

Distributed Operating Systems

6th Chapter
Clock Synchronization

Matthias Werner
Chair Operating Systems

6.1 Motivation

- In the previous chapter, we considered diverse semantics for **correct** orders
- In daily life, orders are usually defined through **time**
- Assumption of a **global time** that is read off from **clocks**

Global Time

- The determination of the time and the measurement of durations is vital for the coordination of human activities
- For this purpose, our clocks need to be synchronized
  - Synchronization of clocks via church clock, telegraphy, radio or GPS
  - It was clock synchronization that enabled the determination of longitudes in shipping
- **We have internalized the existence of a global time**



Bildquelle: NMM London, MoD Art Collection

Discussion!

The existence of a global time is a **model concept**, that orginates from Newton's physics. Other approaches, such as Minkowski spacetime[1] do not allow a global time.

[1] model conception(!) from theory of relativity

Time in Computers

- Usage of time stamps to ...
  - ... evaluate the actuality of data
  - ... conduct performance measurements
  - ... determine the validity of access permissions
  - ... achieve a total order of events (e.g. for synchronization, debugging and audit)
  - ... evaluate sensor data and control actuators

6.2 Time and Clocks

Terms

- A **clock** maps real time t onto a time stamp $C(t)$
- **Resolution**
  Smallest interval by which two values of a clock may differ
  ➡ **tick duration**
- **Offset**
  Deviation of the clock from real time at a certain point in time, meaning $C(t) - t$
- **Drift**
  Deviation of the **velocity** of the clock from the real time
  - $\pm 10^{-6}$ for quartz clocks regarding UTC (two seconds per month)
  - $\pm 10^{-18}$ for atomic clocks regarding the ideal SI-second

Local Clocks

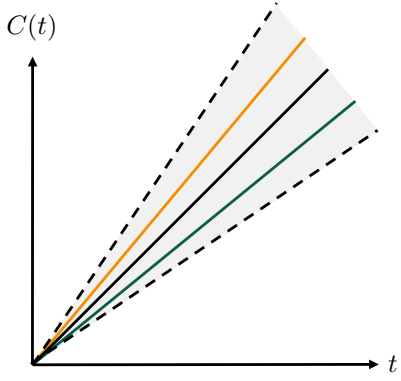
- Each local node has its own inaccurate, digital clock
- The clock drift is different
- Without synchronization, the values of the clocks may drift arbitrarly far apart ➡ **Clock synchronization** needed
- Only possible via **message passing**
- **Latency** of message transfers plays an important role
- **Internal synchronization**: participants in distributed system agree on a time, no connection to physical time
- **External synchronization**: time source outside of the system, take local time zone into account

Correct Clocks

- A **correct** clock has a limited drift ρ_{max}

$$\frac{1}{1+\rho_{max}} \leq \frac{dC(t)}{dt} \leq 1+\rho_{max} \quad (1)$$

- **Remark**: Drift is considered in scope Δt only, whereby Δt is larger than the time resolution, or one assumes a continuous clock, respectively



too fast, $\frac{dC}{dt} > 1$

accurate, $\frac{dC}{dt} = 1$

(Value of clock increases linearly with real time)

too slow, $\frac{dC}{dt} < 1$

- Clocks that **always** show the correct time are **perfect** clocks (model concept!)

Discussion

- Even a clock running too fast or too slow is considered a **correct** clock
- Condition is commonly stated as:

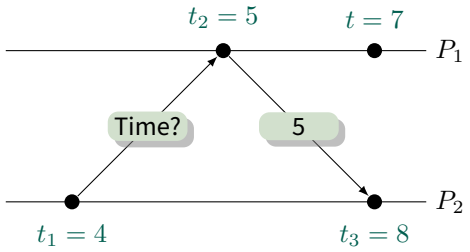
$$1 - \rho \leq \frac{dC(t)}{dt} \leq 1 + \rho \tag{2}$$

- This makes sense for very small drift ($\rho \ll 1$)
  - We consider $\rho = 1$
  - **Condition** (1): Clock is correct, as long as it is not running twice as fast or half as slow as the perfect clock
  - **Condition** (2): Clock is correct, as long as it is not running faster than twice as fast as the perfect clock, but could stand **still**
    - For $\rho > 1$, a correct clock could even run backwards
  - Howerver, for many relevant cases, (2) is sufficient

