Synchronization:
Physical clocks, logical clocks, algorithms

Konrad Iwanicki
University of Warsaw

Supplement for Topic 06: Synchronization
Distributed Systems Course
University of Warsaw

Based on various sources: see the last slide.

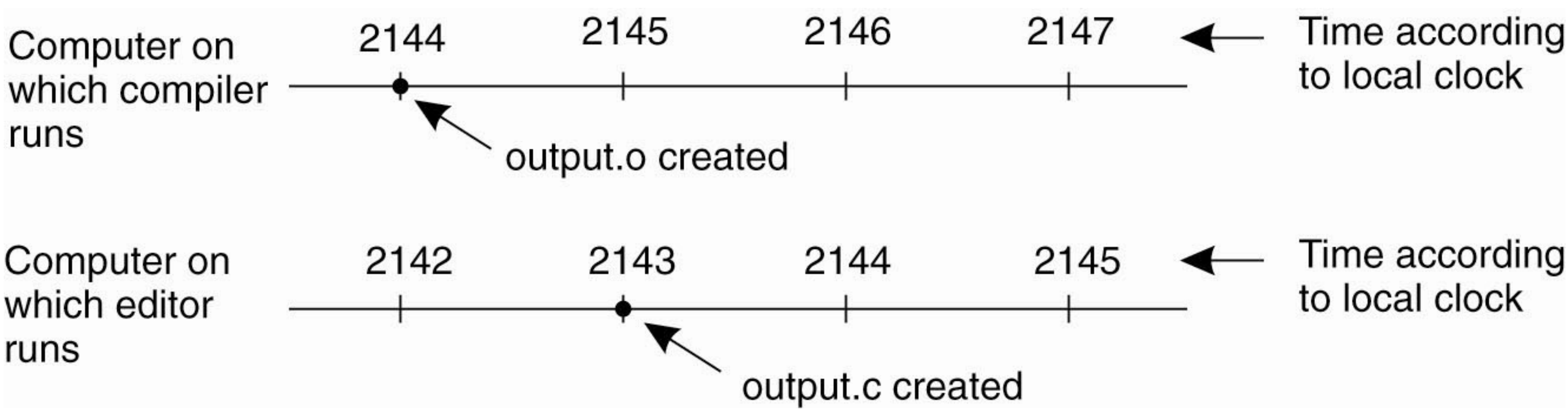
Introduction

- In a centralized (single-node) system time is unambiguous:
  - Process P_A asks for time and gets T_A.
  - Later, process P_B asks for time and gets T_B.
  - For sure, $T_A \leq T_B$.
  - In other words, P_A and P_B always agree on the current time.
- This fact is made use of in various cases:
  - e.g., the *make* tool

Introduction

- Achieving agreement on time in a distributed system is not trivial.
- In some cases, a lack of such an agreement can have grave consequences.

Introduction



Introduction

- There are many cases in which agreeing on time is important:
  - Financial brokerage
  - Security auditing
  - Collaborative sensing
- In general, people analyze events wrt time.

Introduction

- There are many cases in which agreeing on time is important:
  - Financial brokerage
  - Security auditing
  - Collaborative sensing
- In general, people analyze events wrt time.
- Is it possible to synchronize all the clocks in a distributed system?

Physical clocks

- Each computer has a so-called timer:
  - A quartz oscillator with two registers.
- A counter register is decremented on each oscillation.
- When it goes to zero,
  - it is reloaded with the value from a holding register.
  - a clock interrupt is generated => the clock ticks.

Physical clocks

- Each computer has a so-called timer:
  - A quartz oscillator with two registers.
- A counter register is decremented on each oscillation.
- When it goes to zero,
  - it is reloaded with the value from a holding register.
  - a clock interrupt is generated => the clock ticks.

- Effect: we can make the clock tick every second to maintain time for our computer.

Physical clocks

- However, with multiple clocks the situation changes.
- Timers are imperfect oscillators:
  - N computers => N different oscillation frequencies

Physical clocks

- However, with multiple clocks the situation changes.
- Timers are imperfect oscillators:
  - N computers => N different oscillation frequencies

- How do we keep them in sync with each other?
- How do we keep them in sync with the external world (the real time)?

Measuring time

- In the past, time was measured astronomically:
  - Solar day = the period between two consecutive appearances of the sun at the peek point in the sky
  - Solar second = 1 / (24 * 60 * 60) of a solar day

Measuring time

- Solar day is not constant!
  - Permanent changes in the Earth's rotation speed:
    - Days are getting longer.
  - Temporal variations.

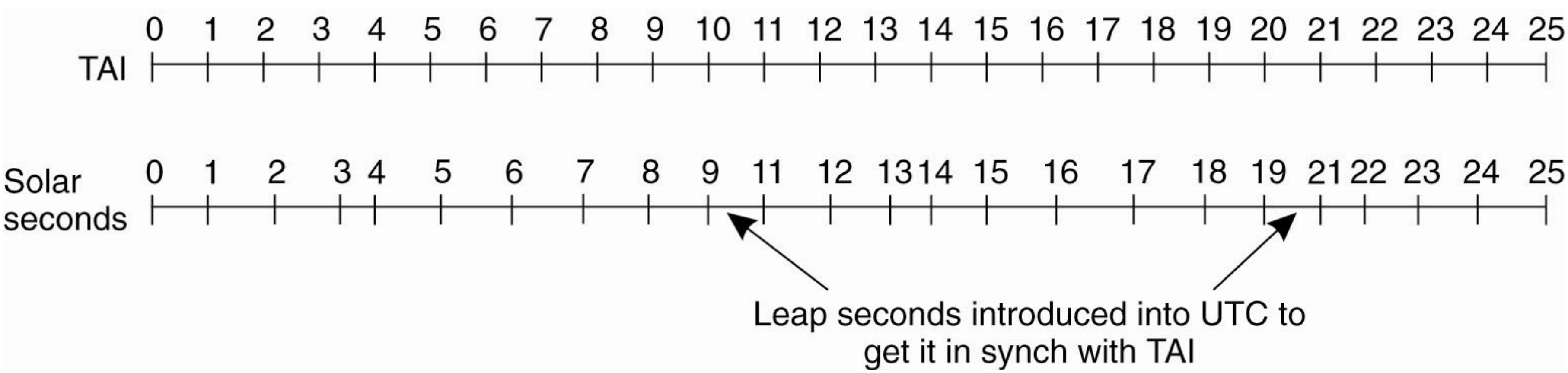
Measuring time

- Atomic clocks can provide accurate time
  - Idea: counting the number of transitions of the cesium 133 atom (earlier also rubidium 87 and thallium 205).
  - 1 second = 9,192,631,770 transitions
- Several laboratories have atomic clocks
- Periodically, they inform the International Time Bureau about the number of ticks
- The average is known as International Atomic Time (TAI)

Measuring time

  - TAI is highly stable.
  - Solar day is getting longer.
- => 86,400 TAI seconds is now about 3 ms less than a mean solar day
- Tolerating this discrepancy = bad idea.
- Solution: leap seconds.

