Physical Computer Clocks

How computers measure time and synchronize their clocks

Lesson on Physical Computer Clocks

1. Introduction
2. Time in Distributed Systems
3. Physical Clock Synchronization
4. Synchronization Algorithms

Time in Distributed Systems

Computer Real Time Clocks

Physical Time in Distributed Systems (i)

- Einstein's Relativity Theory established that time contracts as the speed of an object increases

- Two observers in different moving states will have different opinions regarding the simultaneity of two events e and e' if there is not a causality relationship among them:
  - e -> e'
  - e' -> e
  - e = e'

- In Distributed Systems, the effects derived from Einstein's theory are negligible since speeds do not approach the speed of light and, normally the moving state of the different systems will be roughly the same
  - *In Distributed Systems, the universe is assumed to be Newtonian*

Physical Time in Distributed Systems (ii)

- **Events** of interest that occur within computer systems
  - Changes on data
  - Data transmissions

- **Timestamping.** Sometimes it's required that those events be consistently timestamped
  - Did event A precede event B or it was otherwise?
  - Consistency is easy if the two events occur within a single computer system that has a single Real Time Clock (RTC)
  - Challenge: If A and B occur in separate computer systems, can we state for sure that A preceeded B on the basis of their timestamps?
  - Can we claim that both RTC's were accurately synchronized so that we can be sure about the event's precedence?

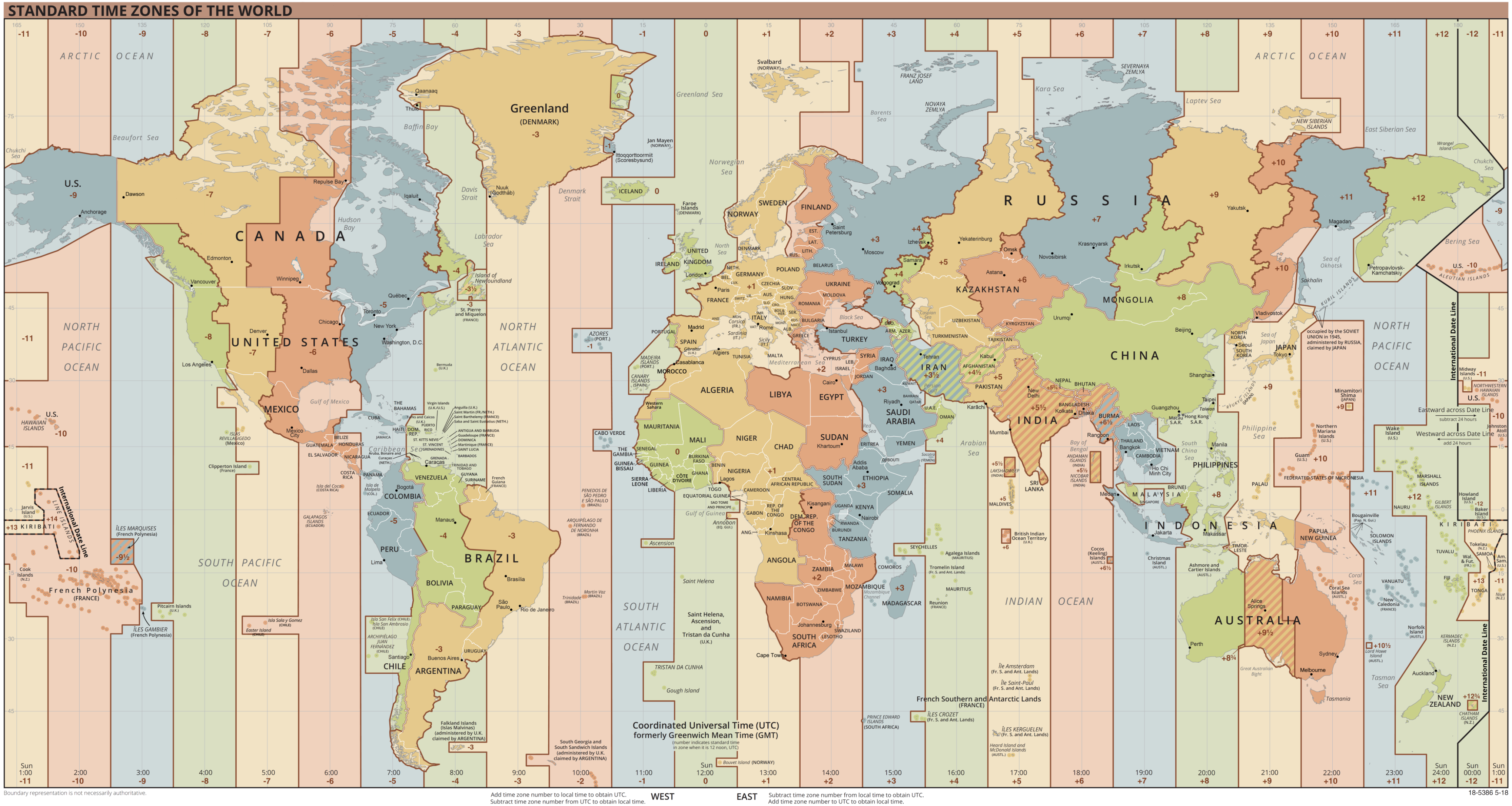
+ UTC: Universel Temps Coordonné

https://time.is/es/European%20Union

+ UTC: Universel Temps Coordonné

- Specified by ITU (Int. Telecommunications Union)
- ITU-R TF.460-6
- **Leap seconds** are irregularly added to compensate for the slowing of earth's rotation speed
- UTC is kept within 0.9 sec of the International Atomic time UT1

UTC: Universel Temps Coordonné



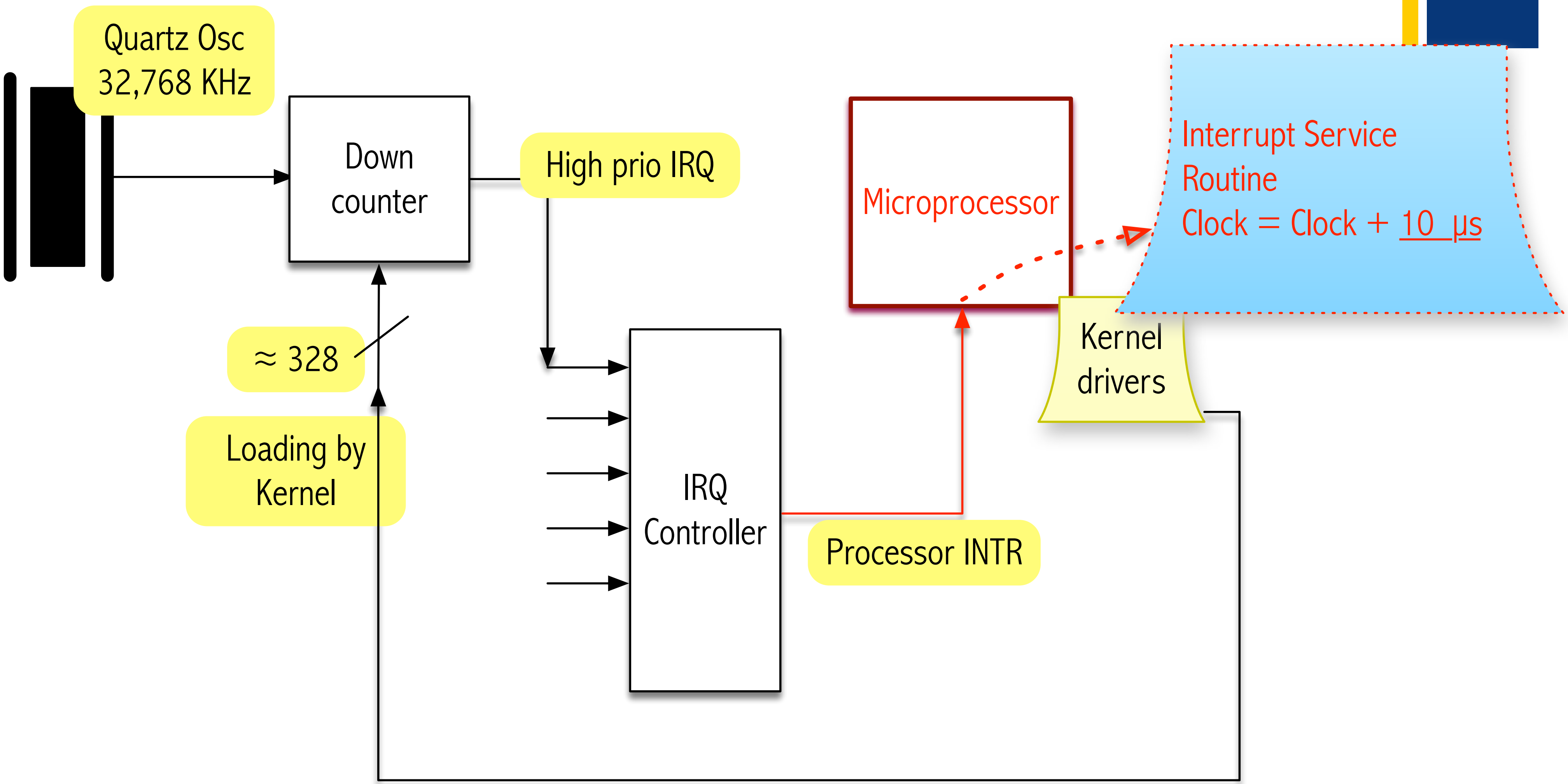
https://upload.wikimedia.org/wikipedia/commons/8/88/World_Time_Zones_Map.png

+ Clocks

- **Atomic clock**
  - The transition between two hyperfine levels of Cesium-133 is used to define the second (9.192.631.770 periods of radiation)
  - Accuracy ≈ 1 second in 6 million years
  - **UTC time (Universel Temps Cordonné) is kept in an atomic clock**
- How to obtain accurate time:
  - **GPS** (±1 msec of UTC)
  - Short wave radio: WWV broadcasts UTC from Fort Collins, Co., USA
- Computer Systems keep time in RTCs (**Real Time Clocks**) even when the computer system is idle or turned off

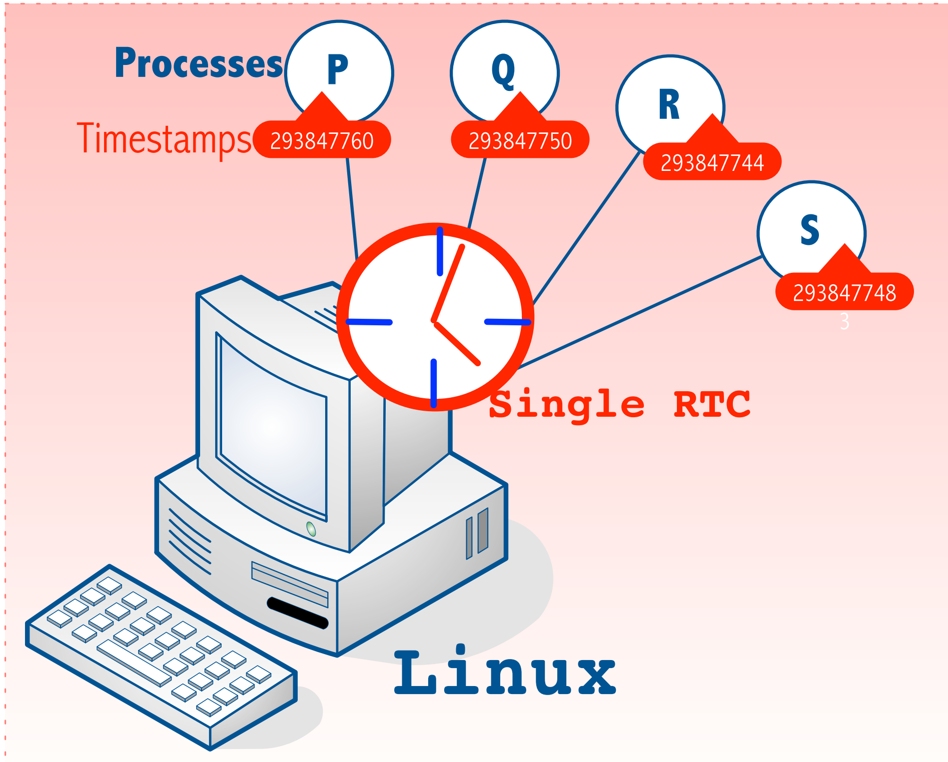
What's a computer RTC (Real Time Clock)?

