing via synchronised clocks in each station

Literature

- H. Kopetz, Real-Time Systems, 2nd Edition, Chapter 3, 2011, Springer
- A. Tanenbaum, Distributed Systems: Principles and Paradigms, 2nd Edition, Prentice Hall
- Official U.S. Government information about GPS http://www.gps.gov/
- http://www.ieee1588.com
- http://www.ptb.de/cms/presseaktuelles/uhrzeitapplikation.html