Measuring time



- This correction is a base of Universal Coordinated Time (UTC).

Obtaining UTC

- Most electric companies synchronize the timing of their 60-Hz or 50-Hz clocks to UTC.
- Shortwave pulses at the start of every second:
  - NIST, Fort Collins, CO, USA
  - MSF, Rugby, England

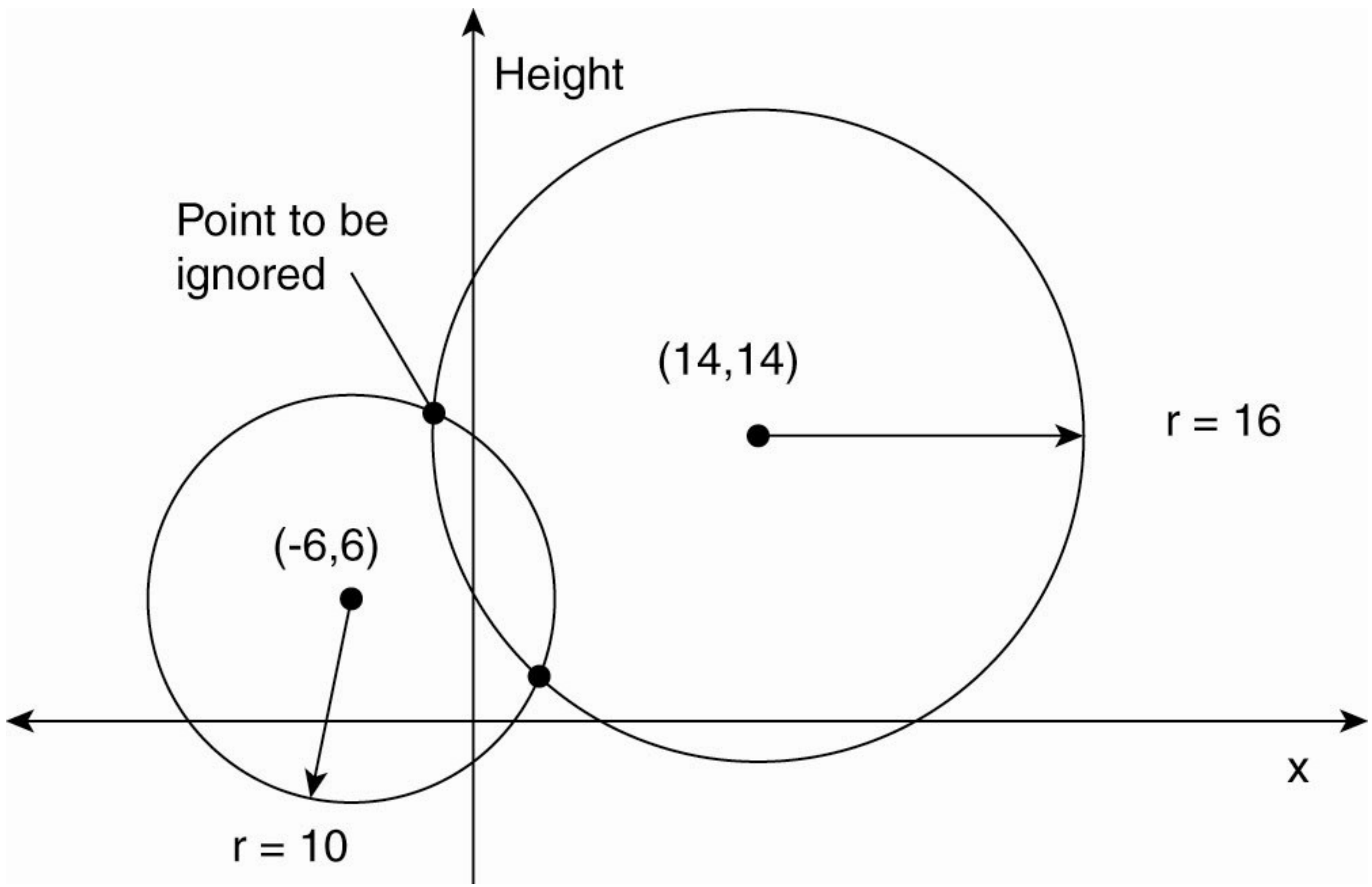
  Accuracy: ± 1 ms (broadcaster), ± 10 ms (recv)
- Earth satellites also offer UTC:
  - GEOS

  Accuracy: ± 0.5 ms

Global Positioning System

- Global Positioning System (GPS) offers time synchronization as a by-product:
  - 29 satelites
  - At ~20,000 km
- Each satellite has up to 4 atomic clocks.
- The clocks are calibrated from stations on Earth.
- Each satellite continuously broadcasts its position and local time.

Global Positioning System



Global Positioning System

- Problem: Assuming that the clock's of satellites are accurate and synchronized:

Global Positioning System

- Problem: Assuming that the clock's of satellites are accurate and synchronized:
  - It takes a while before a satellite's position reaches a GPS receiver.

Global Positioning System

- Problem: Assuming that the clock's of satellites are accurate and synchronized:
  - It takes a while before a satellite's position reaches a GPS receiver.
  - The receiver's clock need not be in sync with the satellite's clock.

Global Positioning System

- Principal operation:
  - Δ_r : unknown deviation of the receiver's clock
  - x_r, y_r, z_r : unknown coordinates of the receiver's clock
  - T_i : timestamp on a message from satellite i
  - $\Delta_i = (T_{now} - T_i) + \Delta_r$: measured delay of the message sent by satellite i
  - Δ_i x c : measured distance to satellite i
  - Real distance is:

$$d_i = c\,\Delta_i - c\,\Delta_r = \sqrt{\left((x_i - x_r)^2 + (y_i - y_r)^2 + (z_i - z_r)^2\right)}$$

  - 4 satellites = 4 equations with 4 unknowns

Global Positioning System

- Principal operation:
  - Δ_r : unknown deviation of the receiver's clock

Global Positioning System

- Principal operation:
  - Δ_r : unknown deviation of the receiver's clock
  - x_r, y_r, z_r : unknown coordinates of the receiver's clock

Global Positioning System

- Principal operation:
  - Δ_r : unknown deviation of the receiver's clock
  - x_r, y_r, z_r : unknown coordinates of the receiver's clock
  - T_i : timestamp on a message from satellite i

Global Positioning System

- Principal operation:
  - Δ_r : unknown deviation of the receiver's clock
  - x_r, y_r, z_r : unknown coordinates of the receiver's clock
  - T_i : timestamp on a message from satellite i
  - $\Delta_i = (T_{now} - T_i) + \Delta_r$: measured delay of the message sent by satellite i