Computers keep time by using Real Time Clocks



Timestamping consistency

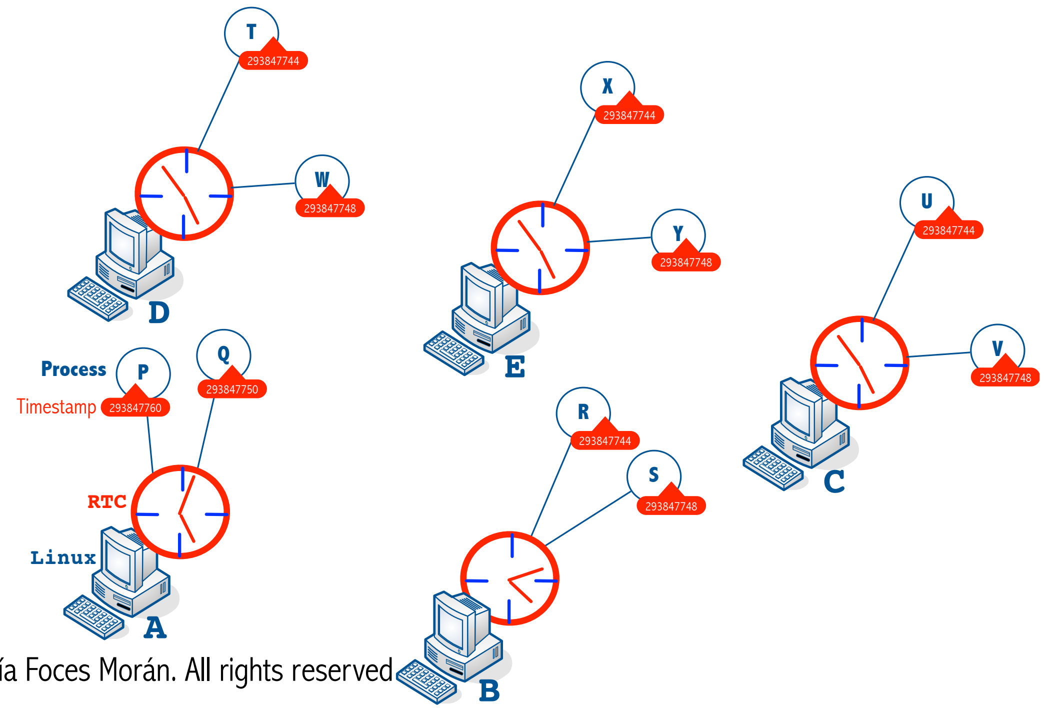
- Consistency is guaranteed if the two events occur within a single computer system that has a single Real Time Clock (RTC)



Timestamping in DS

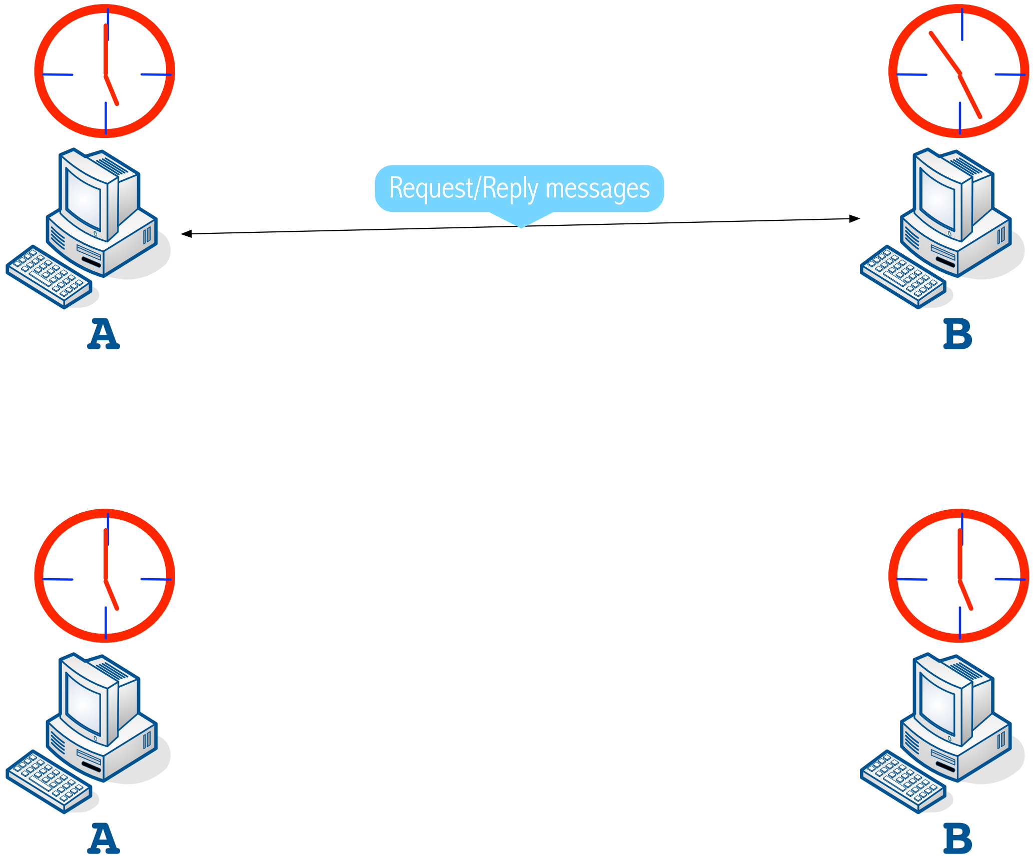
Consistency entails clock synchronization with quality

- Challenge: If A and B occur in separate computer systems, can we claim event A preceeded event B on the basis of their timestamps?
  - **Only** if RTCs were accurately **synchronized**



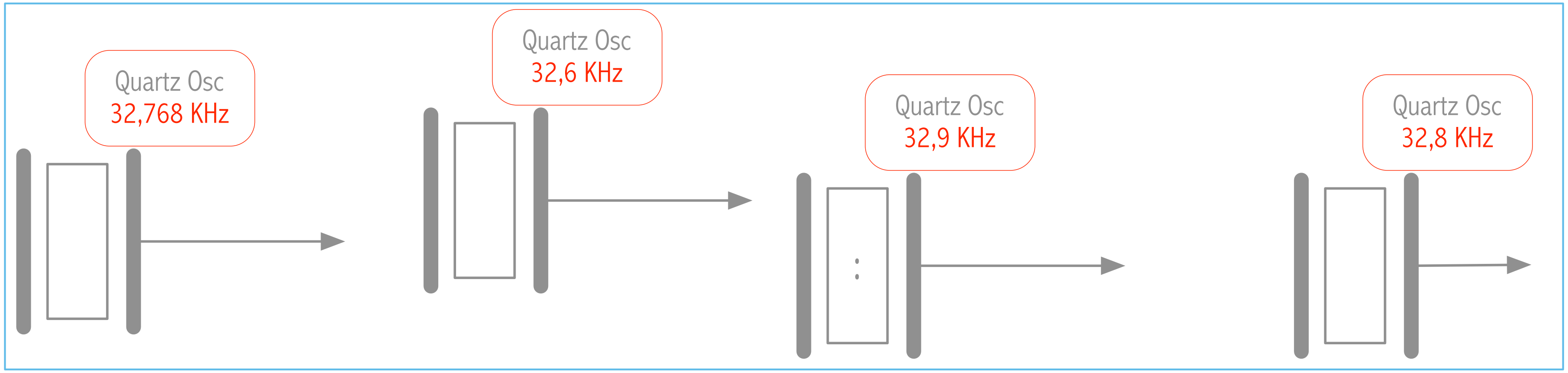
What is clock synchronization

- Setting two clocks to exactly the same time of day
- A and B exchange time-related messages
- Synchronization is attained by iteratively exchanging time-related messages



Why do clocks get out of sync?

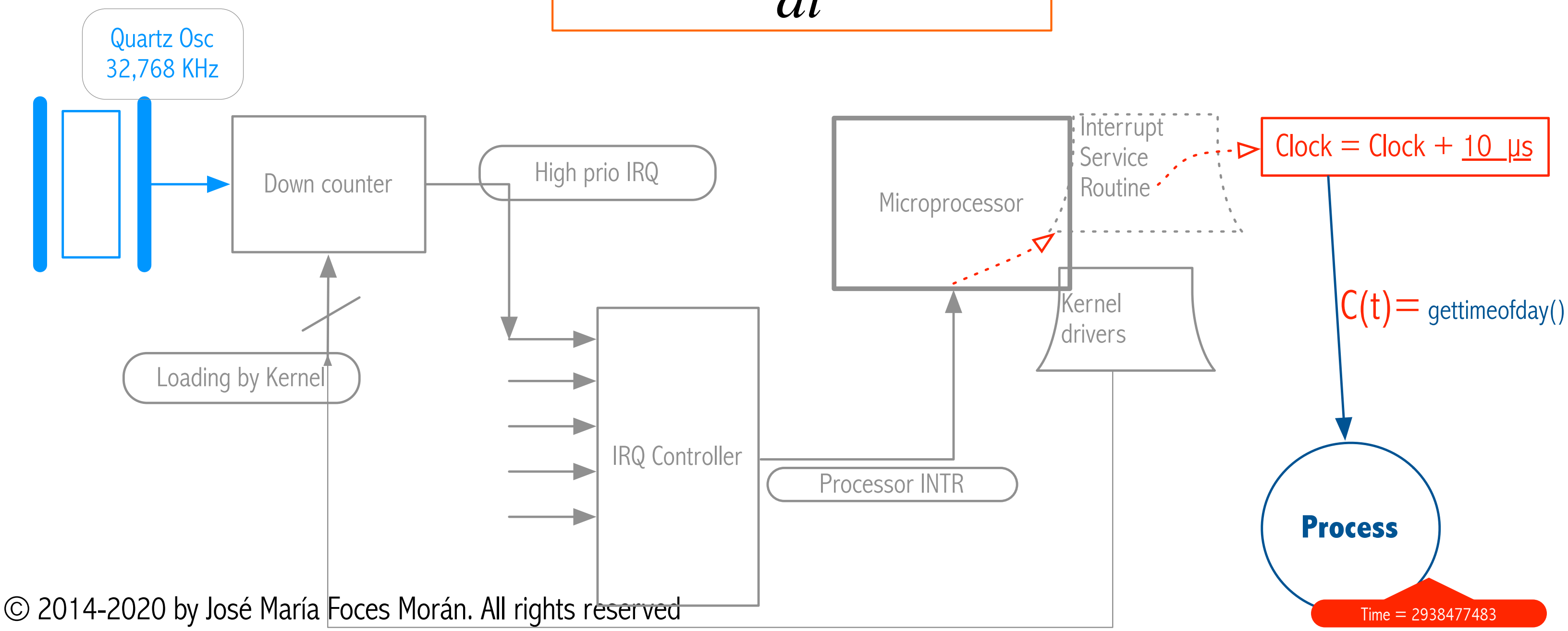
- Quartz crystals oscillate at silightly different frequencies
  - Manufacturing process
  - Temperature
  - This causes clock **drift**: The undesired change in clock speed



