Chapter 14

Time, Clocks & GPS

“A man with a clock knows what time it is – a man with two is never sure.” (Segal’s Law)

14.1 Time & Clocks


Remarks:
- This definition is a bit simplified. The official definition is given by the Bureau International des Poids et Mesures.
- Historically, a second was defined as one in 86,400 parts of a day, dividing the day into 24 hours, 60 minutes and 60 seconds.
- Since the duration of a day depends on the unsteady rotation cycle of the Earth, the novel oscillation-based definition has been adopted. Leap seconds are used to keep time synchronized to Earth’s rotation.

Definition 14.2 (Wall-Clock Time). The wall-clock time $t^*$ is the true time (a perfectly accurate clock would show).

Definition 14.3 (Clock). A clock is a device which tracks and indicates time.

Remarks:
- A clock’s time $t$ is a function of the wall-clock time $t^*$, i.e., $t = f(t^*)$. Ideally, $t = t^*$, but in reality there are often errors.

Definition 14.4 (Clock Error). The clock error or clock skew is the difference between two clocks, e.g., $t - t^*$ or $t - t^*$. In practice the clock error is often modeled as $t = (1 + \delta)t^* + \xi(t^*)$. 

127
Remarks:

- The importance of accurate timekeeping and clock synchronization is reflected in the following statement by physicist Steven Jefferts: “We’ve learned that every time we build a better clock, somebody comes up with a use for it that you couldn’t have foreseen.”

**Definition 14.5 (Drift).** The drift $\delta$ is the predictable clock error.

Remarks:

- Drift is relatively constant over time, but may change with supply voltage, temperature and age of an oscillator.
- Stable clock sources, which offer a low drift, are generally preferred, but also more expensive, larger and more power hungry, which is why many consumer products feature inaccurate clocks.

**Definition 14.6 (ppm).** Clock drift is indicated in parts-per-million (ppm). One ppm corresponds to a time error growth of one microsecond per second.

Remarks:

- In PCs, the so-called real-time clock normally is a crystal oscillator with a maximum drift between 5 and 100 ppm.
- Applications in signal processing, for instance GPS, need more accurate clocks. Common drift values are 0.5 to 2 ppm.

**Definition 14.7 (Jitter).** The jitter $\xi$ is the unpredictable, random noise of the clock error.

Remarks:

- In other words, jitter is the irregularity of the clock. Unlike drift, jitter can vary fast.
- Jitter captures all the errors that are not explained by drift. Figure 14.8 visualizes the concepts.
14.2 Clock Synchronization

**Definition 14.9 (Clock Synchronization).** Clock synchronization is the process of matching multiple clocks (nodes) to have a common time.

**Remarks:**
- A trade-off exists between synchronization accuracy, convergence time, and cost.
- Different clock synchronization variants may tolerate crashing, erroneous or byzantine nodes.

**Algorithm 14.10 Network Time Protocol NTP**

1: Two nodes, client $u$ and server $v$

2: while true do
3: Node $u$ sends request to $v$ at time $t_u$
4: Node $v$ receives request at time $t_v$
5: Node $v$ processes the request and replies at time $t_v'$
6: Node $u$ receives the response at time $t_u'$

7: Propagation delay $\delta = \frac{(t_u' - t_u) - (t_v' - t_v)}{2}$ (assumption: symmetric)
8: Clock skew $\theta = \frac{(t_u - (t_u + \delta)) - (t_v' - (t_v' + \delta))}{2} = \frac{(t_u - t_u') + (t_v' - t_v')}{2}$
9: Node $u$ adjusts clock by $+\theta$
10: Sleep before next synchronization
11: end while

**Remarks:**
- Many NTP servers are public, answering to UDP packets.
- The most accurate NTP servers derive their time from atomic clocks, synchronized to UTC. To reduce those server’s load, a hierarchy of NTP servers is available in a forest (multiple trees) structure.
- The regular synchronization of NTP limits the maximum error despite unpredictable clock errors. Synchronizing clocks just once is only sufficient for a short time period.