Global Positioning System

- Principal operation:
  - Δ_r : unknown deviation of the receiver's clock
  - x_r, y_r, z_r : unknown coordinates of the receiver's clock
  - T_i : timestamp on a message from satellite i
  - $\Delta_i = (T_{now} - T_i) + \Delta_r$: measured delay of the message sent by satellite i
  - $\Delta_i \times c$: measured distance to satellite i

Global Positioning System

- Principal operation:
  - Δ_r : unknown deviation of the receiver's clock
  - x_r, y_r, z_r : unknown coordinates of the receiver's clock
  - T_i : timestamp on a message from satellite i
  - $\Delta_i = (T_{now} - T_i) + \Delta_r$: measured delay of the message sent by satellite i
  - Δ_i x c : measured distance to satellite i
  - Real distance is:

$$d_i = c\,\Delta_i - c\,\Delta_r = \sqrt{\left((x_i - x_r)^2 + (y_i - y_r)^2 + (z_i - z_r)^2\right)}$$

Global Positioning System

- Principal operation:
  - Δ_r : unknown deviation of the receiver's clock
  - x_r, y_r, z_r : unknown coordinates of the receiver's clock
  - T_i : timestamp on a message from satellite i
  - $\Delta_i = (T_{now} - T_i) + \Delta_r$: measured delay of the message sent by satellite i
  - Δ_i x c : measured distance to satellite i
  - Real distance is:

$$d_i = c\,\Delta_i - c\,\Delta_r = \sqrt{\left(\left(x_i - x_r\right)^2 + \left(y_i - y_r\right)^2 + \left(z_i - z_r\right)^2\right)}$$

  - 4 satellites = 4 equations with 4 unknowns

Global Positioning System

- The measurements are not accurate.
  - GPS does not consider leap seconds.
  - Atomic clocks of satellites are not in perfect sync.
  - The position of a satellite is not known precisely.
  - The receiver's clock has a finite accuracy.
  - Signal propagation is not constant.
  - Earth is not a perfect sphere.
- Computing a position and time is far from trivial.
- Nevertheless, GPS offers good accuracy:
  - Professional receivers: 20-35 nanosecs.

Time synchronization

- Suppose that one computer has a shortwave time pulse receiver.
- The goal is to synchronize other machines with the time provided by the receiver...

Time synchronization

- Suppose that one computer has a shortwave time pulse receiver.
- The goal is to synchronize other machines with the time provided by the receiver...
- … and then, to keep the machines in sync.

Time synchronization

- Assumptions:
  - Each machine, P, has a timer that ticks H times per second.

Time synchronization

- Assumptions:
  - Each machine, P, has a timer that ticks H times per second.
  - The timer is used as a base of P's clock that ticks on each interrupt. Let's denote the value of this clock at UTC time t as $C_p(t)$.

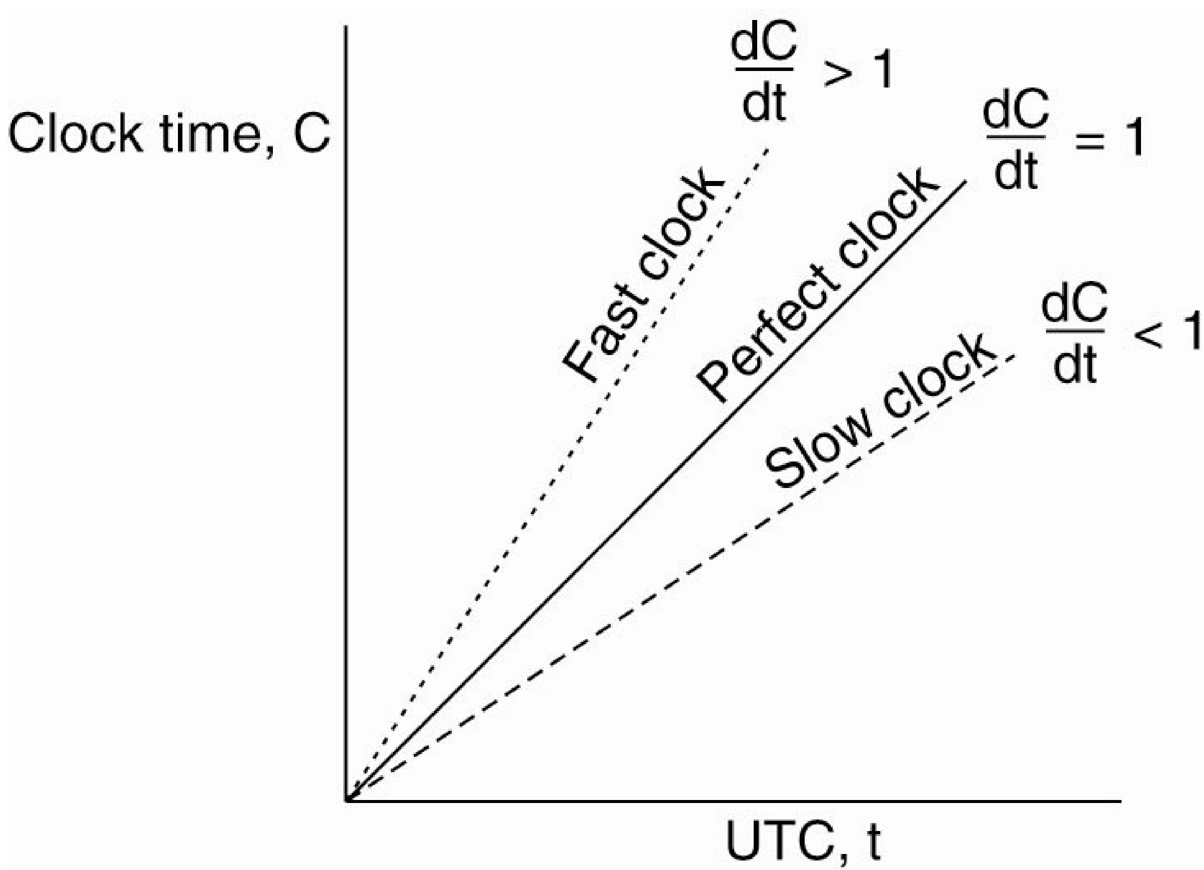
Time synchronization

- Assumptions:
  - Each machine, P, has a timer that ticks H times per second.
  - The timer is used as a base of P's clock that ticks on each interrupt. Let's denote the value of this clock at UTC time t as $C_p(t)$.
- Ideally, we would like to have $C_p(t) = t$, that is:
  - $dC / dt = 1$.