Clock drift

- Quartz crystals oscillate at slightly different frequencies
  - Assume ρ represents the **maximum absolute drift rate:** Rate of change of clock speed
  - C(t) is the clock speed
  - C'(t) = dC(t)/dt is the instantaneous clock speed *change* per second (An acceleration)

$$1-\rho \le \frac{dC(t)}{dt} \le 1+\rho$$



Clock drift

- Quartz crystals oscillate at slightly different frequencies
  - Assume ρ represents the maximum drift rate
  - t is the UTC physical time

- Fast clock: $$\frac{dC(t)}{dt} > 1$$

- Slow clock: $$\frac{dC(t)}{dt} < 1$$

- Perfect clock: $$\frac{dC(t)}{dt} = 1$$

- Clock working within **acceptable drift limits** if there exists a constant ρ such that:

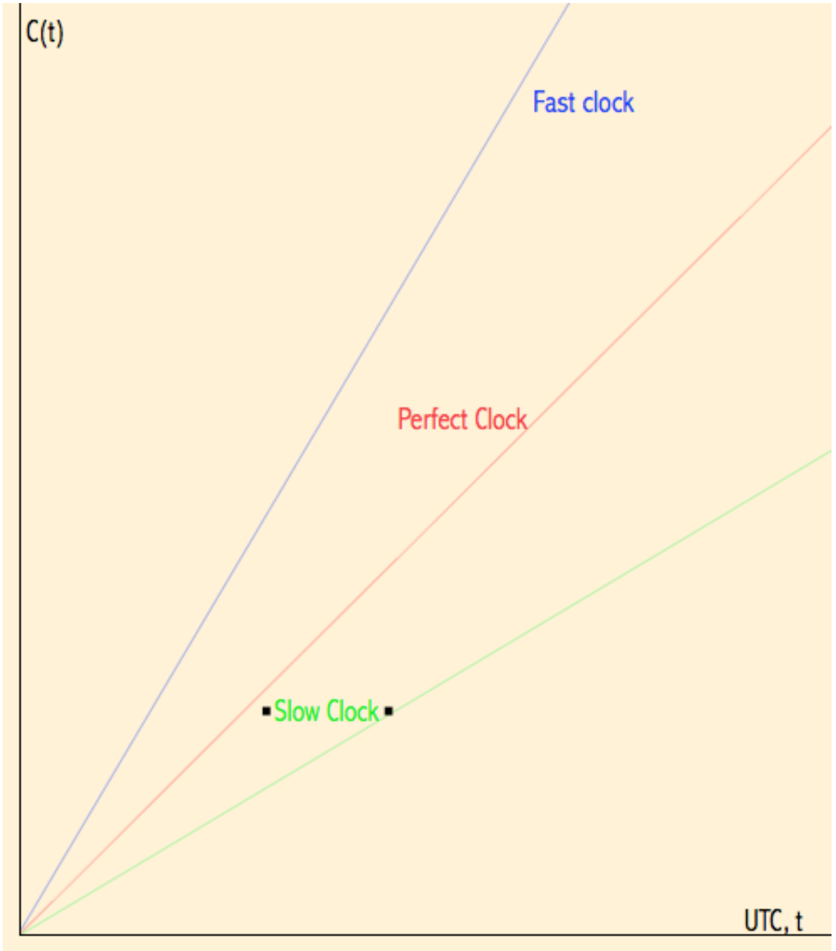
$$1 - \rho \leq \frac{dC(t)}{dt} \leq 1 + \rho$$

- UNIX system call gettimeofday() returns the time

$$C(t) = gettimeofday();$$

Clock drift

- Quartz crystals oscillate at slightly different frequencies
  - Assume ρ represents the maximum drift rate
  - t is the UTC physical time

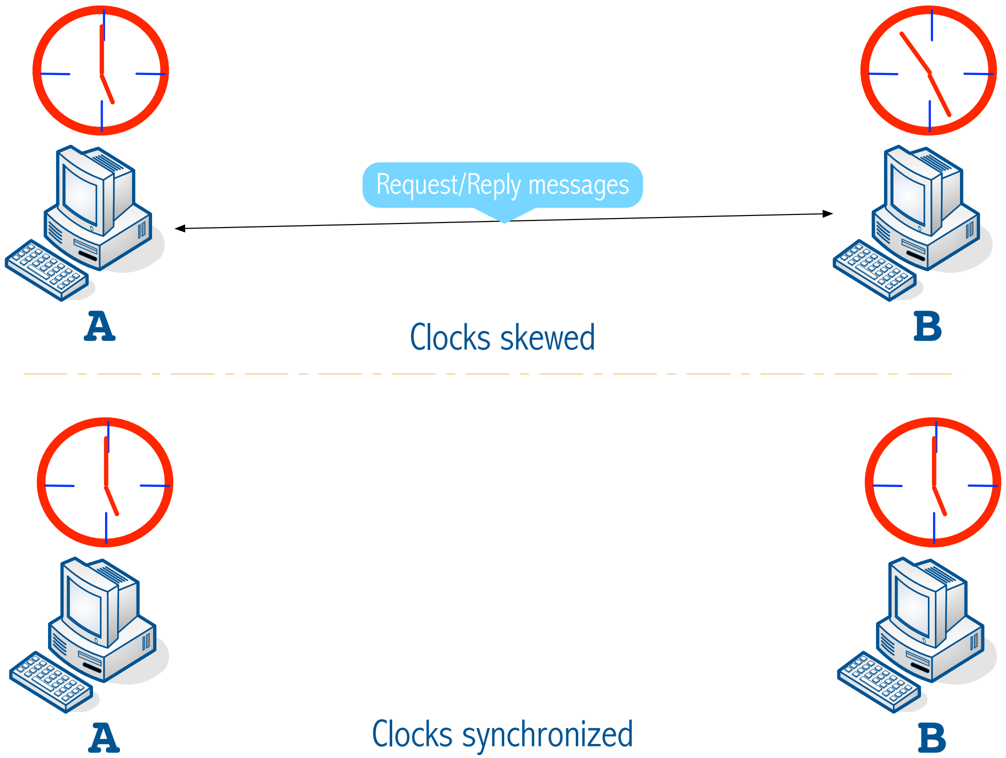


From "Distributed Systems", Andrew Tanenbaum, Ch. 5
© Prentice-Hall 2005

- Fast clock: $$\frac{dC(t)}{dt} > 1$$
- Slow clock: $$\frac{dC(t)}{dt} < 1$$
- Perfect clock: $$\frac{dC(t)}{dt} = 1$$

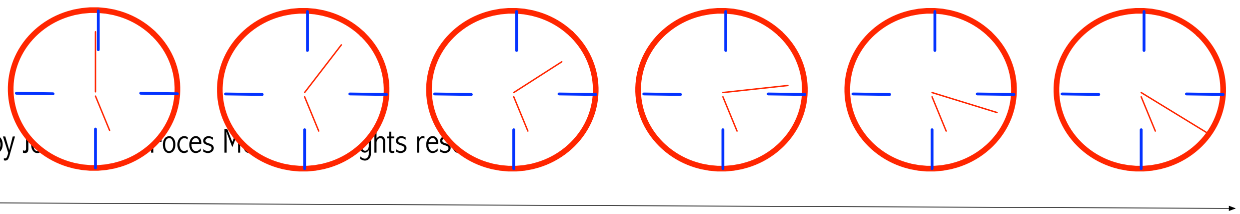
Clock skew

- Over time, **drift causes skew**
- Clocks must be brought back into sync, **how?**

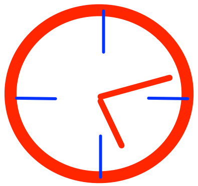


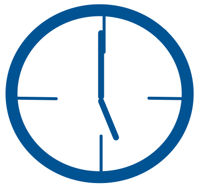
Computer clocks and monotonicity

- Computer clocks must always count up:
  - 10 11 12 13 µs . . .
- Clocks must never be run backward
- If a clock time change is necessary, it must be applied gradually
- Otherwise, undesirable effects will be caused, for example:
  - A background task is daily executed at 3 am
  - At 5 am we set the clock back to 3 am, then the task results unexpectedly repeated which might have negative conseuqences