6.3 Internal Synchronization

6.3.1 Clock Synchronization by Cristian

- **Scenario**: Process P_2 likes to adjust its clock to P_1
- Preliminary assumption: constant message latency t_x
- P_2 adjusts its clock by $t_2 + t_x - t_3$
- If the maximum offset shall stay smaller than Δ, a resynchronization is necessary at the latest after $\frac{\Delta}{\rho}$

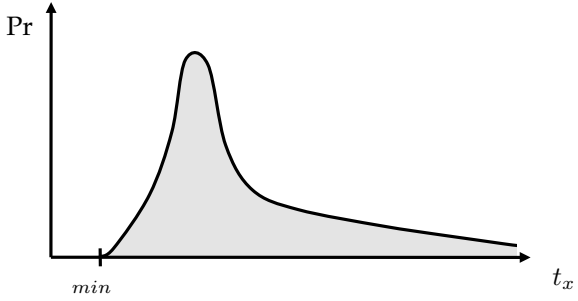


➡ P_2 sets its clock 1 time unit back

Unpredictable Message Latencies

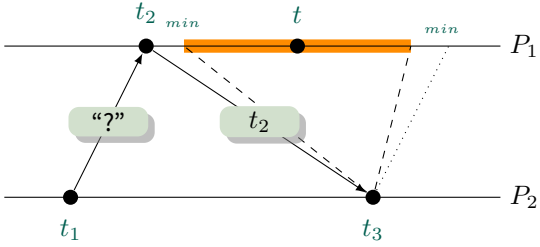
- In reality, latencies are mostly load-dependent and unlimited
  - Latency is for high loads higher than for low loads
  - Latency may be arbitrarily high
  - Minimal latency min may not be deceeded

- In this case, the previous method leads to an **error**



Unpredictable Message Latencies (cont.)

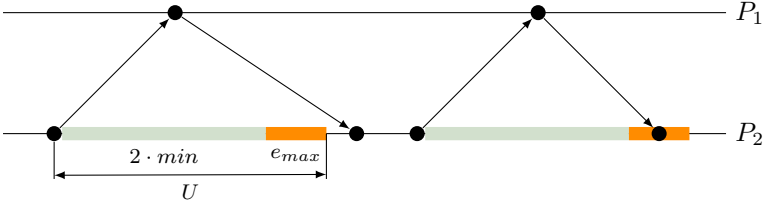
- **Round Trip Time** (RTT) $D \stackrel{def}{=} (t_3 - t_1)$
- Let t be the local time of P_1, if P_2 has local time t_3
- t is in interval $[t_2 + min, t_2 + D - min]$
- The "optimal" approximation for t is the center of the interval $t_2 + \frac{D}{2}$
- Hence, P_2 corrects its clock by $t_2 + \frac{D}{2} - t_3$
- The maximum error of this adjustment is $e_{max} = \frac{D}{2} - min$



Probabilistic Limitation of the Maximum Error

- **Idea:**
  1. Accept the value of the reference clock only, if the RTT D does not exceed a certain limit U ➡ $D \leq U$
  2. At a failed attempt, repeat the attempt at most k times; always after a waiting time of $t_W > U$

- Let $\Pr_U$ be the probability that $D > U$ holds for an arbitrary attempt (failed attempt)
- Probability for n failed attempts: $(\Pr_U)^n$
- Expectancy value for a number of failed attempts: $E = \frac{1}{1-(\Pr_U)}$



Maximum Synchronization Interval

- Two correct clocks with drift ρ shall not drift apart further than Δ
  ➡ After which time a synchronization has to take part at the latest?
- **Assumption:** at $t = 0$ clocks are synchronous
- **Considering the worst case:** maximum offset of the clocks ➡ clocks C_1 and C_2 are maximum slow/fast

$$\frac{dC_1(t)}{dt} = 1 + \rho \qquad \frac{dC_2(t)}{dt} = \frac{1}{1+\rho}$$

Worst case at time t:

$$C_1(t) = (1+\rho)t \qquad C_2(t) = \frac{t}{1+\rho}$$

Maximum Synchronization Interval (cont.)