**Definition 14.11 (PTP).** The Precision Time Protocol (PTP) is a clock synchronization protocol similar to NTP, but which uses medium access control (MAC) layer timestamps.

**Remarks:**
- MAC layer timestamping removes the unknown time delay incurred through messages passing through the software stack.
- PTP can achieve sub-microsecond accuracy in local networks.

**Definition 14.12 (Global Synchronization).** Global synchronization establishes a common time between any two nodes in the system.
Remarks:

- For example, email needs global timestamps. Also, event detection for power grid control and earthquake localization need global timestamps.

- Earthquake localization does not need real-time synchronization; it is sufficient if a common time can be reconstructed when needed, also known as "post factum" synchronization.

- NTP and PTP are both examples of clock synchronization algorithms that optimize for global synchronization.

- However, two nodes that constantly communicate may receive their timestamps through different paths of the NTP forest, and hence they may accumulate different errors. Because of the clock skew, a message sent by node $u$ might arrive at node $v$ with a timestamp in the future.

Algorithm 14.13 Local Time Synchronization

1: while true do
2:   Exchange current time with neighbors
3:   Adapt time to neighbors, e.g., to average or median
4:   Sleep before next synchronization
5: end while

Definition 14.14 (Local Synchronization). **Local synchronization establishes a common time between close-by (neighbor) nodes.**

Remarks:

- Local synchronization is the method of choice to establish time-division multiple access (TDMA) and coordination of wake-up and sleeping times in wireless networks. Only close-by nodes matter as far-away nodes will not interfere with their transmissions.

- Local synchronization is also relevant for precise event localization. For instance, using the speed of sound, measured sound arrival times from co-located sensors can be used to localize a shooter.

- While global synchronization algorithm such as NTP usually synchronize to an external time standard, local algorithms often just synchronize among themselves, i.e., the notion of time does not reflect any time standards.

- In wireless networks, one can simplify and improve synchronization.
14.3. TIME STANDARDS

**Algorithm 14.15** Wireless Clock Synchronization with Known Delays

1: Given: transmitter $s$, receivers $u,v$, with known transmission delays $d_u,d_v$ from transmitter $s$, respectively.

2: $s$ sends signal at time $t_s$
3: $u$ receives signal at time $t_u$
4: $v$ receives signal at time $t_v$

5: $\Delta_u = t_u - (t_s + d_u)$
6: $\Delta_v = t_v - (t_s + d_v)$

7: Clock skew between $u$ and $v$: $\theta = \Delta_v - \Delta_u = t_v - d_v + d_u - t_u$

14.3 Time Standards

**Definition 14.16** (TAI). The International Atomic Time (TAI) is a time standard derived from over 400 atomic clocks distributed worldwide.

Remarks:

- Using a weighted average of all involved clocks, TAI is an order of magnitude more stable than the best clock.
- The involved clocks are synchronized using simultaneous observations of GPS or geostationary satellite transmissions using Algorithm 14.15.
- While a single satellite measurement has a time uncertainty on the order of nanoseconds, averaging over a month improves the accuracy by several orders of magnitude.

**Definition 14.17** (Leap Second). A leap second is an extra second added to a minute to make it irregularly 61 instead of 60 seconds long.

Remarks:

- Time standards use leap seconds to compensate for the slowing of the Earth’s rotation. In theory, also negative leap seconds can be used to make some minutes only 59 seconds long. But so far, this was never necessary.
- For easy implementation, not all time standards use leap seconds, for instance TAI and GPS time do not.

**Definition 14.18** (UTC). The Coordinated Universal Time (UTC) is a time standard based on TAI with leap seconds added at irregular intervals to keep it close to mean solar time at 0° longitude.
Remarks:

- The global time standard *Greenwich Mean Time (GMT)* was already established in 1884. With the invention of caesium atomic clocks and the subsequent redefinition of the SI second, UTC replaced GMT in 1967.