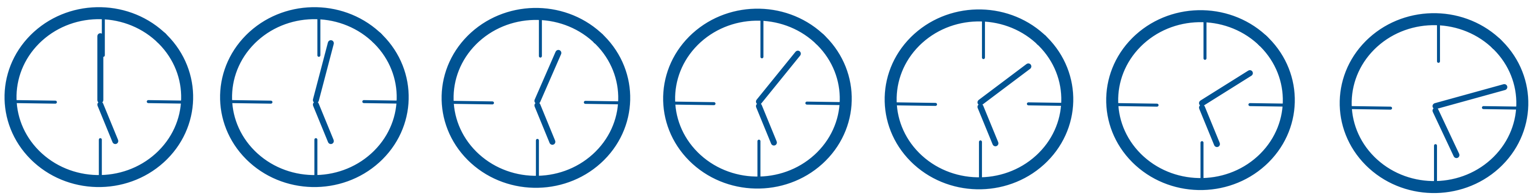


Monotonous clock synchronization



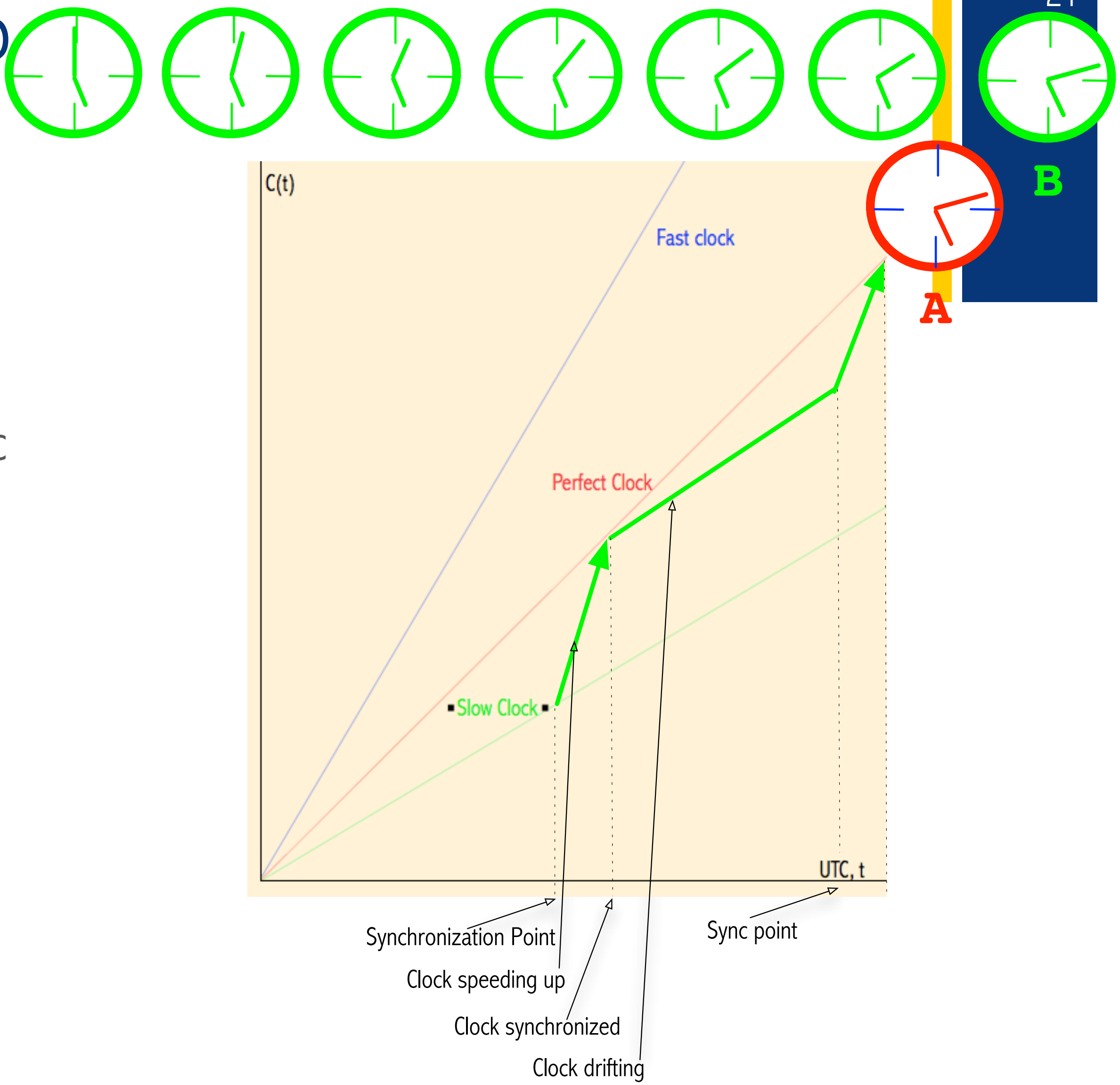


A **B** is slow vs. A



B's speed is increased until it reaches the target. The change is gradual.

+ Gradual clock speedup to achieve synchronization

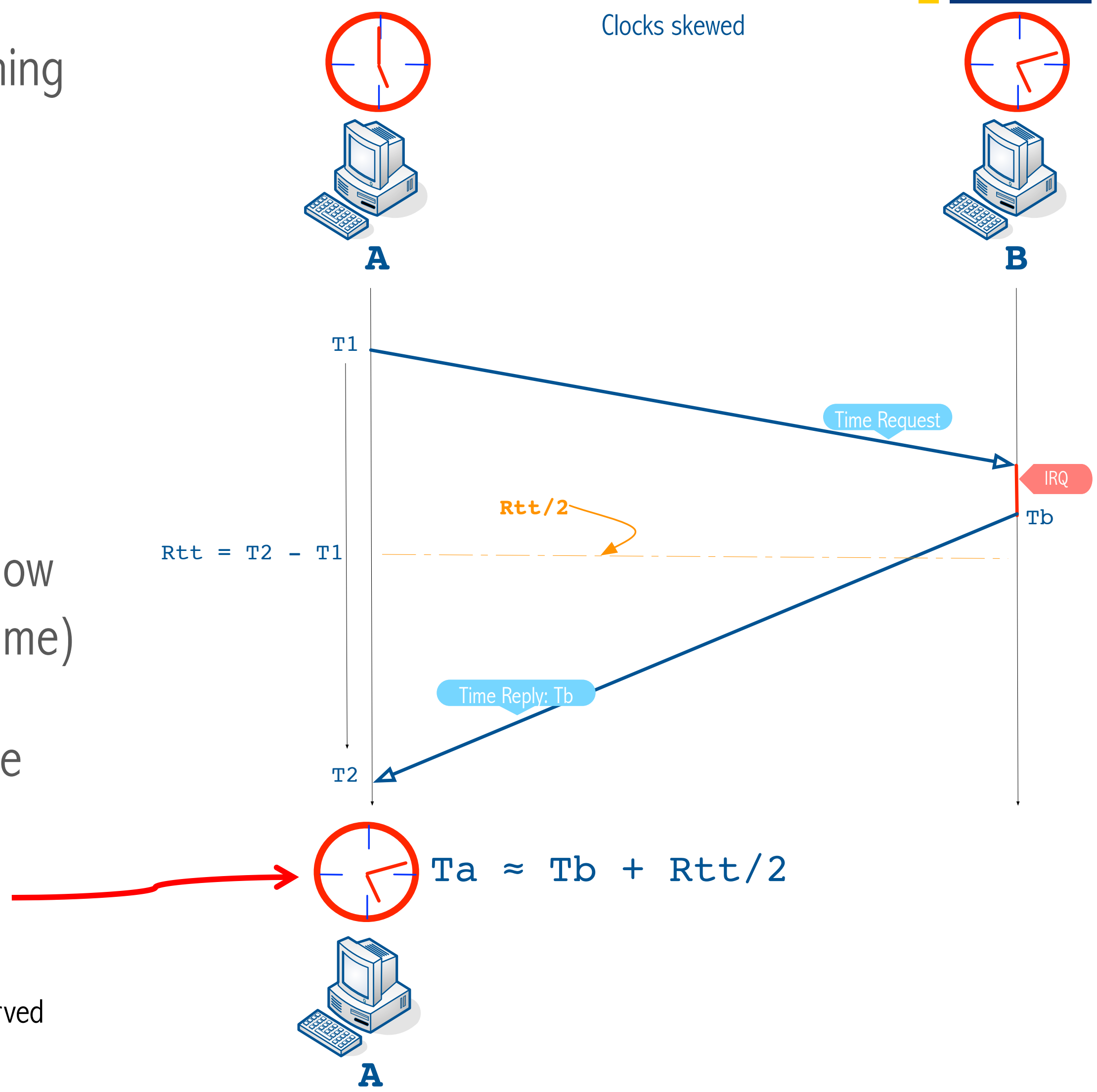


- Clocks must be brought back in sync
- How?
- How often?
- Always gradually?
  - Sometimes, jumping a clock forward is acceptable
  - Daylight savings time?

If the clock has a max drift rate of ρ, and synchronization must occur in ∂ seconds max, the synchronization points must be at a max distance of ∂/2ρ

Naïve clock synchronization

- A and B exchange messages containing timestamps
- The one way delays A-B and B-A, in general, will be different
- **Host A measures Rtt**
- A priori, it assumes that IRQ time is low (Actually, the timeserver Response time)
- Assumes that delays A-B and B-A are equal, Ta is set in a gradual fashion:

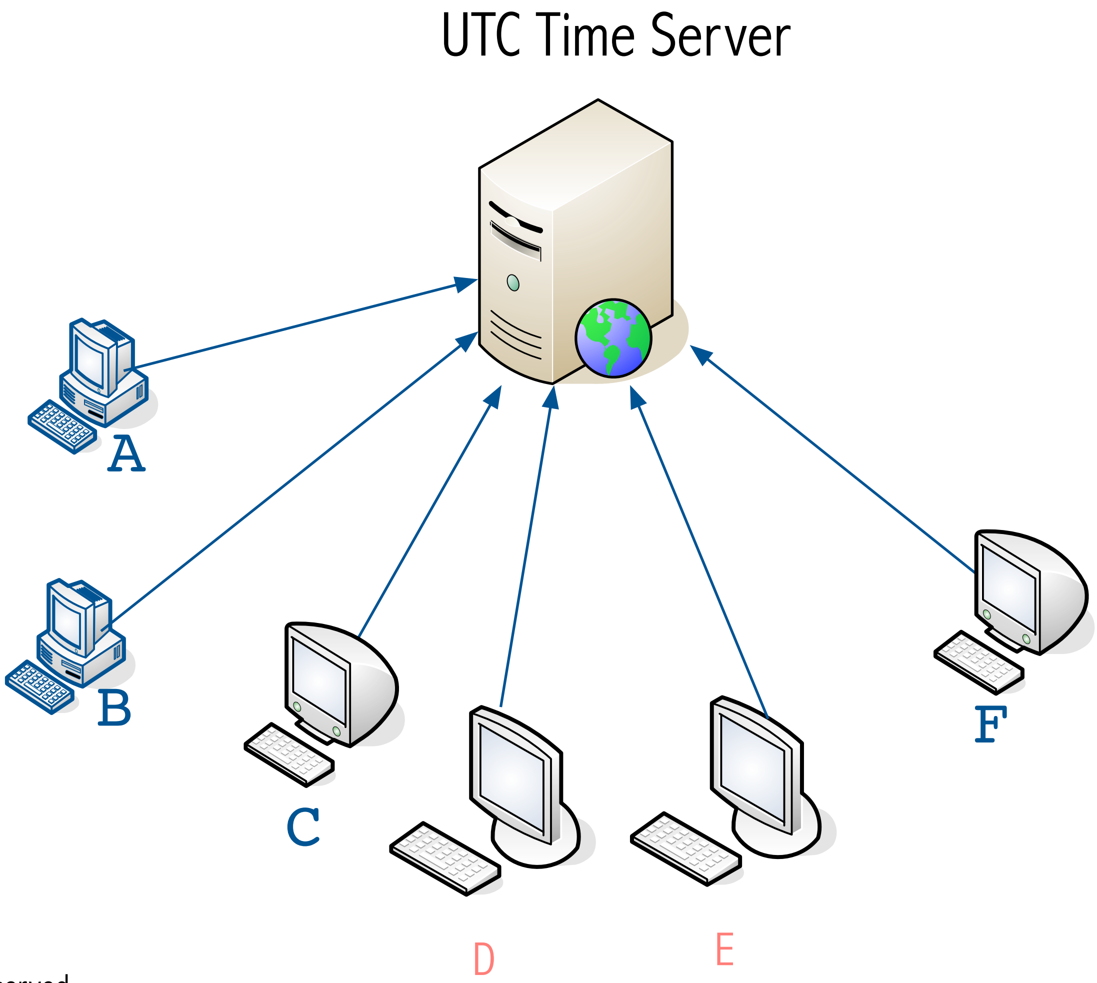


Algorithms for clock synchronization

Can clock sync scale with the Internet?