Time synchronization

- Real timers do not interrupt exactly H times per second.
  - In theory, with H = 60, we should have 216,000 ticks per hour.
  - In practice, with modern oscillators, the relative error is about 10^{-5}:
    - Between 215,998 and 216,002 ticks per hour.
- Clock skew = $C_p(t) - 1$

Time synchronization



Time synchronization

- In practice, for a given clock, there exists a maximum drift rate, ρ:

$$1 - \rho \leq dC / dt \leq 1 + \rho$$

Time synchronization

- In practice, for a given clock, there exists a maximum drift rate, ρ:

  $1 - \rho \leq dC / dt \leq 1 + \rho$

- Goal: Never let two clocks drift more than δ time units.

Time synchronization

- In practice, for a given clock, there exists a maximum drift rate, ρ:

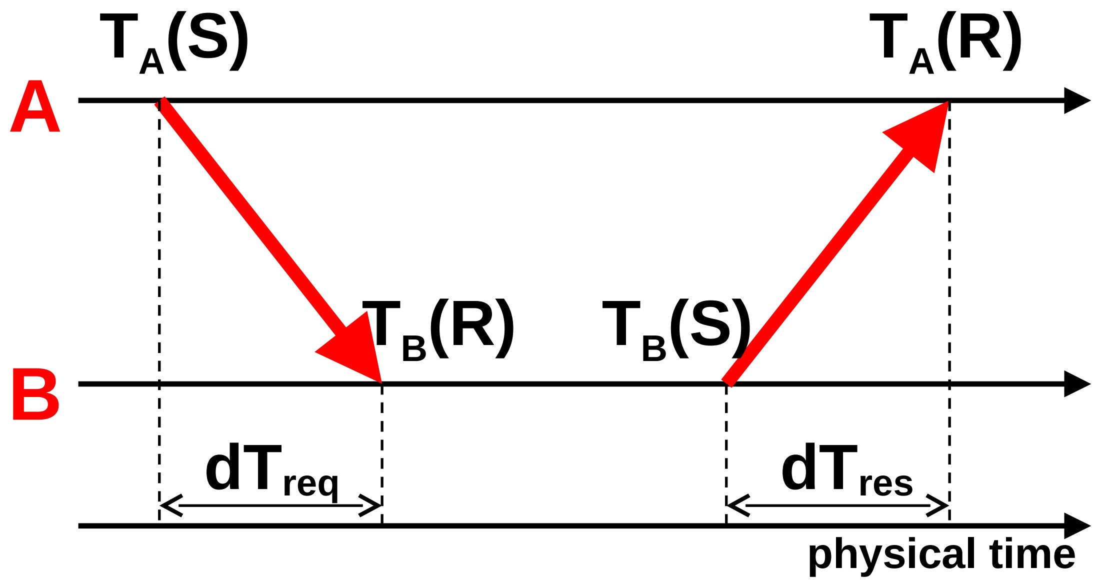
  $1 - \rho \leq dC / dt \leq 1 + \rho$

- Goal: Never let two clocks drift more than δ time units.
- Solution: Resynchronize at least every $\delta / (2\rho)$ time units.

Time synchronization

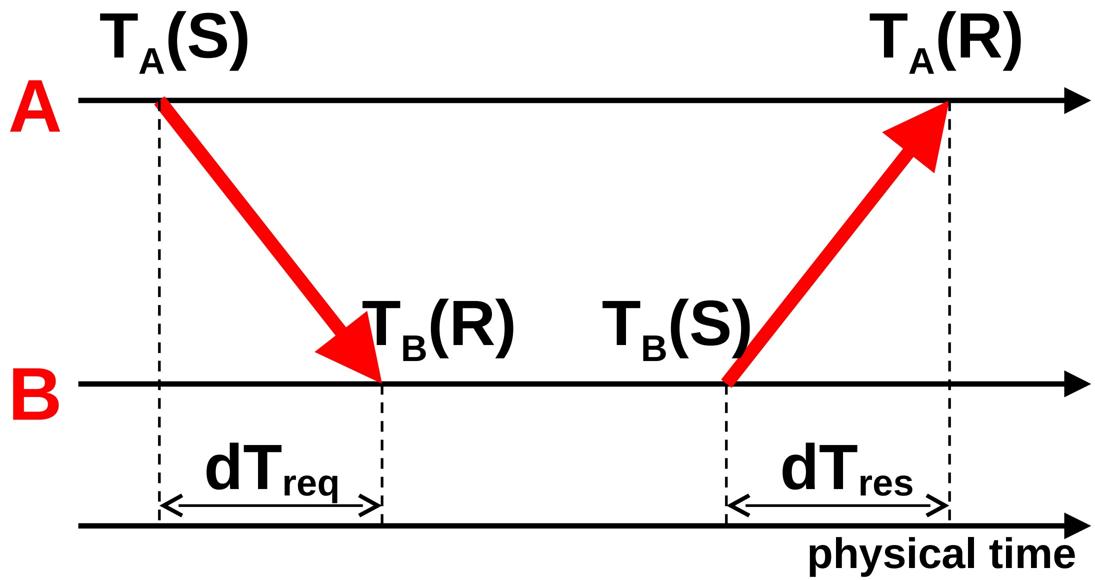
- Approach I:
  - Every machine asks a time server for the current time at least every $\delta / (2\rho)$ time units (Network Time Protocol – NTP).

Time synchronization



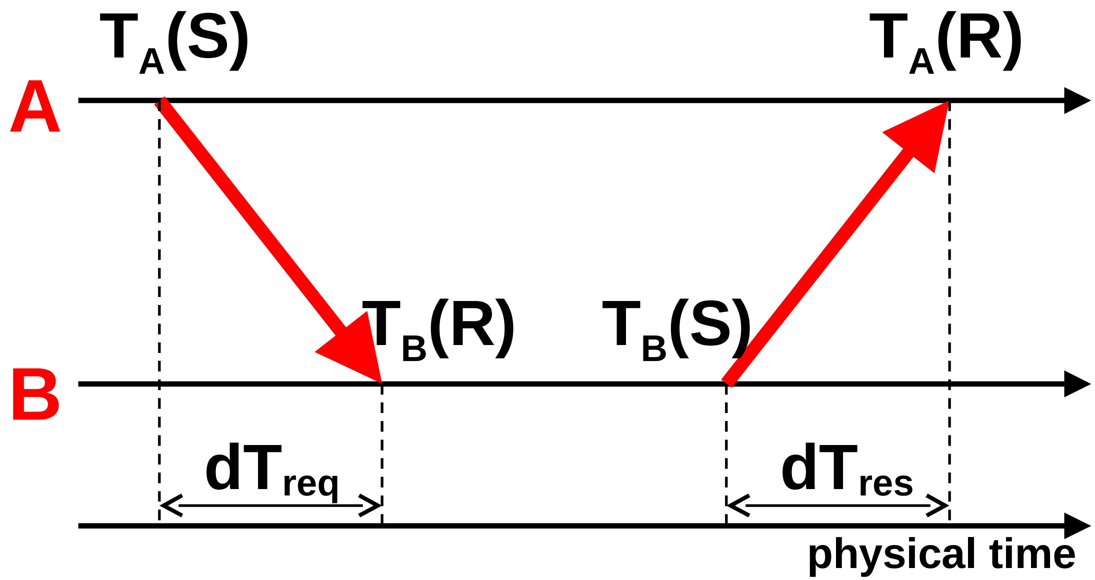
- Assuming $dT_{req} = dT_{res} = 0$, A's offset from B:
  - $\theta = T_B(S) - T_A(R)$