Requirement: $C_1(t) - C_2(t) \overset{!}{\leq} \Delta$

Consequently:

$$(1+\rho)t - \frac{t}{1+\rho} \leq \Delta$$

$$\frac{(1+\rho)^2 t - t}{1+\rho} \leq \Delta$$

$$t\frac{2\rho+\rho^2}{1+\rho} \leq \Delta$$

Synchronization time

$$t \leq \Delta\frac{1+\rho}{2\rho+\rho^2}$$

- Hence, clocks need to be synchronizon done at the latest after $\Delta\frac{1+\rho}{2\rho+\rho^2}$
- For very small ρ: $t \approx \frac{1}{2\rho}\Delta$

Maximum Synchronization Interval (cont.)

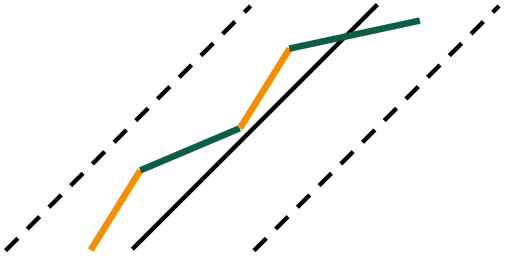
- We relinquish the assumption that clocks are synchronous at $t = 0$ ➡ Probabilistic synchronization, see Slide 6-12
  - After the last successfull synchronization is the offset maximal $e_{max} = \frac{D}{2} - min$
  - Synchronization needs to take place in such a way that no more time than $\Delta' = \Delta - e_{max}$ elapses till a successfull attempt
  - Since the synchronization procedure needs time itself, it has to be started early enough

Time until start next synchonization (after last one)

$$t \approx \frac{1}{2\rho}(\Delta - e_{max}) - \frac{t_W}{1 - \mathrm{Pr}_U} \tag{3}$$

Adjustment of Local Time

- **Requirements** (from correctness requirements)
  - Large leaps in time shall be avoided
  - The clock must not run backwards
- **Lösung**
  - The local clock is operated a bit slower or faster until the offset is compensated

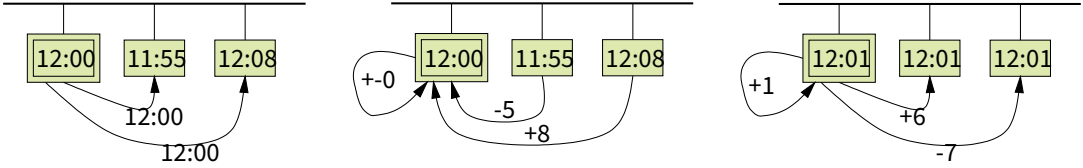


Berkley Algorithm

- Algorithm for **internal** clock synchronization
  - Synchronizes a set of clocks among eachother
  - Not necessarily connected to real time
  - Developed at University of California, Berkeley in 1989 [GuZa87]

- **Functionality**
  - System consists of several clocks
    - No assumption that one of the clocks provides the “true” time
  - If needed, election of a leader ➡ see Chapter 9
  - The Leader determines offset for each other clock ➡ calculation by means of the RTT as in Cristion’s algorithm
  - Correction by difference from average value and offset ➡ no transmission of absolute time
    - no additional error due to RTT

Berkley Algorithm (cont.)



$$Avg = \frac{0 - 5 + 8}{3} = 1$$
$$L = 1 - 0 = 1$$
$$F_1 = 1 - (-5) = 6$$
$$F_2 = 1 - 8 = -7$$

- To reduce the impact of faulty clocks, the algorithm ignores clock that shows a too large offset in time
- However, even these clocks get a correction value

Berkley Algorithm (cont.)

- Selection of a threshold for the offset
  - Sort out heavily deviant machines (averaging)
  - Too small: synchronization of a small subset of slaves only
  - Too large: faulty slaves decrease the precision of the synchronized time
  - Value of $20ms$ in the original publication from 1987

- An internal synchronisation should usually be oriented towards a “real” clock as well
- What is that?