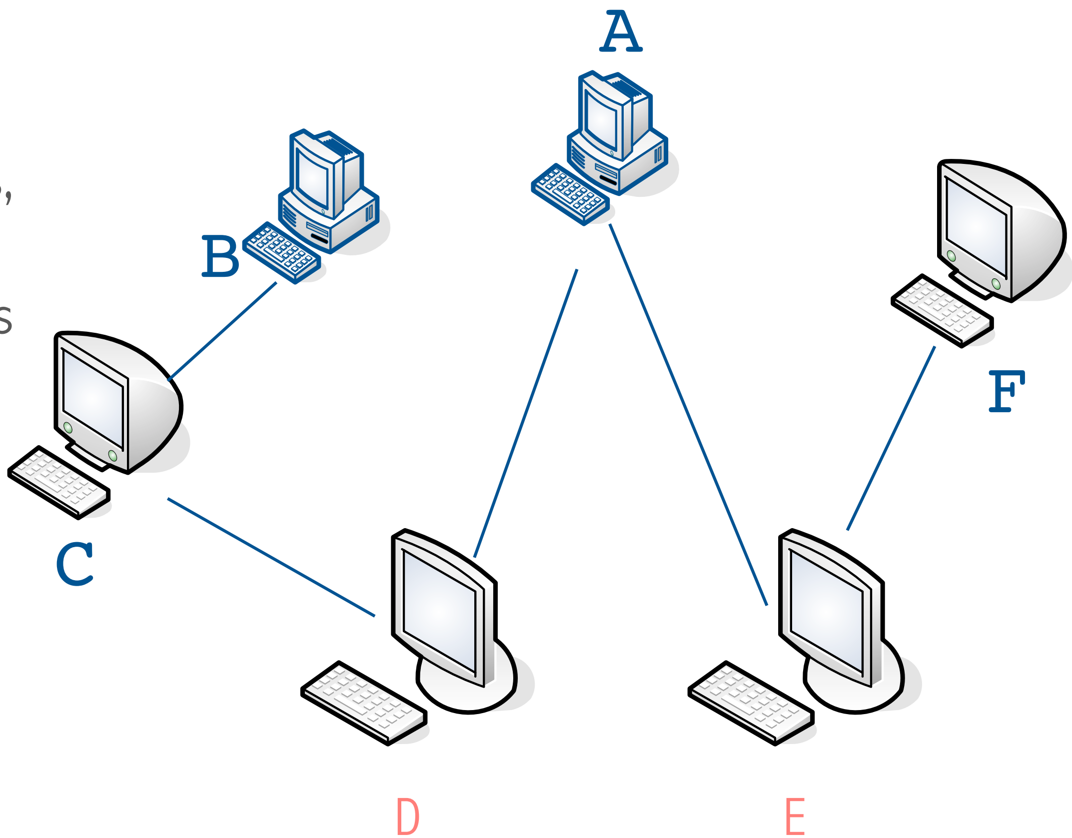
External synchronization

- Each host synchronizes individually with a Time Server
- Correct for **timestamping events** in a distributed system
- If the synchronization bound is D then the group is <u>internally synchronized with tolerance 2D</u>



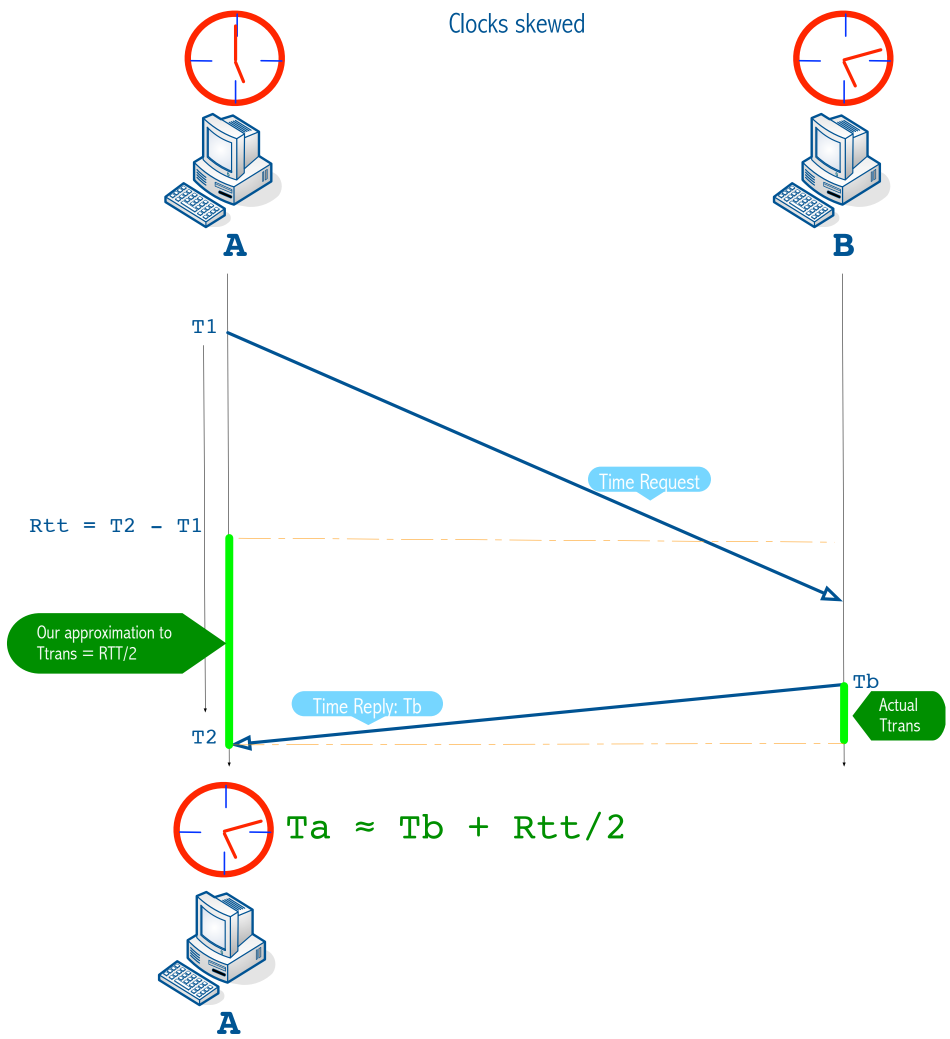
Internal synchronization

- A group of hosts synchronizes with other hosts from the group only
- Correct **for measuring time intervals**, but *not for timestamping* if considering events beyond the hosts of the group