Time synchronization



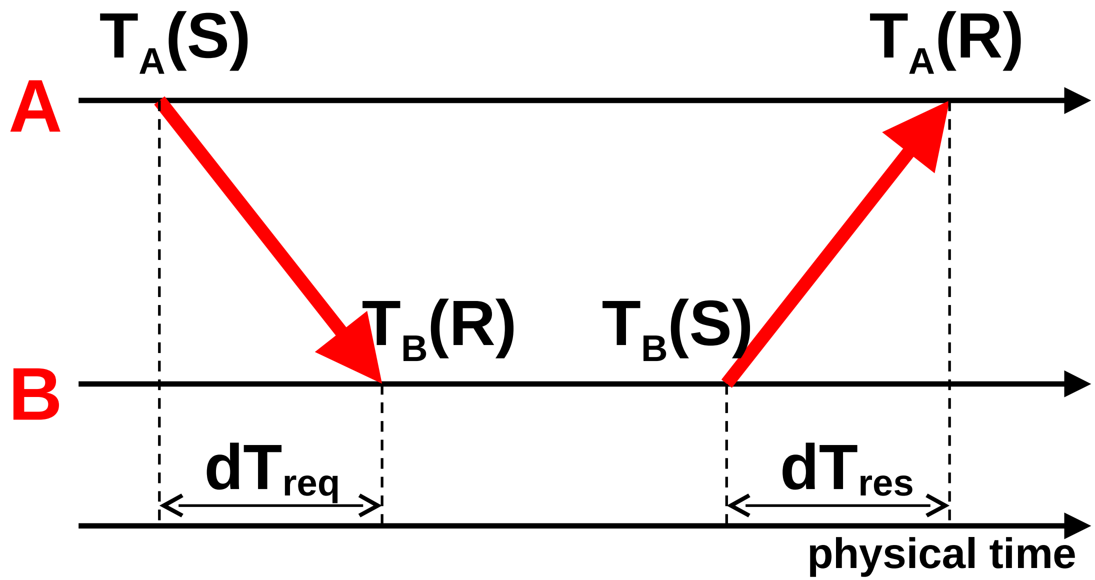
- Assuming $dT_{req} = dT_{res} = 0$, A's offset from B:
  - $\theta = T_B(S) - T_A(R)$
- In practice, dT_{req} , $dT_{res} > 0$

Time synchronization



- Assuming $dT_{req} = dT_{res} = 0$, A's offset from B:
  - $\theta = T_B(S) - T_A(R)$
- In practice, dT_{req} , $dT_{res} > 0$
- Problem: How to estimate the offset?

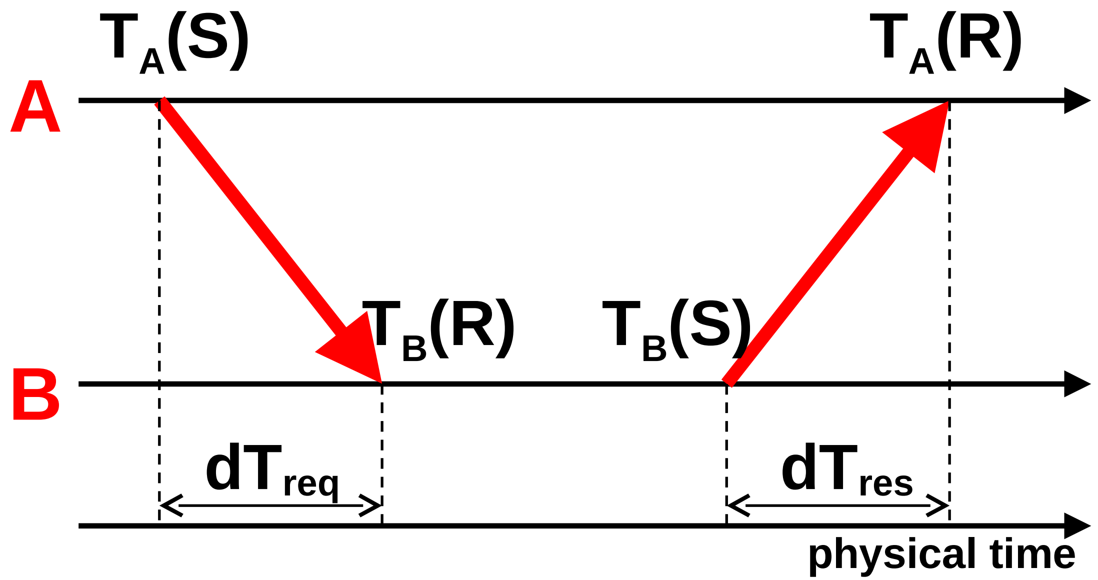
Time synchronization



- Round-trip delay:

$$\boldsymbol{\delta} = T_A(R) - T_A(S) - (T_B(S) - T_B(R))$$

Time synchronization

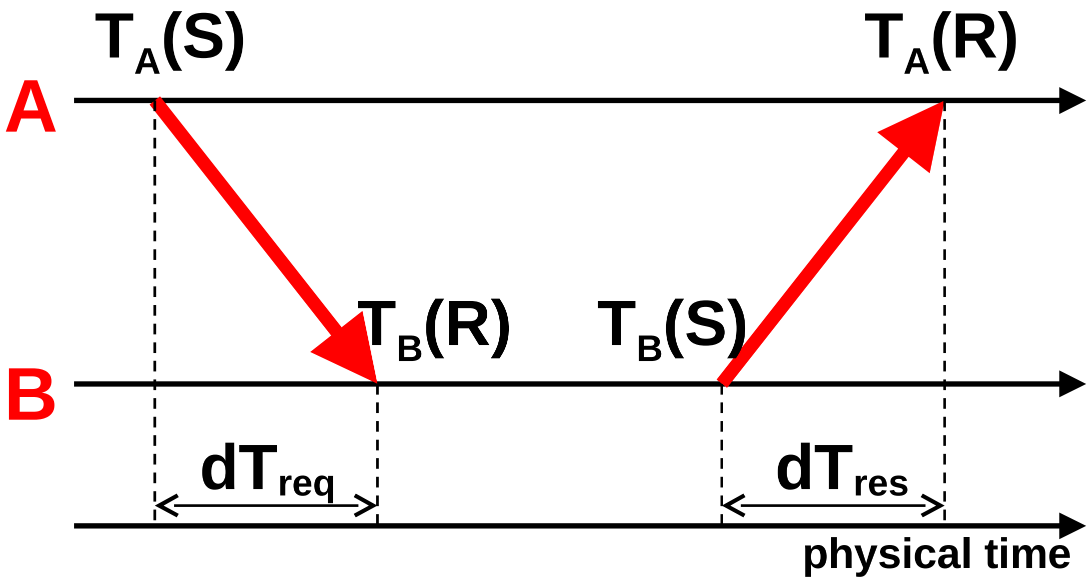


- Round-trip delay:

$$\delta = T_A(R) - T_A(S) - (T_B(S) - T_B(R))$$

- Assume $dT_{req} = dT_{res}$

Time synchronization



- Round-trip delay:

  $$\delta = T_A(R) - T_A(S) - (T_B(S) - T_B(R))$$

- Assume $dT_{req} = dT_{res}$
- Time offset: $\theta = T_B(S) + \frac{1}{2} \times \delta - T_A(R)$

Time synchronization

- Assuming $dT_{req} = dT_{res}$ introduces errors.
- The reasons for errors:
  - Network delays
  - Interrupt handling
  - OS delays
  - Message processing

Time synchronization

- NTP:
  - estimates errors using round trip delays.
  - rejects samples that suffer from large errors.
  - divides servers into strata:
    - Stratum 0: an atomic clock
    - Stratum 1: a machine with shortwave time pulse receiver
    - Stratum i + 1: a machine that obtained its time from synchronizing with a stratum-i machine
- NTP's accuracy (world-wide): 1-50 ms
- Stratum-less synchronization: Gossiping Time Protocol (GTP).

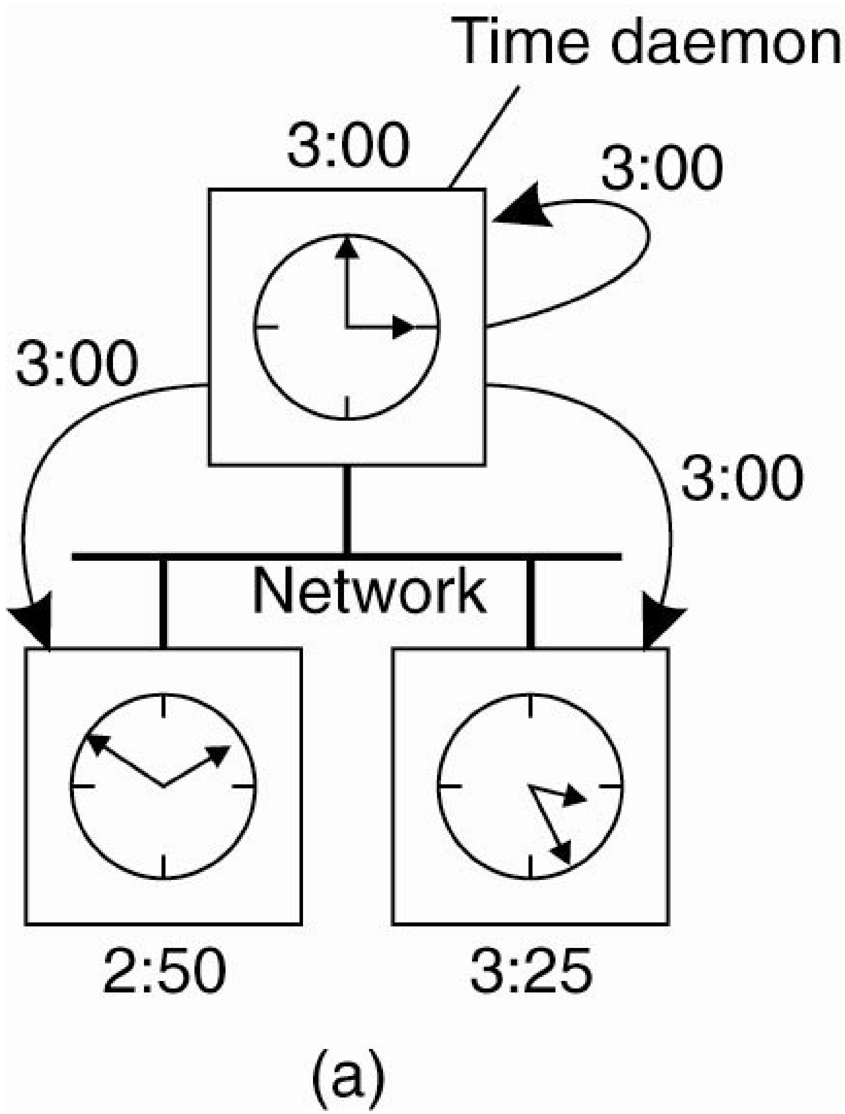
Time synchronization

- Approach II:
  - NTP provides external synchronization (to a stratum-0 clock).
  - An alternative is internal synchronization:
    - Machines synchronize with each other.
    - Not necessarily with an external clock.