- Before time standards existed, each city set their own time according to the local mean solar time, which is difficult to measure exactly. This was changed by the upcoming rail and communication networks.

- Different notations for time and date are in use. A standardized format for timestamps, mostly used for processing by computers, is the ISO 8601 standard. According to this standard, a UTC timestamp looks like this: 1712-02-30T07:39:52Z. T separates the date and time parts while Z indicates the time zone with zero offset from UTC.

- Why UTC and not “CUT”? Because France insisted. Same for other abbreviations in this domain, e.g. TAI.

**Definition 14.19 (Time Zone).** A *time zone* is a geographical region in which the same time offset from UTC is officially used.

Remarks:

- Time zones serve to roughly synchronize noon with the sun reaching the day’s highest apparent elevation angle.

- Some time zones’ offset is not a whole number of hours, e.g. India.

### 14.4 Clock Sources

**Definition 14.20 (Atomic Clock).** An *atomic clock* is a clock which keeps time by counting oscillations of atoms.

Remarks:

- Atomic clocks are the most accurate clocks known. They can have a drift of only about one second in 150 million years, about 2e-10 ppm!

- Many atomic clocks are based on caesium atoms, which led to the current definition of a second. Others use hydrogen-1 or rubidium-87.

- In the future, atoms with higher frequency oscillations could yield even more accurate clocks.

- Atomic clocks are getting smaller and more energy efficient. Chip-scale atomic clocks (CSAC) are currently being produced for space applications and may eventually find their way into consumer electronics.

- Atomic clocks can serve as a fallback for GPS time in data centers.

**Definition 14.21 (System Clock).** The *system clock* in a computer is an oscillator used to synchronize all components on the motherboard.
14.4. CLOCK SOURCES

Remarks:

- Usually, a quartz crystal oscillator with a frequency of some tens to hundreds MHz is used.
- Therefore, the system clock can achieve a precision of some ns!
- The CPU clock is usually a multiple of the system clock, generated from the system clock through a clock multiplier.
- To guarantee nominal operation of the computer, the system clock must have low jitter. Otherwise, some components might not get enough time to complete their operation before the next (early) clock pulse arrives.
- Drift however is not critical for system stability.
- Applications of the system clock include thread scheduling and ensuring smooth media playback.
- If a computer is shut down, the system clock is not running; it is reinitialized when starting the computer.

**Definition 14.22** (RTC). The *real-time clock (RTC)* in a computer is a battery backed oscillator which is running even if the computer is shut down or unplugged.

Remarks:

- The RTC is read at system startup to initialize the system clock.
- This keeps the computer’s time close to UTC even when the time cannot be synchronized over a network.
- RTCs are relatively inaccurate, with a common maximum drift of 5, 20 or even 100 ppm, depending on quality and temperature.
- In many cases, the RTC frequency is 32.768 kHz, which allows for simple timekeeping based on binary counter circuits because the frequency is exactly $2^{15}$ Hz.

**Definition 14.23** (Radio Time Signal). A *Radio Time Signal* is a time code transmitted via radio waves by a time signal station, referring to a time in a given standard such as UTC.

Remarks:

- Time signal stations use atomic clocks to send as accurate time codes as possible.
- Radio-controlled clocks are an example application of radio signal time synchronization.
- In Europe, most radio-controlled clocks use the signal transmitted by the DCF77 station near Frankfurt, Germany.
Radio time signals can be received much farther than the horizon of the transmitter due to signal reflections at the ionosphere. DCF77 for instance has an official range of 2,000 km.

The time synchronization accuracy when using radio time signals is limited by the propagation delay of the signal. For instance the delay Frankfurt–Zurich is about 1 ms.

**Definition 14.24 (Power Line Clock).** A **power line clock** measures the oscillations from electric AC power lines, e.g. 50 Hz.