6.4 External Synchronization

Global Time

- Oldest foundation of time measurement: Astronomy
- *UT - Universal Time*: Mean Solar Time
  - Determined through astronomical observation
  - A mean solar second is 1/86400 of a mean solar day
  - Follows the variations of the earth's rotation ➡ varying time unit
  - Solar day becomes 1.7ms longer each century (moon effects), 300 million years ago a year had 400 days

Global Time (cont.)

- *TAI - Temps Atomique International*: International Atomic Time
- Based on the frequency of the natural oscillation of the caesium 133 atom
- Weighted mean of about 260 atomic clocks around the world
- Standard deviations in the range of a few nano seconds
- Counting since 1.1.1958, increasing offset between UT and TAI
- Effects through atmosphere, radiation of black holes and gravitation
- Synchronization through measurement of GPS signals and satellite communication



Source: http://www.bipm.org/

Global Time (cont.)

- *UTC*: Coordinated Universal Time
  - *Temps universel coordonné* or *Koordinierte Weltzeit*
  - Length of UTC second is constant; defined by international atomic time
  - Leap seconds for compensation of UT-variations ($\Delta > 0.6s$)
  - Distribution via radio signal ($\pm 10ms$) or satellite ($\pm 0.5ms$)
  - Time zones relative to UTC

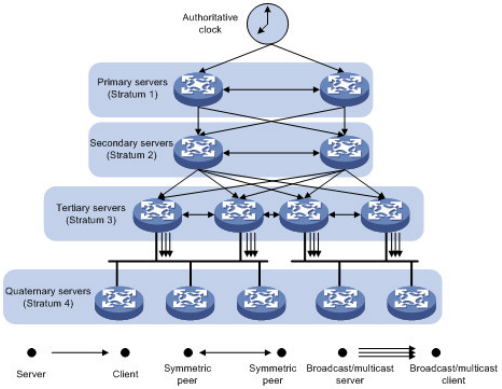


Source of image: Wikipedia

Network Time Protocol – NTP

- Protocol for the synchronization of (computer-) clocks via internet
- One of the oldest UDP protocols (RFC 958 from 1985), meanwhile version 4 (RFC 5905)
  - Mechanism for **external** clock synchronization
  - Large and varying delays of messages
  - Varying quality of time servers
  - Redundant communication paths
  - Scalability for a large number of requesting clients
  - Protection through authentication
- Tree from servers to end-user devices (`ntpq -pn`)
- Origin is UTC time source, e.g. an atomic clock
- For 99% of all NTP clients the error regarding the reference time is $\Delta < 30\,\text{ms}$

Network Time Protocol – NTP (cont.)



source of image: http://www.h3c.com.hk

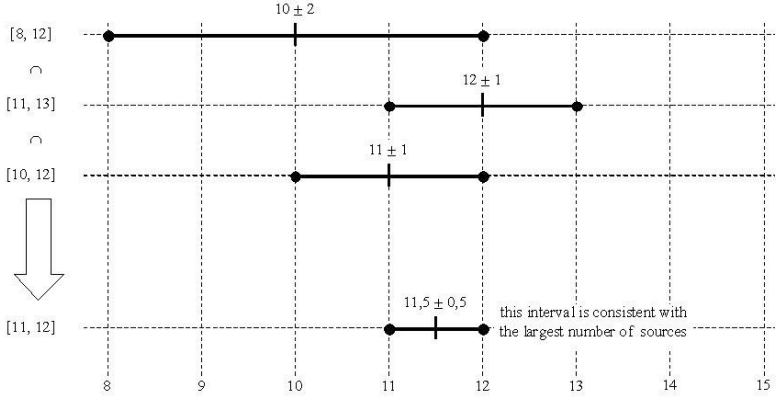
David Mills

"Boiled down, really the only thing the protocol does is occasionally read the clock in another machine, tell whether the system clock is fast or slow, and nudge the clock oscillator one way or the other, just like the Big Ben timekeepers in London. Of course, the devil is in the details"

- Synchronization within a level (stratum) similar to Berkeley algorithm
- Synchronisation between levels similar to Cristian's algorithm

NTP Detail: Marzullo's algorithm

- Partial algorithm for the determination of a time interval in case of overlapping sources
- **Assumption**: time sources offer time with confidence interval
- **Goal**: smallest interval with the largest amount of consistent sources



NTP Detail: Marzullo's algorithm (cont.)