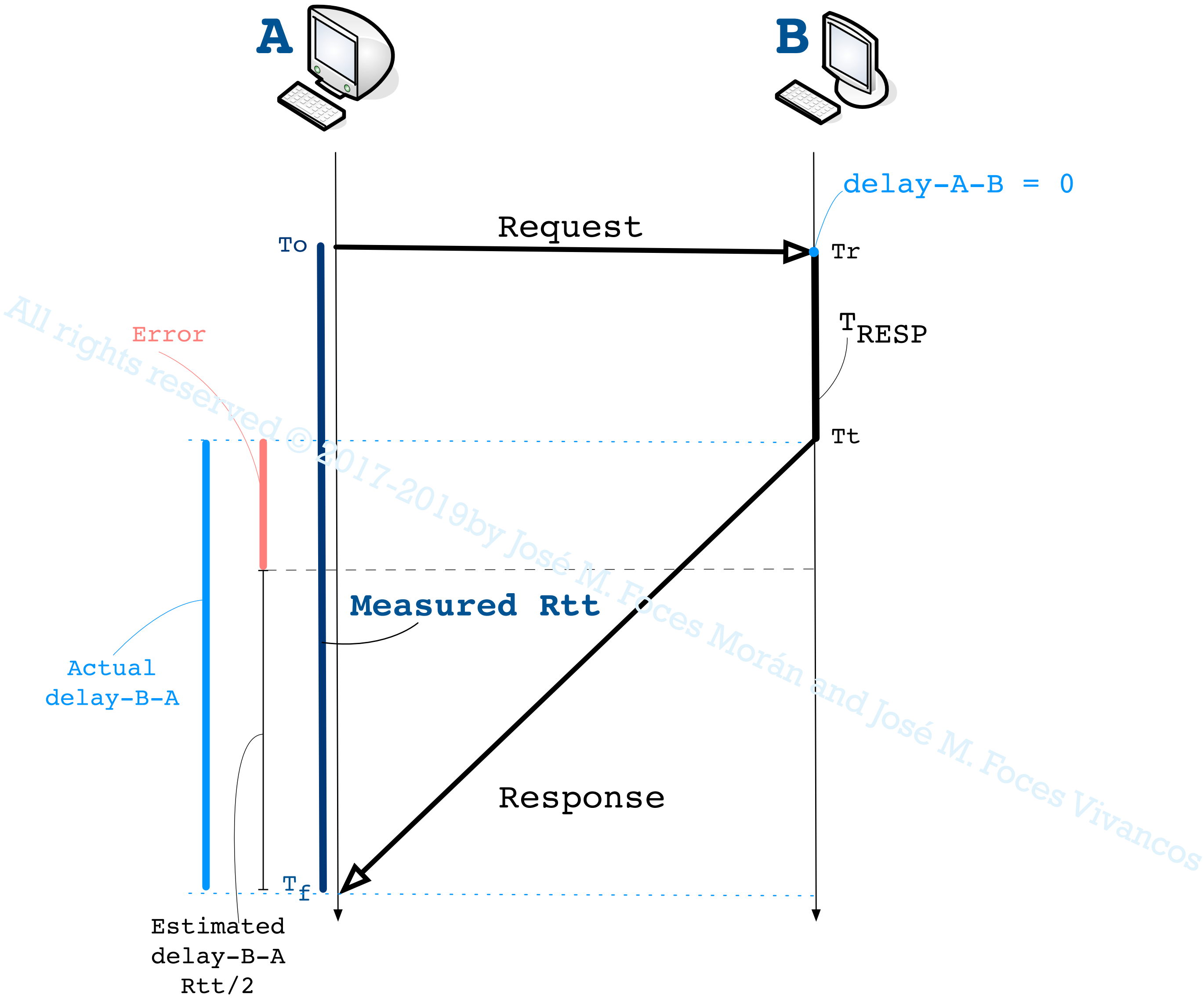


The accuracy of clock synchronization

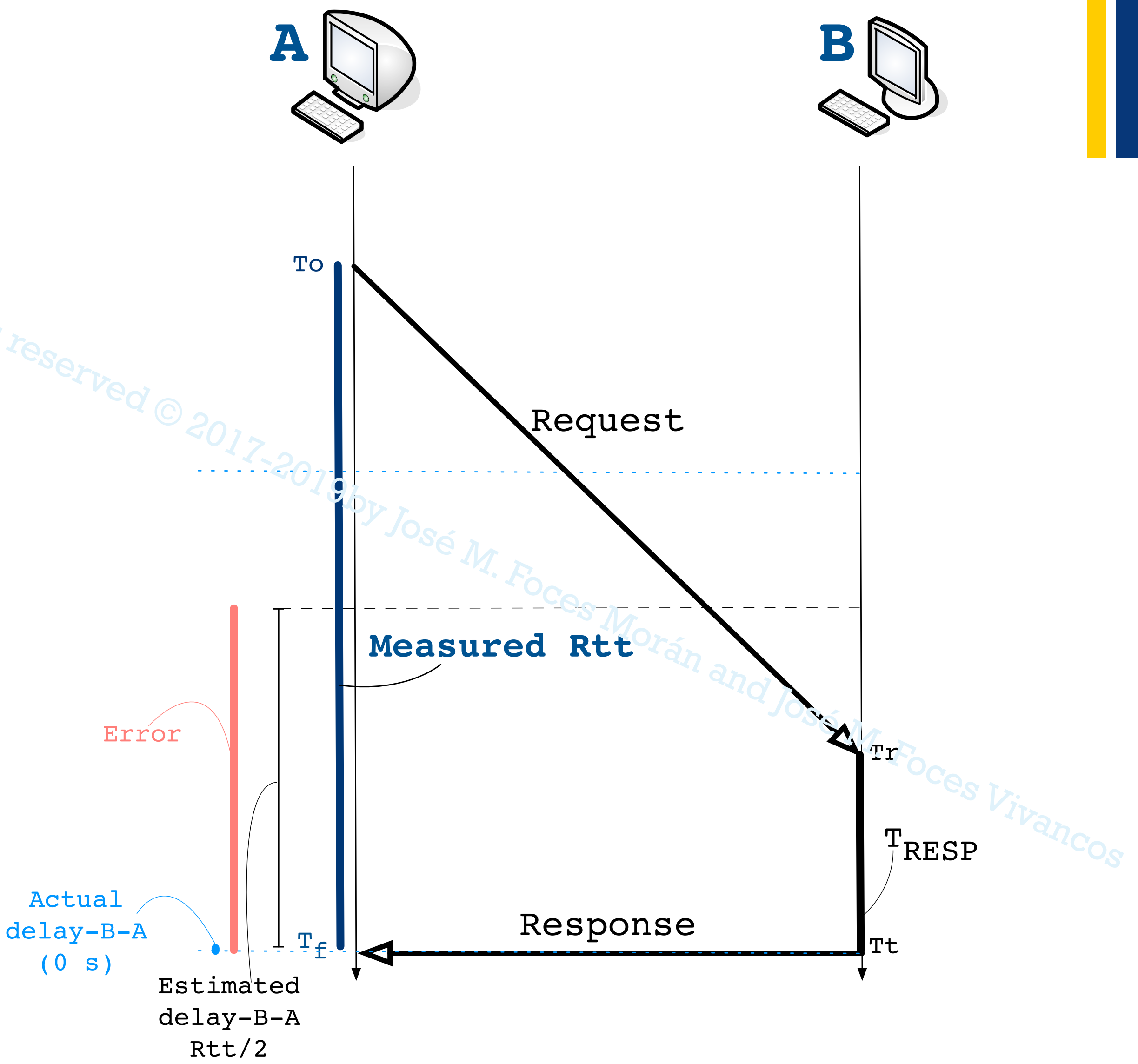
- When the network is undergoing congestion our assumption that T_{transB} is roughly equal to Rtt/2 is likely to be wrong
  - This will cause clock A time to have an offset vs. that of B
- Can we estimate the accuracy of this synchronization mechanism?
- Let's assume a minimum for Ttrans in our network: min



The accuracy of clock synchronization

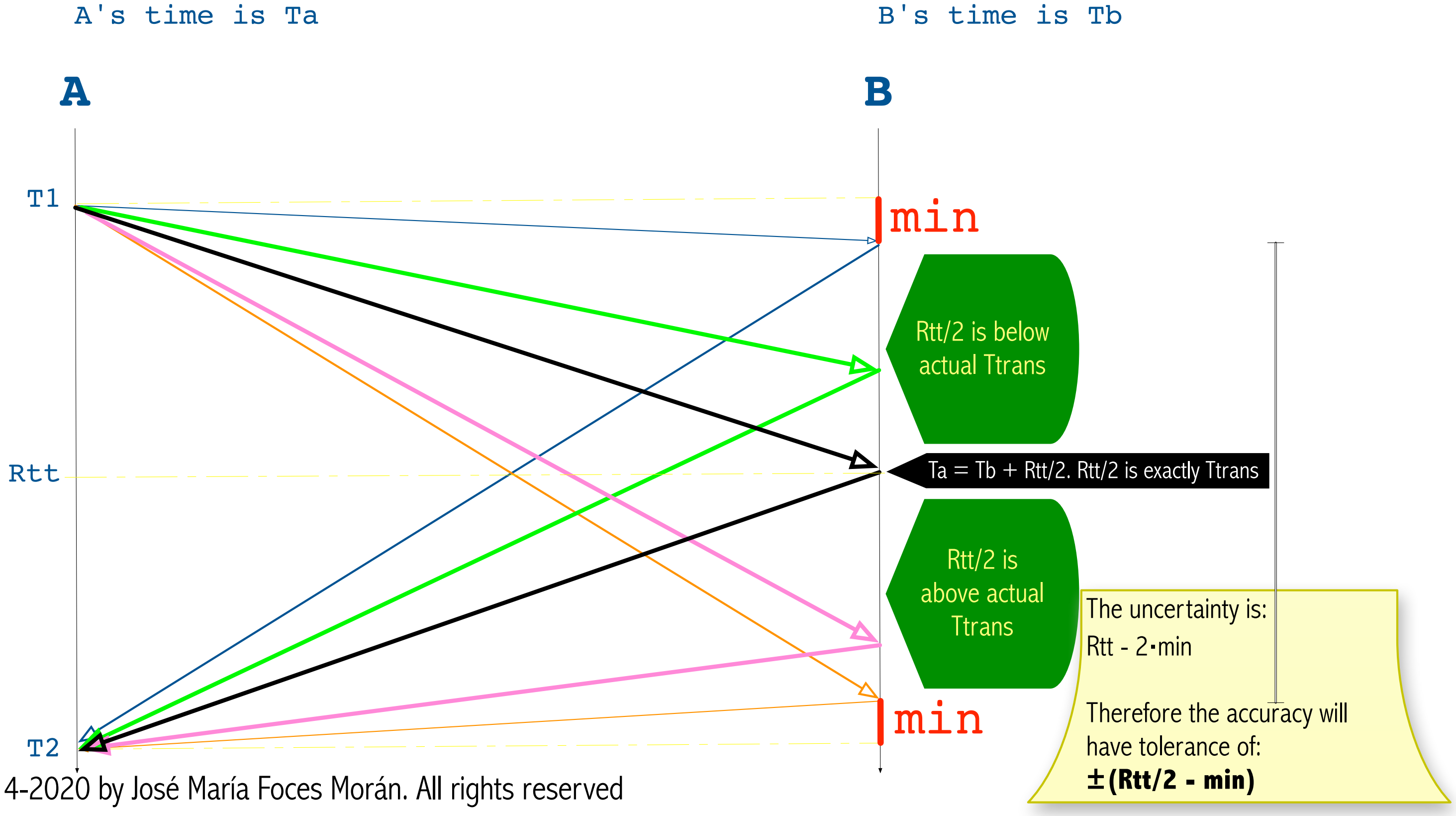


The accuracy of clock synchronization



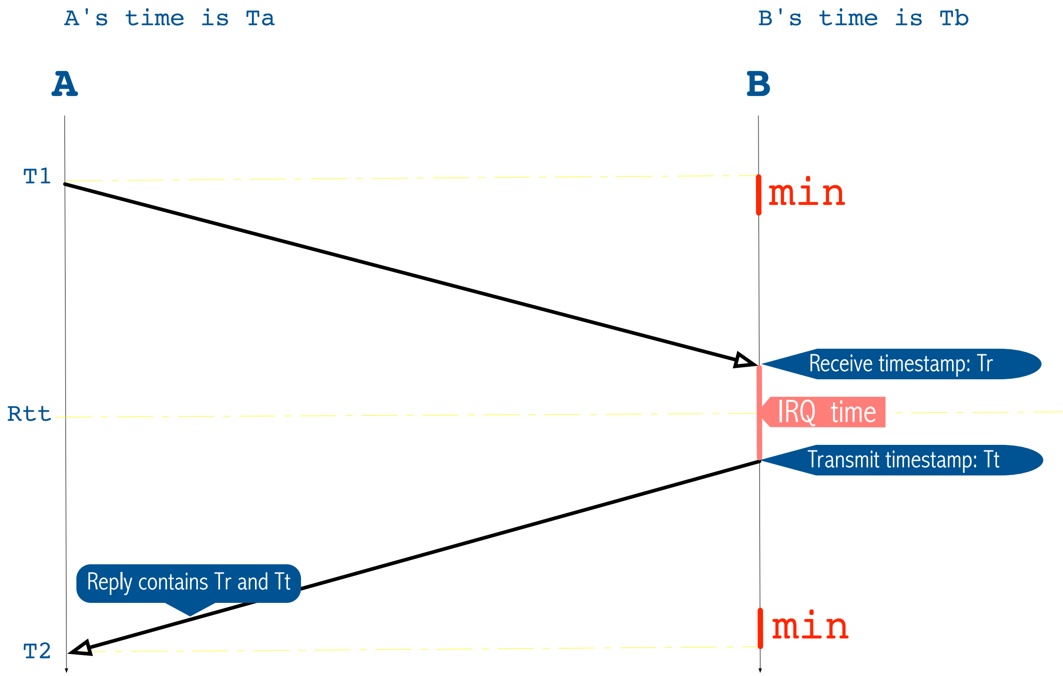
Degree of accuracy of clock synchronization

- Let's assume we know minimum for Ttrans in our network: min
- The attained accuracy depends on Rtt: a high Rtt will cause a low accuracy of estimate of Ttrans as Rtt/2



How long is the IRQ/ISR time?

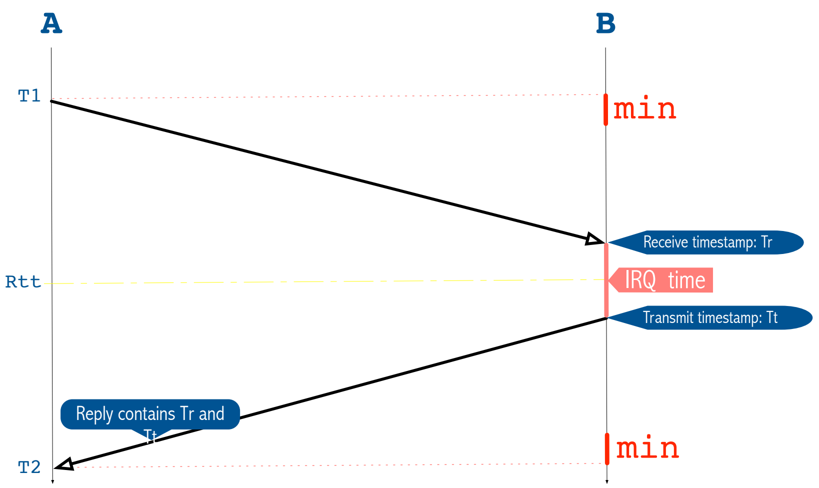
- The time server can transmit the time reply alongside the time at which the request ws received
- This will enable the client to calculate the IRQ time thereby contributing to improving the accuracy of Rtt/2



What's the precision if Irq time is significant?