**Remarks:**
- Clocks in kitchen ovens are usually driven by power line oscillations.
- AC power line oscillations drift about 10 ppm, which is remarkably stable.
- The magnetic field radiating from power lines is strong enough that power line clocks can work wirelessly.
- Power line clocks can be synchronized by matching the observed noisy power line oscillation patterns.
- Power line clocks operate with as little as a few ten µW.

**Definition 14.25 (Sunlight Time Synchronization).** **Sunlight time synchronization** is a method of reconstructing global timestamps by correlating annual solar patterns from light sensors’ length of day measurements.

**Remarks:**
- Sunlight time synchronization is relatively inaccurate.
- Due to low data rates from length of day measurements, sunlight time synchronization is well-suited for long-time measurements with data storage and post-processing, requiring no communication at the time of measurement.
- Historically, sun and lunar observations were the first measurements used for time determination. Some clock towers still feature sun dials.
- ...but today, the most popular source of time is probably GPS!

### 14.5 GPS

**Definition 14.26 (Global Positioning System).** The **Global Positioning System (GPS)** is a Global Navigation Satellite System (GNSS), consisting of at least 24 satellites orbiting around the Earth, each continuously transmitting its position and time code.
Remarks:

• Positioning is done in space and time!

• GPS provides position and time information to receivers anywhere on Earth where at least four satellite signals can be received.

• Line of sight (LOS) between satellite and receiver is advantageous. GPS works poorly indoors, or with reflections.

• Besides the US GPS, three other GNSS exist: the European Galileo, the Russian GLONASS and the Chinese BeiDou.

• GPS satellites orbit around Earth approximately 20,000 km above the surface, circling Earth twice a day. The signals take between 64 and 89 ms to reach Earth.

• The orbits are precisely determined by ground control stations, optimized for a high number of satellites being concurrently above the horizon at any place on Earth.

Algorithm 14.27 GPS Satellite

1: Given: Each satellite has a unique 1023 bit (±1, see below) PRN sequence, plus some current navigation data D (also ±1).
2: The code below is a bit simplified, concentrating on the digital aspects, ignoring that the data is sent on a carrier frequency of 1575.42 MHz.
3: while true do
4:   for all bits $D_i \in D$ do
5:     for $j = 0 \ldots 19$ do
6:       for $k = 0 \ldots 1022$ do \{this loop takes exactly 1 ms\}
7:         Send bit $PRN_k \cdot D_i$
8:       end for
9:     end for
10:   end for
11: end while

Definition 14.28 (PRN). Pseudo-Random Noise (PRN) sequences are pseudo-random bit strings. Each GPS satellite uses a unique PRN sequence with a length of 1023 bits for its signal transmissions.

Remarks:

• The GPS PRN sequences are so-called Gold codes, which have low cross-correlation with each other.

• To simplify our math (abstract from modulation), each PRN bit is either 1 or −1.

Definition 14.29 (Navigation Data). Navigation Data is the data transmitted from satellites, which includes orbit parameters to determine satellite positions, timestamps of signal transmission, atmospheric delay estimations and status information of the satellites and GPS as a whole, such as the accuracy and validity of the data.
Remarks:

- As seen in Algorithm 14.27 each bit is repeated 20 times for better robustness. Thus, the navigation data rate is only 50 bit/s.
- Due to this limited data rate, timestamps are sent every 6 seconds, satellite orbit parameters (function of the satellite position over time) only every 30 seconds. As a result, the latency of a first position estimate after turning on a receiver, which is called \textit{time-to-first-fix} (TTFF), can be high.

**Definition 14.30** (Circular Cross-Correlation). The \textit{circular cross-correlation} is a similarity measure between two vectors of length $N$, circularly shifted by a given displacement $d$:

$$
cxcorr(a, b, d) = \sum_{i=0}^{N-1} a_i \cdot b_{i+d \mod N}
$$

Remarks:

- The two vectors are most similar at the displacement $d$ where the sum (cross-correlation value) is maximum.