Time synchronization

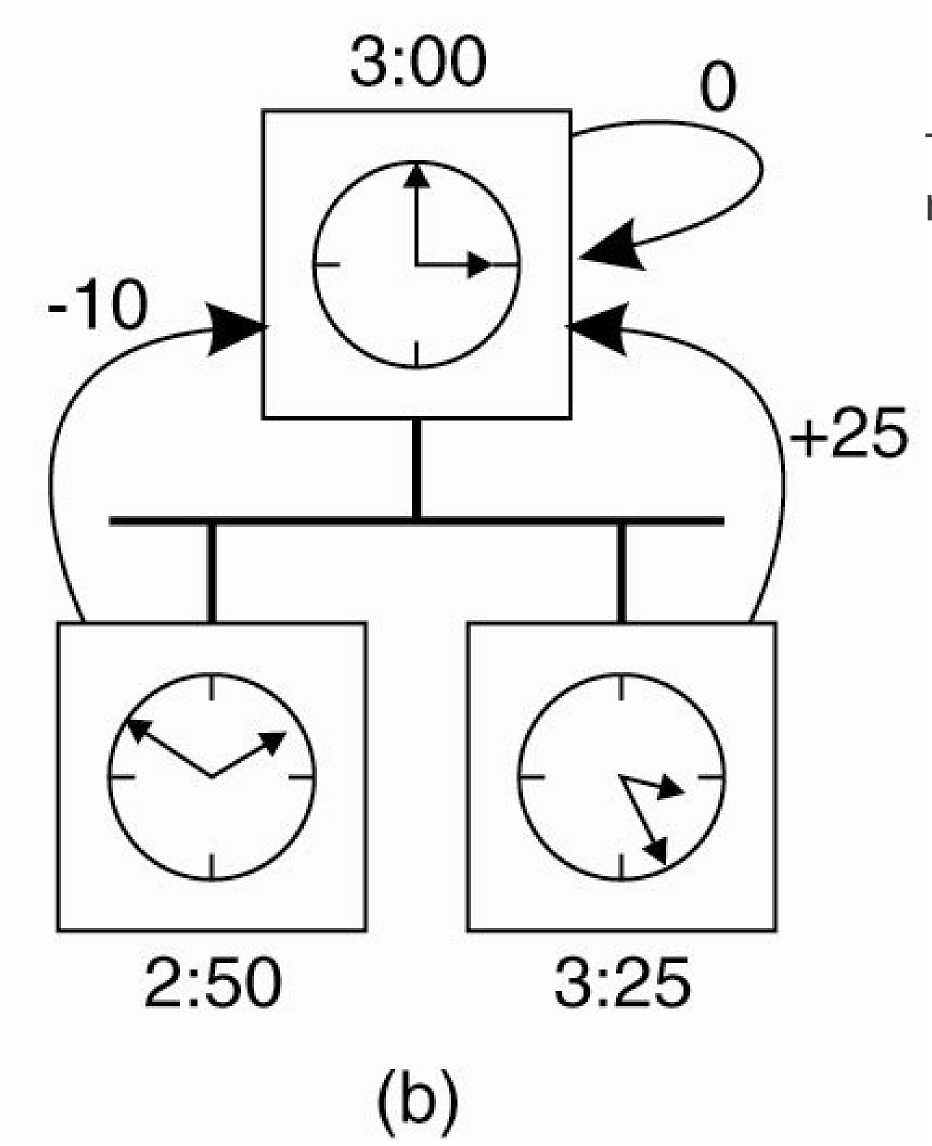
- The Berkeley algorithm:
  - Works in a local area network.
  - A special process, time daemon is responsible for synchronizing clocks of different machines.

Time synchronization



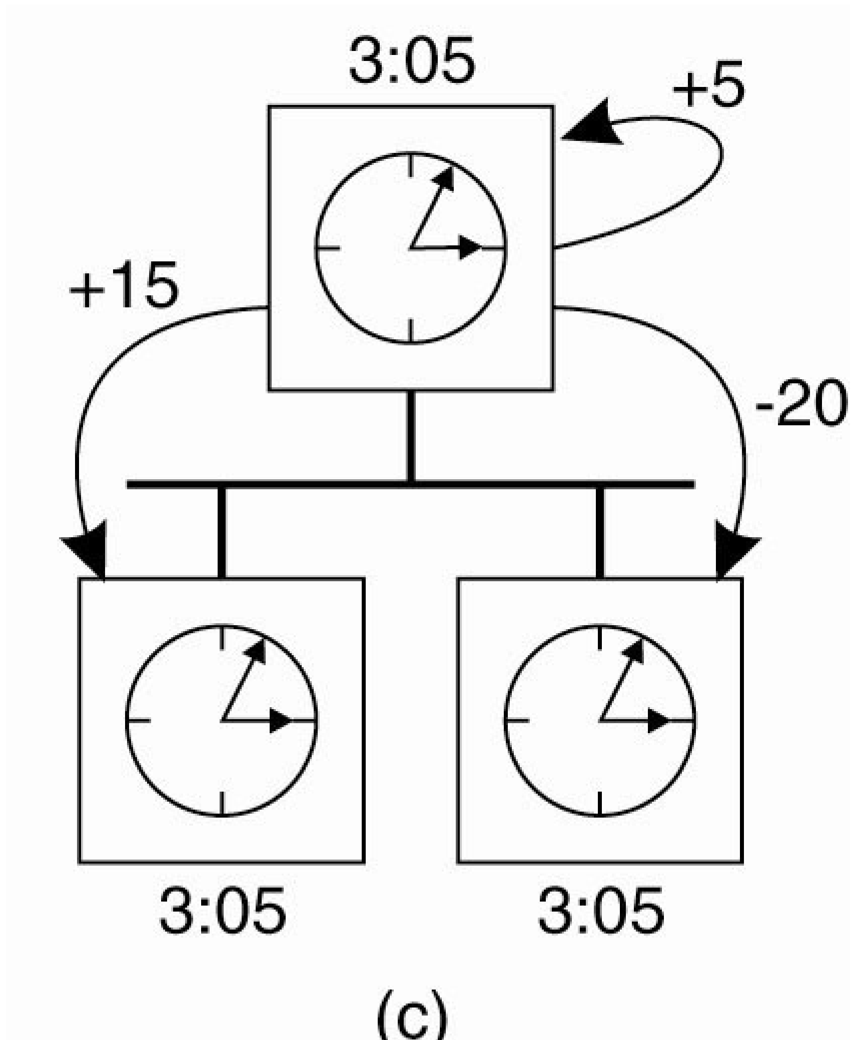
The time daemon periodically asks other machines for their local time.

Time synchronization



The machines
reply with their offsets.

Time synchronization



The time deamon
tells each machine
how to adjust its clock.