- The fact that we know how much time it takes for a system to process the time request, which most significant component is Irq time, the time it takes to process an interrupt, will allow us to reduce the uncertainty when estimating Rtt/2 in the receiver

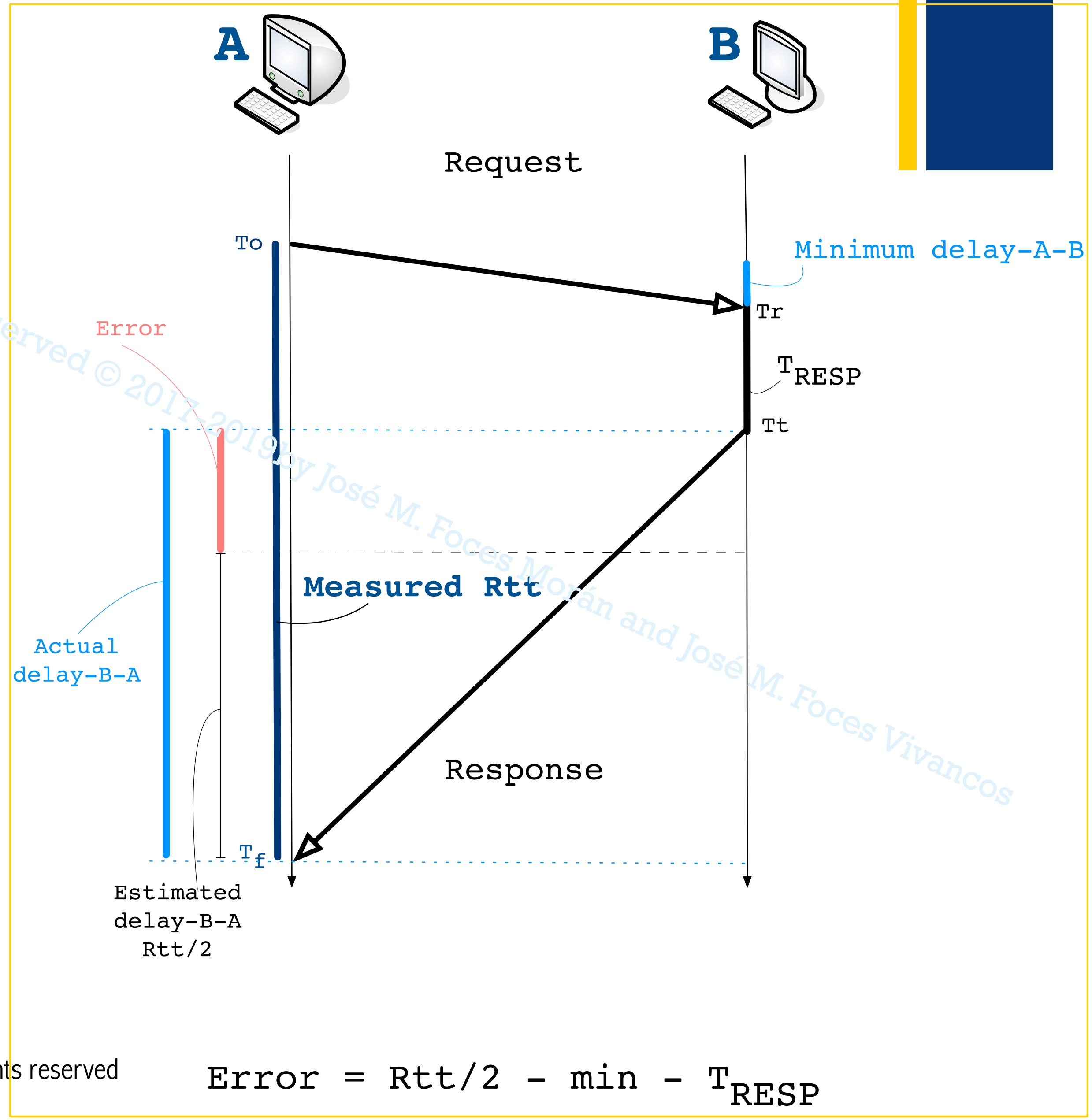
- Since Irq time is known, the uncertainty will be = Rtt – Irq – 2 · min



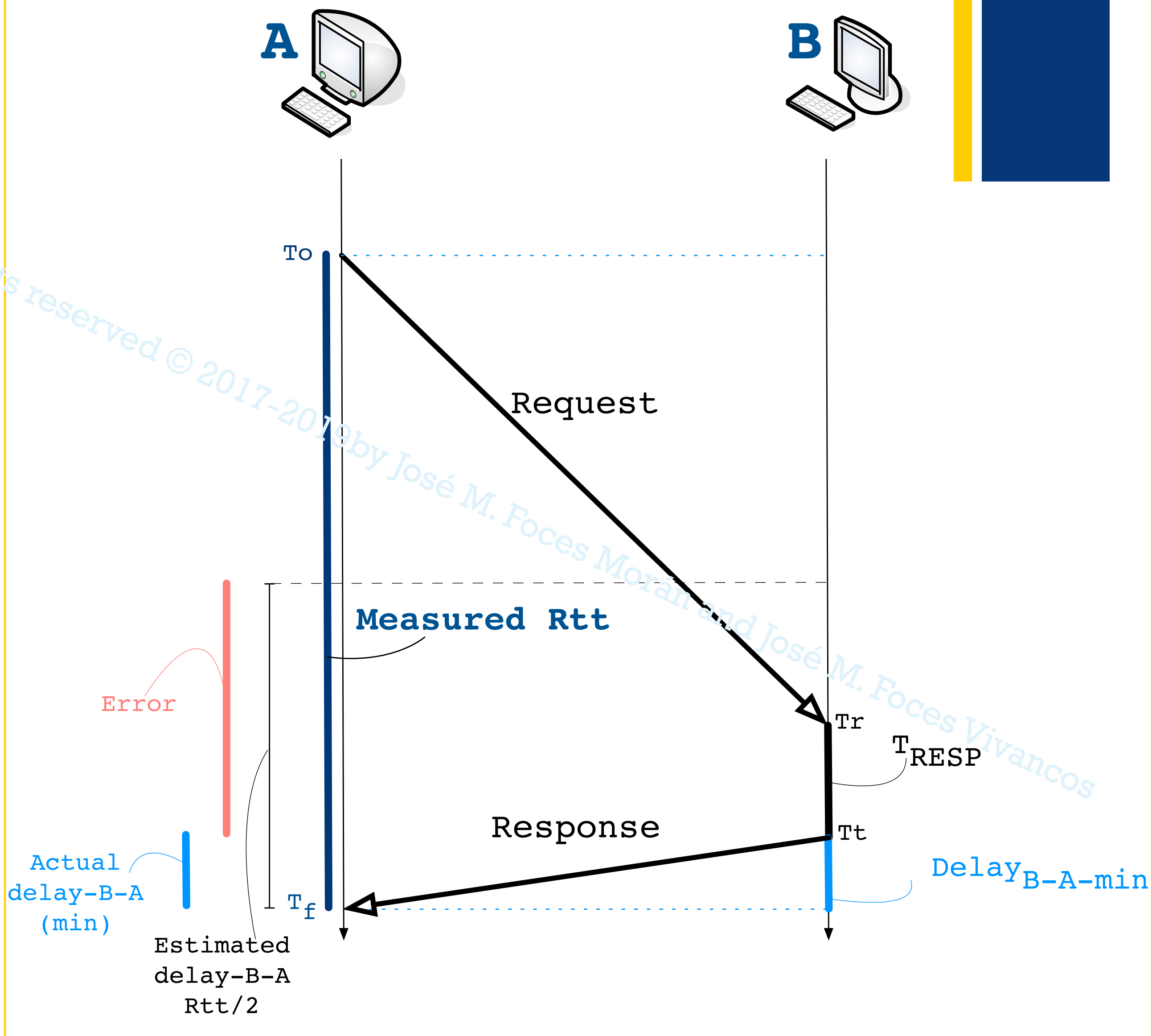
We know the IRQ time term, therefore the uncertainty in this case is:
Rtt - Irq - 2·min, let Rtt' = Rtt - Irq, then the uncertainty is = Rtt' - 2·min

Therefore the accuracy will have tolerance of:
± (Rtt-Irq)/2 - min =± Rtt'/2 - min