**Algorithm 14.31** Acquisition

1: Received 1 ms signal $s$ with sampling rate $r \cdot 1,023 \text{ kHz}$
2: Possible Doppler shifts $F$, e.g. \{-10 kHz, -9.8 kHz, \ldots, +10 kHz\}
3: Tensor $A = 0$: Satellite $\times$ carrier frequency $\times$ time

4: for all satellites $i$ do
5: \hspace{1em} $PRN_i' = PRN_i$ stretched with ratio $r$
6: \hspace{1em} for all Doppler shifts $f \in F$ do
7: \hspace{2em} Build modulated $PRN_i''$ with $PRN_i'$ and Doppler frequency $f$
8: \hspace{2em} for all delays $d \in \{0, 1, \ldots, 1,023 \cdot r - 1\}$ do
9: \hspace{3em} $A_i(f, d) = |cxcorr(s, PRN_i'', d)|$
10: \hspace{2em} end for
11: \hspace{1em} end for
12: Select $d*$ that maximizes $\max_d \max_f A_i(f, d)$
13: Signal arrival time $r_i = d*/(r \cdot 1,023 \text{ kHz})$
14: end for

Remarks:

- Multiple milliseconds of acquisition can be summed up to average out noise and therefore improve the arrival time detection probability.

**Definition 14.32** (Acquisition). \textit{Acquisition} is the process in a GPS receiver that finds the visible satellite signals and detects the delays of the PRN sequences and the Doppler shifts of the signals.
Remarks:

- The relative speed between satellite and receiver introduces a significant Doppler shift to the carrier frequency. In order to decode the signal, a frequency search for the Doppler shift is necessary.

- The nested loops make acquisition the computationally most intensive part of a GPS receiver.

Algorithm 14.33 Classic GPS Receiver

1: \( h \): Unknown receiver handset position
2: \( \theta \): Unknown handset time offset to GPS system time
3: \( r_i \): measured signal arrival time in handset time system
4: \( c \): signal propagation speed (GPS: speed of light)
5: Perform Acquisition (Algorithm 14.31)
6: Track signals and decode navigation data
7: \textbf{for all} satellites \( i \) \textbf{do}
8: Using navigation data, determine signal transmit time \( s_i \) and position \( p_i \)
9: Measured satellite transmission delay \( d_i = r_i - s_i \)
10: \textbf{end for}
11: Solve the following system of equations for \( h \) and \( \theta \):
12: \( \frac{||p_i - h||}{c} = d_i - \theta \), for all \( i \)

Remarks:

- GPS satellites carry precise atomic clocks, but the receiver is not synchronized with the satellites. The arrival times of the signals at the receiver are determined in the receiver’s local time. Therefore, even though the satellite signals include transmit timestamps, the exact distance between satellites and receiver is unknown.

- In total, the positioning problem contains four unknown variables, three for the handset’s spatial position and one for its time offset from the system time. Therefore, signals from at least four transmitters are needed to find the correct solution.

- Since the equations are quadratic (distance), with as many observations as variables, the system of equations has two solutions in principle. For GPS however, in practice one of the solutions is far from the Earth surface, so the correct solution can always be identified without a fifth satellite.

- More received signals help reducing the measurement noise and thus improving the accuracy.

- Since the positioning solution, which is also called position fix, includes the handset’s time offset \( \Delta \), this establishes a global time for all handsets. Thus, GPS is useful for global time synchronization.
• For a handset with unknown position, GPS timing is more accurate than time synchronization with a single transmitter, like a time signal station (cf. Definition 14.23). With the latter, the unknown signal propagation delays cannot be accounted for.

**Definition 14.34 (A-GPS).** An Assisted GPS (A-GPS) receiver fetches the satellite orbit parameters and other navigation data from the Internet, for instance via a cellular network.

**Remarks:**

• A-GPS reduces the data transmission time, and thus the TTFF, from a maximum of 30 seconds per satellite to a maximum of 6 seconds.