- Time data as interval $[c - r, c + r]$
- Two tuples *(offset, type)* per source: $(c - r, -1)$ and $(c + r, 1)$
- Sort list of tuples by first value
- Subsequently:

```
INIT: best=0  /* largest number of overlapping intervals */
INIT: count=0 /* current number of overlapping intervals */
DO LOOP FOR ALL Tupels i in List
  count := count - type[i]
  IF count > best THEN
    best := count
    best_start := offset[i]
    best_end := offset[i+1]
  END IF
END LOOP
RETURN [best_start, best_end]
```

6.5 Limits of Synchronization

- How well can clocks be synchronized? ➡ [LuLy84]
- **Assumptions**:
  - Local clocks do not have a drift
  - Communication times in interval $[t, t + \varepsilon]$
  - Clock values are initially unknown

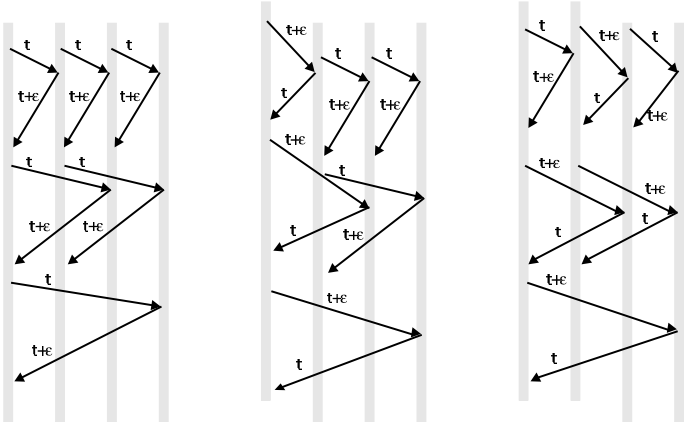
Theorem 1

There is no algorithm that guarantees synchronization of n clocks closer than an interval of size $\varepsilon(1 - \frac{1}{n})$.

- Border is hard: There are algorithms that guarantee a border.

Limits of Synchronization (cont.)

- Consider sequences in worst case
- Sequences e^i, $i = 1, \ldots, n$ are locally **not distinguishable**



Limits of Synchronization (cont.)

- Let C_i be the local clock value of node i and δ the deviation among eachother

$$C_n \leq C_1 + \delta \qquad \text{for sequence } e^1 \tag{4}$$

$$C_{i-1} + \varepsilon \leq C_i + \delta$$

$$\text{or } C_{i-1} \leq C_i + \delta - \varepsilon \qquad \text{for sequence } e^i,\ i = 2, \ldots, n \tag{5}$$

add (4) and (5)

$$\sum_{i=1}^{n} C_i \leq \sum_{i=1}^{n} C_i + n \cdot \delta - (n-1) \cdot \varepsilon$$

$$(n-1) \cdot \varepsilon \leq n \cdot \delta$$

$$\delta \geq \varepsilon \left(1 - \frac{1}{n}\right) \qquad \square$$

Incorrect Clocks

- **Incorrect clocks** may show any time – and that with every query!
  - Even leaps and backwards-running clocks possible
  - Every averaging fails
  - **Solution**: **Byzantine Agreement**
    ➡ Lecture “Dependable Systems”
  - **Requirement**: less than a third of all clocks may be faulty
- **Sample Application**: synchronization in FlexRay (communication platform for motor vehicles)

Literature

[Cri89] Cristian, F.: “Probabilistic Clock Synchronization”. *Distributed Computing*, 3(3)1989, 146–158

[GuZa87] Gusella, R. and S. Zatti: *The Accuracy of the Clock Synchronization Achieved by TEMPO in Berkeley UNIX 4.3BSD*, technical report, EECS Department, University of California, Berkeley

[LuLy84] Lundelius, J. and N. A. Lynch: “An Upper and Lower Bound for Clock Synchronization”. *Information and Control*, 62(2/3)1984, 190–204