Error when DelayA-B is min



Error when DelayB-A is min

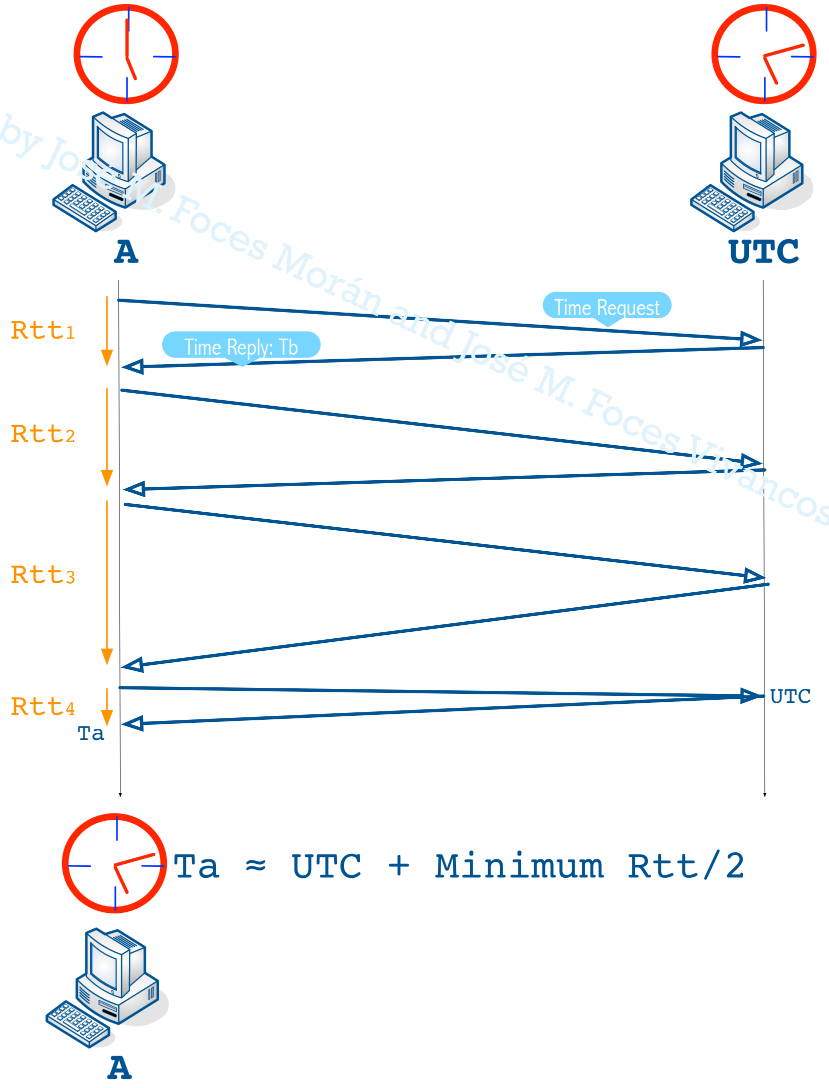


Error = Rtt/2 - min

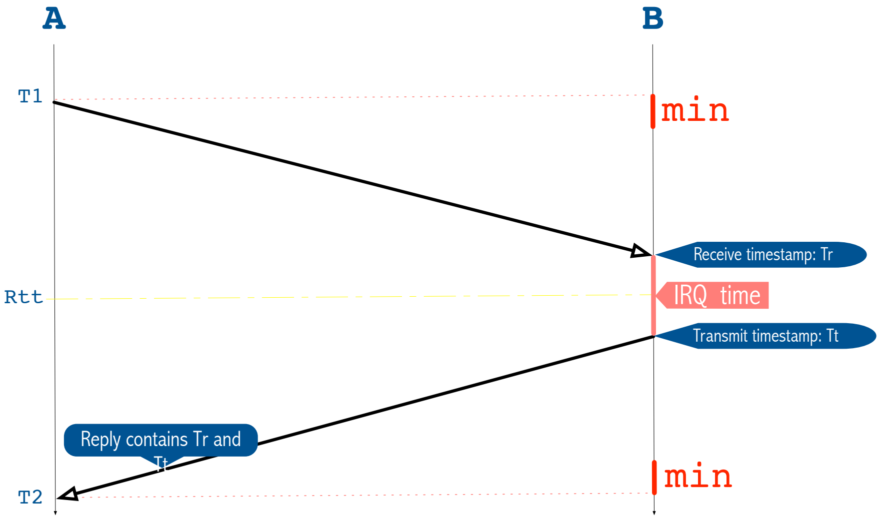
Improving clock synchronization

Cristian's Algorithm

- How can we improve the $T_{trans-B}$ estimate?
- By measuring Rtt ($T_2 - T_1$) several times and discarding the maximum values of Rtt obtained greater than a threshold
  - Due to network congestion
- We accept the minimum sampled Rtt assuming it was due to a lack of net congestion
  - It serves us for setting A's clock and for calculating the precision of the attained clock synchronization



+ Calculating the time synchro precision: Example



- **Calculate error** in the two following cases. Assume min = 10ms:

1. DelayB–A is min
2. DelayA–B is min

(See preceding slides)

	Timestamps at host A		Timestamps at host B					
Request	Originate	TimeOfDay A	Receive TS	Transmit TS	Rtt (ms)	Residence time (ms)	RttN (ms)	
1	1:10:32.133	1:10:32.163	1:00:32.150	1:00:32.156	30	6	24	
2	1:10:34.025	1:10:34.056	1:00:34.040	1:00:34.045	31	5	26	
3	1:10:37.494	1:10:37.524	1:00:37.510	1:00:37.514	30	4	26	
4	1:10:40.305	1:10:40.334	1:00:40.320	1:00:40.326	29	6	23	
5	1:10:44.734	1:10:44.766	1:00:44.750	1:00:44.752	32	2	30	
6	1:10:49.005	1:10:49.039	1:00:49.020	1:00:49.027	34	7	27	
7	1:10:55.254	1:10:55.282	1:00:55.270	1:00:55.276	28	6	**22**	0,022 s
8	1:10:58.635	1:10:58.665	1:00:58.650	1:00:58.657	30	7	23	
9	1:11:02.325	1:11:02.357	1:01:02.340	1:01:02.344	32	4	28	
10	1:11:08.885	1:11:08.921	1:01:08.900	1:01:08.906	36	6	30	
			RttN/2 =	0,011 s				
			Target time for A in sync with B =	Transmit TS + RttN/2 =		1:00:55.276 + 0,011 =	1:00:55.287	
			Delta for adjtime() =	Target - TimeOfDay A =		1:00:55.287 - 1:10:55.282 =	-599,995	s
			Delta for adjtime() =	-599,995	s			

Improving clock synchronization

Berkeley Algorithm

- Time server is active in this algorithm
  - It polls every host from time to time to ask what is the time there

Improving clock synchronization

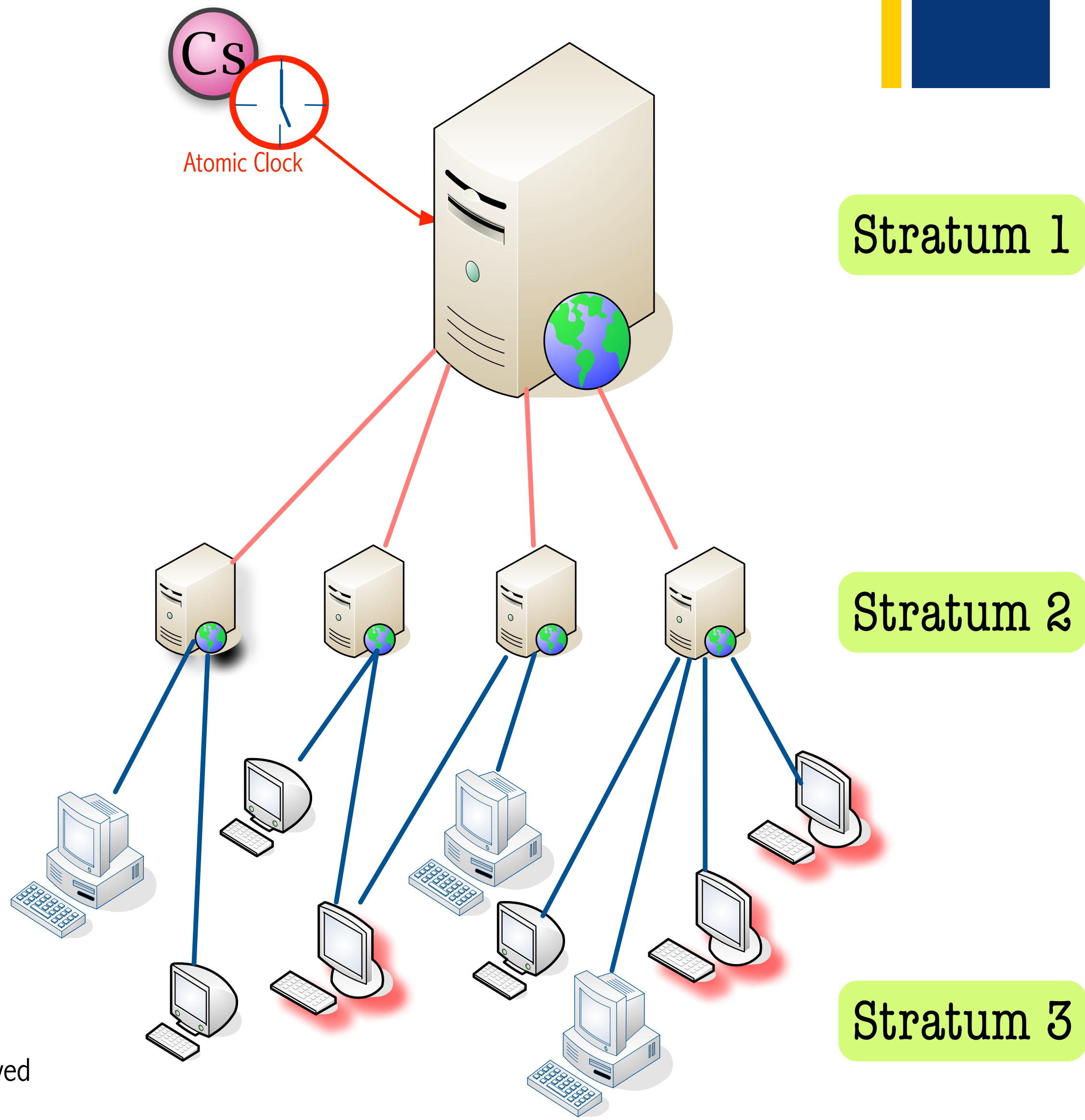
NTP: Network Time Protocol

- Client hosts attain synchronization with UTC time servers
- Employs statistical data filtering for establishing the quality degree of the timing data (RTT)
- Services situations of lack of connectivity for lengthy times
- Designed for a large scale
- Requires authentication
- Based on UDP
- RFC 5905

Improving clock synchronization

NTP: Network Time Protocol

- It's based on a hierarchy of servers made up of strata
  - Primary servers are connected to an Atomic-clock UTC base (Stratum 1)
  - Secondary servers synchronize with primary servers (Stratum 2)
  - Hosts reside on stratum 3



+ Exercises

EXERCISES

14.1 Why is computer clock synchronization necessary? Describe the design requirements for a system to synchronize the clocks in a distributed system. *page 596*

14.2 A clock is reading 10:27:54.0 (hr:min:sec) when it is discovered to be 4 seconds fast. Explain why it is undesirable to set it back to the right time at that point and show (numerically) how it should be adjusted so as to be correct after 8 seconds have elapsed. *page 600*

14.3 A scheme for implementing *at-most-once* reliable message delivery uses synchronized clocks to reject duplicate messages. Processes place their local clock value (a 'timestamp') in the messages they send. Each receiver keeps a table giving, for each sending process, the largest message timestamp it has seen. Assume that clocks are synchronized to within 100 ms, and that messages can arrive at most 50 ms after transmission.

i) When may a process ignore a message bearing a timestamp T, if it has recorded the last message received from that process as having timestamp T' ?

ii) When may a receiver remove a timestamp 175,000 (ms) from its table? (Hint: use the receiver's local clock value.)

iii) Should the clocks be internally synchronized or externally synchronized? *page 601*

14.4 A client attempts to synchronize with a time server. It records the round-trip times and timestamps returned by the server in the table below.

Which of these times should it use to set its clock? To what time should it set it? Estimate the accuracy of the setting with respect to the server's clock. If it is known that the time between sending and receiving a message in the system concerned is at least 8 ms, do your answers change?

Round-trip (ms)	*Time (hr:min:sec)*
22	*10:54:23.674*
25	*10:54:25.450*
20	*10:54:28.342*

page 601

14.5 In the system of Exercise 14.4 it is required to synchronize a file server's clock to within ±1 millisecond. Discuss this in relation to Cristian's algorithm. *page 601*

+

We used the following references in the composition of the present work (In order of importance):

1. Dollimore, Kindberg, Blair, Coulouris
   Distributed Systems 5th ed, Ch. 14
   Prentice Hall 2012
2. Andrew Tannenbaum
   Distributed Systems, Ch. 5
   Prentice-Hall 2005
3. Flaviu Cristian's article on clock synchronization
4. Leandro Navarro Moldes
   Conceptos de Sistemas Distribuidos
   UOC
5. IETF RFC's 958 and 5905