• Smartphones regularly use A-GPS. However, coarse positioning is usually done based on nearby Wi-Fi base stations only, which saves energy compared to GPS.

• Another GPS improvement is Differential GPS (DGPS): A receiver with a fixed location within a few kilometers of a mobile receiver compares the observed and actual satellite distances. This error is then subtracted at the mobile receiver. DGPS achieves accuracies in the order of 10 cm.

**Definition 14.35 (Snapshot GPS Receiver).** A snapshot receiver is a GPS receiver that captures one or a few milliseconds of raw GPS signal for a position fix.

**Remarks:**

• Snapshot receivers aim at the remaining latency that results from the transmission of timestamps from the satellites every six seconds.

• Since time changes continuously, timestamps cannot be fetched together with the satellite orbit parameters that are valid for two hours.

• Snapshot receiver can determine the ranges to the satellites modulo 1 ms, which corresponds to 300 km. An approximate time and location of the receiver is used to resolve these ambiguities without a timestamp from the satellite signals themselves.

**Definition 14.36 (CTN).** Coarse Time Navigation (CTN) is a snapshot receiver positioning technique measuring sub-millisecond satellite ranges from correlation peaks, like conventional GPS receivers.

**Remarks:**

• A CTN receiver determines the signal transmit times and satellite positions from its own approximate location by subtracting the signal propagation delay from the receive time. The receiver location and time is not exactly known, but since signals are transmitted exactly whole milliseconds, rounding to the nearest whole millisecond gives the signal transmit time.
• With only a few milliseconds of signal, noise cannot be averaged out well and may lead to wrong signal arrival time estimates. Such wrong measurements usually render the system of equations unsolvable, making positioning infeasible.

Algorithm 14.37 Collective Detection Receiver
1: Given: A raw 1 ms GPS sample $s$, a set $H$ of location/time hypotheses
2: In addition, the receiver learned all navigation and atmospheric data
3: for all hypotheses $h \in H$ do
4: Vector $r = 0$
5: Set $V$ = satellites that should be visible with hypothesis $h$
6: for all satellites $i$ in $V$ do
7: $r = r + r_i$, where $r_i$ is expected signal of satellite $i$. The data of vector $r_i$ incorporates all available information: distance and atmospheric delay between satellite and receiver, frequency shift because of Doppler shift due to satellite movement, current navigation data bit of satellite, etc.
8: end for
9: Probability $P_h = c_{xcorr}(s, r, 0)$
10: end for
11: Solution: hypothesis $h \in H$ maximizing $P_h$

Definition 14.38 (Collective Detection). Collective Detection (CD) is a maximum likelihood snapshot receiver localization method, which does not determine an arrival time for each satellite, but rather combine all the available information and take a decision only at the end of the computation.

Remarks:
• CD can tolerate a few low quality satellite signals and is thus more robust than CTN.
• In essence, CD tests how well position hypotheses match the received signal. For large position and time uncertainties, the high number of hypotheses require a lot of computation power.
• CD can be sped up by a branch and bound approach, which reduces the computation per position fix to the order of one second even for uncertainties of 100 km and a minute.

14.6 Lower Bounds
In the clock synchronization problem, we are given a network (graph) with $n$ nodes. The goal for each node is to have a (logical) clock such that the clock values are well synchronized, and close to real time. Each node is equipped with a hardware (system) clock, that ticks more or less in real time, i.e., the time between two pulses is arbitrary between $[1 - \epsilon, 1 + \epsilon]$, for a constant $\epsilon \ll 1$. We assume that messages sent over the edges of the graph have a delivery time
between $[0, 1]$. In other words, we have a bounded but variable drift on the hardware clocks and an arbitrary jitter in the delivery times. The goal is to design a message-passing algorithm that ensures that the logical clock skew of adjacent nodes is as small as possible at all times.