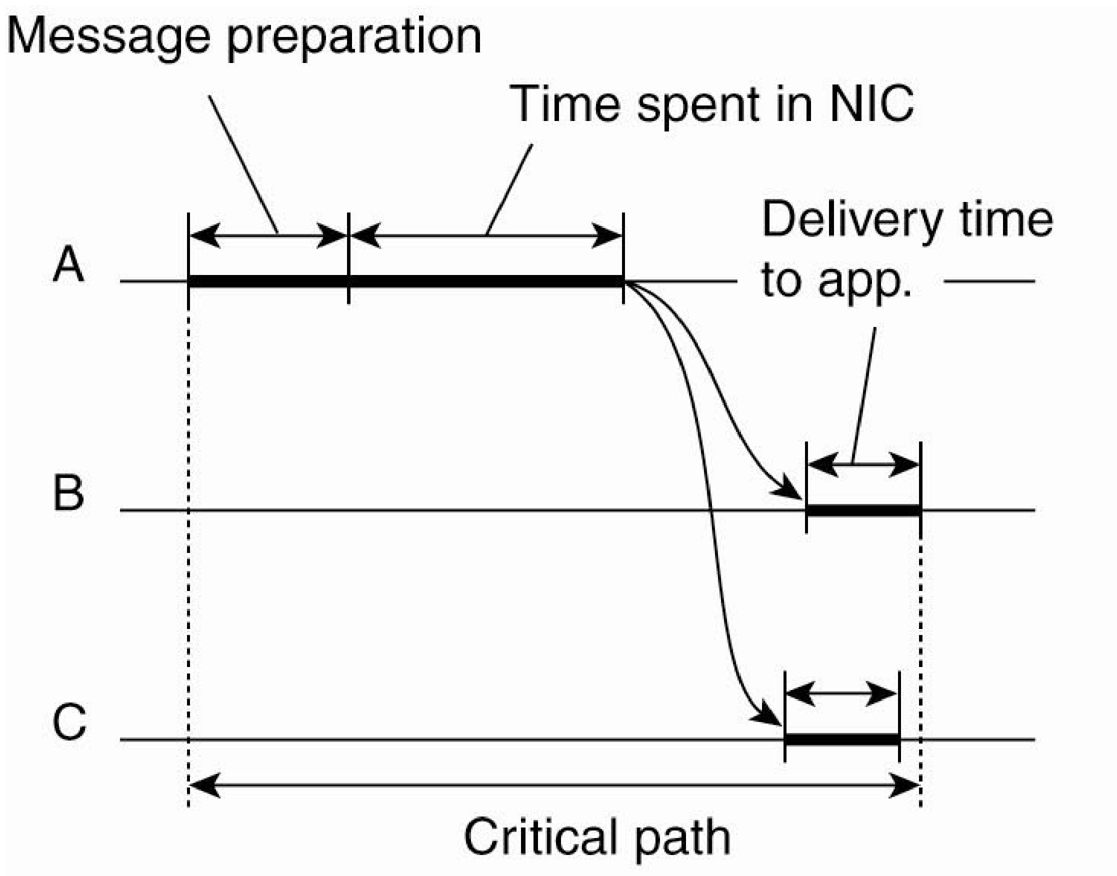
Time synchronization

- Approach III:
  - Wireless sensor networks require tight time synchronization:
    - e.g., seismic activity monitoring
  - On the other hand, they are built of inexpensive hardware.
  - Special algorithms are necessary.
    - e.g., Reference Broadcast Synchronization (RBS)

Time synchronization

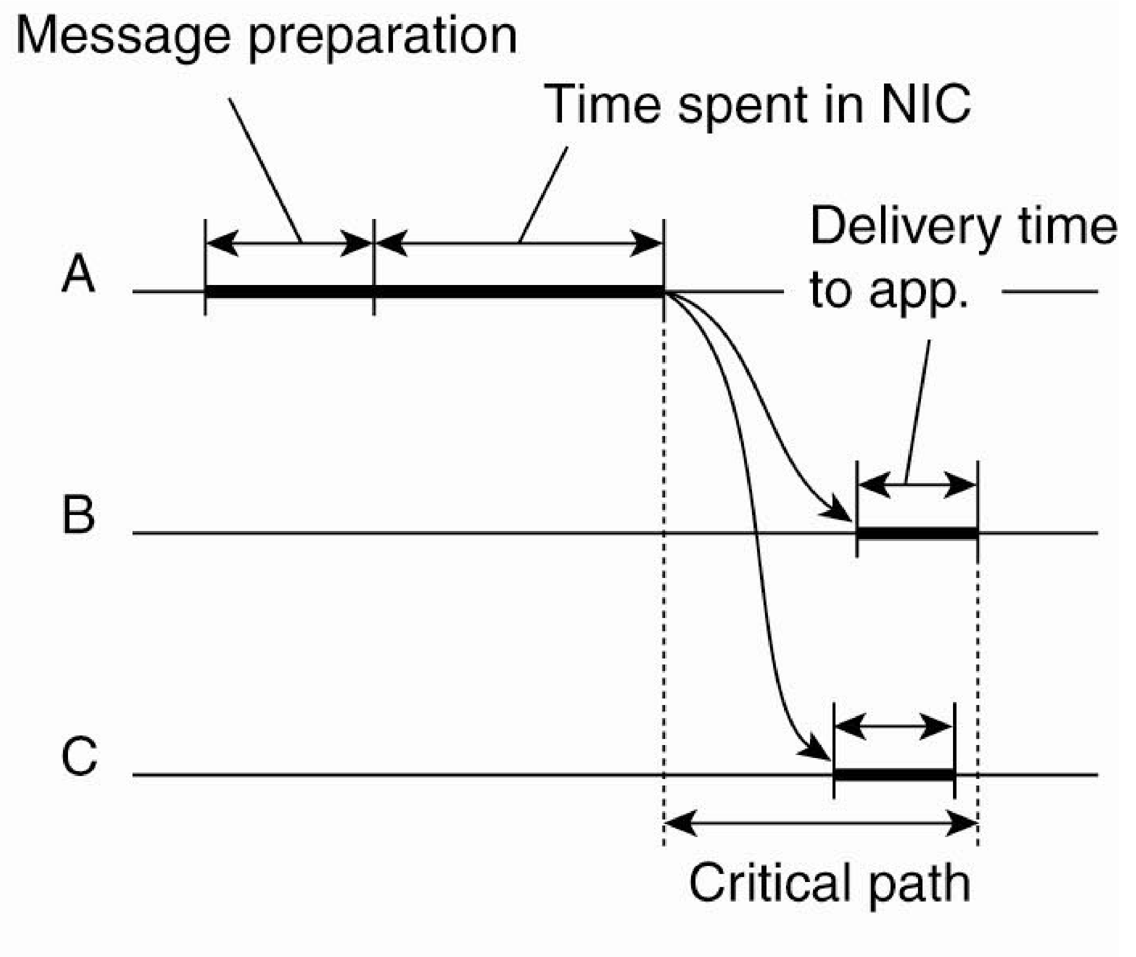
- Approach III:
  - Wireless sensor networks require tight time synchronization:
    - e.g., seismic activity monitoring
  - On the other hand, they are built of inexpensive hardware.
  - Special algorithms are necessary.
    - e.g., Reference Broadcast Synchronization (RBS)
- Idea: To eliminate various delays that introduce synchronization errors.

Time synchronization



(a)

Time synchronization



(b)

Time synchronization

- RBS:
  - A node broadcasts a reference message.

Time synchronization

- RBS:
  - A node broadcasts a reference message.
  - Each node, p, records the local time of reception, t^m_p.

Time synchronization

- RBS:
  - A node broadcasts a reference message.
  - Each node, p, records the local time of reception, t^m_p.
  - Nodes exchange their recorded reception times.

Time synchronization

- RBS:
  - A node broadcasts a reference message.
  - Each node, p, records the local time of reception, t^m_p.
  - Nodes exchange their recorded reception times.
  - Each node can compute its offset to another node.

Time synchronization

- RBS:
  - A node broadcasts a reference message.
  - Each node, p, records the local time of reception, t^m_p.
  - Nodes exchange their recorded reception times.
  - Each node can compute its offset to another node.

- Extremely tight synchronization: 1.85 ± 2.57 μs