**Theorem 14.39.** The global clock skew (Definition 14.12) is $\Omega(D)$, where $D$ is the diameter of the network graph.

**Proof.** For a node $u$, let $t_u$ be the logical time of $u$ and let $(u \rightarrow v)$ denote a message sent from $u$ to a node $v$. Let $t(m)$ be the time delay of a message $m$ and let $u$ and $v$ be neighboring nodes. First consider a case where the message delays between $u$ and $v$ are $1/2$. Then all the messages sent by $u$ and $v$ at time $t$ according to the clock of the sender arrive at time $t + 1/2$ according to the clock of the receiver.

Then consider the following cases

- $t_u = t_v + 1/2, t(u \rightarrow v) = 1, t(v \rightarrow u) = 0$
- $t_u = t_v - 1/2, t(u \rightarrow v) = 0, t(v \rightarrow u) = 1,$

where the message delivery time is always fast for one node and slow for the other and the logical clocks are off by $1/2$. In both scenarios, the messages sent at time $i$ according to the clock of the sender arrive at time $i + 1/2$ according to the logical clock of the receiver. Therefore, for nodes $u$ and $v$, both cases with clock drift seem the same as the case with perfectly synchronized clocks. Furthermore, in a linked list of $D$ nodes, the left- and rightmost nodes $l, r$ cannot distinguish $t_l = t_r + D/2$ from $t_l = t_r - D/2$.

**Remarks:**

- From Theorem 14.39, it directly follows that any reasonable clock synchronization algorithm must have a global skew of $\Omega(D)$.
- Many natural algorithms manage to achieve a global clock skew of $\mathcal{O}(D)$.
- As both message jitter and hardware clock drift are bounded by constants, it feels like we should be able to get a constant drift at least between neighboring nodes.
- Let us look at the following algorithm:

**Algorithm 14.40 Local Clock Synchronization (at node $v$)**

1: repeat
2: send logical time $t_v$ to all neighbors
3: if Receive logical time $t_u$, where $t_u > t_v$, from any neighbor $u$ then
4: $t_v = t_u$
5: end if
6: until done

**Lemma 14.41.** The clock synchronization protocol of Algorithm 14.40 has a local skew of $\Omega(n)$. 
14.6. LOWER BOUNDS

Proof. Let the graph be a linked list of $D$ nodes. We denote the nodes by $v_1, v_2, \ldots, v_D$ from left to right and the logical clock of node $v_i$ by $t_i$. Apart from the left-most node $v_1$ all hardware clocks run with speed 1 (real time). Node $v_1$ runs at maximum speed, i.e. the time between two pulses is not 1 but $1 - \epsilon$. Assume that initially all message delays are 1. After some time, node $v_1$ will start to speed up $v_2$, and after some more time $v_2$ will speed up $v_3$, and so on. At some point of time, we will have a clock skew of 1 between any two neighbors. In particular $t_1 = t_D + D - 1$.

Now we start playing around with the message delays. Let $t_1 = T$. First we set the delay between the $v_1$ and $v_2$ to 0. Now node $v_2$ immediately adjusts its logical clock to $T$. After this event (which is instantaneous in our model) we set the delay between $v_2$ and $v_3$ to 0, which results in $v_3$ setting its logical clock to $T$ as well. We perform this successively to all pairs of nodes until $v_{D-2}$ and $v_{D-1}$. Now node $v_{D-1}$ sets its logical clock to $T$, which indicates that the difference between the logical clocks of $v_{D-1}$ and $v_D$ is $T - (T - (D - 1)) = D - 1$.

Remarks:

- The introduced examples may seem cooked-up, but examples like this exist in all networks, and for all algorithms. Indeed, it was shown that any natural clock synchronization algorithm must have a bad local skew. In particular, a protocol that averages between all neighbors (like Algorithm 14.13) is even worse than Algorithm 14.40. An averaging algorithm has a clock skew of $\Omega(D^2)$ in the linked list, at all times.

- It was shown that the local clock skew is $\Theta(\log D)$, i.e., there is a protocol that achieves this bound, and there is a proof that no algorithm can be better than this bound!

- Note that these are worst-case bounds. In practice, clock drift and message delays may not be the worst possible, typically the speed of hardware clocks changes at a comparatively slow pace and the message transmission times follow a benign probability distribution. If we assume this, better protocols do exist, in theory as well as in practice.

Chapter Notes

Atomic clocks are used as a GPS fallback for data center synchronization [CDE+13].

Several people worked on snapshot GPS receivers, but the technique has not penetrated into commercial receivers yet. Liu et al. [LPH+12] presented a practical CTN receiver and reduced the solution space by eliminating solutions not lying on the ground. CD receivers are studied since at least 2011 [ABD+11] and have recently been made practically feasible through branch and bound [BEW17].

It has been known for a long time that the global clock skew is $\Theta(D)$ [LL84, ST87]. The problem of synchronizing the clocks of nearby nodes was introduced by Fan and Lynch in [LF04]; they proved a surprising lower bound of $\Omega(\log D / \log \log D)$ for the local skew. The first algorithm providing a non-trivial local skew of $O(\sqrt{D})$ was given in [LW02]. Later, matching upper and
lower bounds of $\Theta(\log D)$ were given in \cite{LLW10}. The problem has also been
studied in a dynamic setting \cite{KLO09, KLLO10}.

Clock synchronization is a well-studied problem in practice, for instance
regarding the global clock skew in sensor networks, e.g. \cite{EGE02, GKS03, MKSL04, PSJ04}. One more recent line of work is focussing on the problem
of minimizing the local clock skew \cite{BvRW07, SW09, LSW09, FW10, FZTS11}.

This chapter was written in collaboration with Manuel Eichelberger.

Bibliography

\cite{ABD11} Penina Axelrad, Ben K Bradley, James Donna, Megan Mitchell, and
Shan Mohiuddin. Collective Detection and Direct Positioning Using

\cite{BEW17} Pascal Bissig, Manuel Eichelberger, and Roger Wattenhofer. Fast
and Robust GPS Fix Using One Millisecond of Data. In \textit{Information
Processing in Sensor Networks (IPSN), 2017 16th ACM/IEEE

\cite{BvRW07} Nicolas Burri, Pascal von Rickenbach, and Roger Wattenhofer.
\textit{International Conference on Information Processing in Sensor
Networks (IPSN), Cambridge, Massachusetts, USA}, April 2007.

\cite{CDE13} James C Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes,
Christopher Frost, Jeffrey John Furman, Sanjay Ghemawat, Andre
Gubarev, Christopher Heiser, Peter Hochschild, et al. Span-
nner: Google’s globally distributed database. \textit{ACM Transactions on

\cite{EGE02} Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained
Network Time Synchronization Using Reference Broadcasts. \textit{ACM

\cite{FW10} Roland Flury and Roger Wattenhofer. Slotted Programming for
Sensor Networks. In \textit{International Conference on Information
Processing in Sensor Networks (IPSN), Stockhol, Sweden}, April 2010.

\cite{FZTS11} Federico Ferrari, Marco Zimmerling, Lothar Thiele, and Olga Saukh.
Efficient Network Flooding and Time Synchronization with Glossy.
In \textit{Proceedings of the 10th International Conference on Information
Processing in Sensor Networks (IPSN)}, pages 73–84, 2011.

\cite{GKS03} Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava. Timing-
sync Protocol for Sensor Networks. In \textit{Proceedings of the 1st interna-
tional conference on Embedded Networked Sensor Systems (SenSys)},
2003.

\cite{KLLO10} Fabian Kuhn, Christoph Lenzen, Thomas Locher, and Rotem Osh-
man. Optimal Gradient Clock Synchronization in Dynamic Net-
works. In \textit{29th Symposium on Principles of Distributed Computing
(PODC), Zurich, Switzerland}, July 2010.


