# Lecture 2

# Lower Bound on the Local Skew

In the previous lesson, we proved essentially matching upper and lower bounds on the worst-case global skew for the clock synchronization problem. We saw that during an execution of the Max algorithm (Algorithm 1.2), all logical clocks in all executions eventually agree up to an additive term of $\mathcal{O}(uD)$ (ignoring other parameters). The lower bound we proved in Section 1.3 shows that a global skew of $\Omega(uD)$ is unavoidable for any algorithm in which clocks run at an amortized constant rate, at least in the worst case. In our lower bound construction, the two nodes $v$ and $w$ that achieved the maximal skew were distance $D$ apart. However, the lower bound did not preclude neighboring nodes from remaining closely synchronized throughout an execution. In fact, this is straightforward if one is willing to slow down clocks arbitrarily (or simply stop them), even if the amortized rate is constant.

Today, we look into what happens if one requires that clocks progress at a constant rate at all times. In many applications, it is sufficient that neighboring clocks are closely synchronized, while nodes that are further apart are only weakly synchronized. To model this situation, we introduce the *gradient clock synchronization (GCS) problem*. Intuitively, we want to ensure a small skew between neighbors despite maintaining "proper" clocks. That is, we seek to minimize the *local skew* under the requirement that logical clocks always run at least at rate 1.

## 2.1 Formalizing the Problem

Let $G = (V, E)$ be a network. As in the previous lecture, each node $v \in V$ has a hardware clock $H_v : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ that satisfies for all $t, t' \in \mathbb{R}_0^+$ with $t' < t$

$$t - t' \leq H_v(t) - H_v(t') \leq \vartheta \cdot (t - t') \; .$$

Again, we denote by $h_v(t)$ the rate of $H_v(t)$ at time $t$, i.e., $1 \leq h(t) \leq \vartheta$ for all $t \in \mathbb{R}_0^+$. Recall that each node $v$ computes a logical clock $L_v \colon \mathbb{R}_0^+ \to \mathbb{R}_0^+$ from its hardware clock and messages received from neighbors. During an execution $\mathcal{E}$, for each edge $e = \{v, w\} \in E$, we define the *local skew* of $e$ at time $t$ to be

$\mathcal{L}_e(t) = |L_v(t) - L_w(t)|$. The *gradient skew at time* $t$ in the network, denoted $\mathcal{L}(t)$, is the largest local skew across any edge: $\mathcal{L}(t) = \max_{e \in E}\{\mathcal{L}_e(t)\}$. Finally, the *gradient skew over an execution* $\mathcal{E}$ is defined to be

$$\mathcal{L} = \sup_{t \in \mathbb{R}_0^+} \{\mathcal{L}(t)\} .$$

The goal of the *gradient clock synchronization problem* is to minimize $\mathcal{L}$ for any possible execution $\mathcal{E}$.

## 2.2 Averaging Protocols

In this section, we consider a natural strategy for achieving gradient clock synchronization: trying to bring the own logical clock to the average value between the neighbors whose clocks are furthest ahead and behind, respectively. Specifically, each node can be in either *fast mode* or *slow mode*. If a node $v$ detects that its clock is behind the average of its neighbors, it will run in fast mode, and increase its logical clock at a rate faster than its hardware clock by a factor of $1+\mu$, where $\mu$ is some appropriately chosen constant. On the other hand, if $v$'s clock is at least the average of its neighbors, it will run in slow mode, increasing its logical clock only as quickly as its hardware clock. Note that this strategy results in logical clocks that behave like "real" clocks of drift $\vartheta' = \vartheta(1+\mu) - 1$. If $\mu \in \mathcal{O}(\vartheta - 1)$, these clocks are roughly as good as the original hardware clocks.

The idea of switching between fast and slow modes gives a well-defined protocol if neighboring clock values are known precisely.[1] However, ambiguity arises in the presence of uncertainty.

To simplify our presentation of the gradient clock synchronization algorithms we abstract away from the individual messages and message delays from the previous chapter. Instead, we assume that throughout an execution, each node $v$ maintains an estimate of its neighbors' logical clocks. Specifically, for each neighbor $w \in N_v$, $v$ maintains a variable $\tilde{L}_w^v(t)$. The parameter $\delta$ represents the *error* in the estimates: for all $\{v, w\} \in E$ and $t \in \mathbb{R}_0^+$, we have

$$L_w(t) - \delta < \tilde{L}_w^v(t) \leq L_w(t) \tag{2.1}$$

When the node $v$ is clear from context, we will omit the superscript $v$, and simply write $\tilde{L}_w$.

In order to obtain the estimates $\tilde{L}_w^v(t)$, each node $w$ periodically broadcasts its logical clock value to its neighbors. Each neighbor $v$ then computes $\tilde{L}_w^v(t)$ using the known bounds on message delays, and increases $\tilde{L}_w^v$ at rate $h_v/\vartheta$ between messages from $w$. Thus, an upper bound on the error parameter $\delta$ can be computed as a function of $u$ (the uncertainty in message delay), $\vartheta$ (the maximum clock drift), $T$ (the frequency of broadcasts), and $\mu$ (a parameter determining how fast logical clocks may run, see below); you do this in the exercises.

We consider two natural ways of dealing with the uncertainty. Set $L_{N_v}^{\max}(t) := \max_{w \in N_v}\{L_w\}$ and $L_{N_v}^{\min}(t) := \min_{w \in N_v}\{L_w\}$.

[1] There is one issue of pathological behavior in which nodes could switch infinitely quickly between fast and slow modes. This can be avoided by introducing a small threshold $\varepsilon$ so that a node only changes, say, from slow to fast mode if it detects that its clock is $\varepsilon$ time units behind the average.

**Aggresive strategy:** each $v$ computes an *upper bound* on the average between $L_{N_v}^{\max}$ and $L_{N_v}^{\min}$, and determines whether to run in fast or slow mode based on this upper bound;

**Conservative strategy:** each $v$ computes a *lower bound* on the average between $L_{N_v}^{\max}$ and $L_{N_v}^{\min}$ and determines the mode accordingly.

We will see that both strategies give bad results, but for opposite reasons.

## Aggressive Averaging

Here we analyze the aggressive averaging protocol described above. Specifically, each node $v \in V$ computes an upper bound on the average of its neighbors' logical clock values:

$$\tilde{L}_v^{\mathrm{up}}(t) = \frac{\max_{w \in N_v}\{\tilde{L}_w\} + \min_{w \in N_v}\{\tilde{L}_w\}}{2} + \delta \geq \frac{L_{N_v}^{\max} + L_{N_v}^{\min}}{2} .$$

The algorithm then increases the logical clock of $v$ at a rate of $h_v(t)$ if $L_v(t) > \tilde{L}_v^{\mathrm{up}}(t)$, and a rate of $(1+\mu)h_v(t)$ otherwise. We show that the algorithm performs poorly for any choice of $\mu \geq 0$.

**Claim 2.1.** *Consider the aggressive averaging protocol on a path network of diameter $D$, i.e., $V = \{1, 2, \ldots, D+1\}$ and $E = \{\{i, i+1\} \,|\, i \in [D]\}$. Then there exists an execution $\mathcal{E}$ such that the gradient skew satisfies $\mathcal{L} \in \Omega(\delta D)$.*

*Proof Sketch.* Throughout the execution, we will assume that all clock estimates are correct: for all $v \in V$ and $w \in N_v$, we have $\tilde{L}_v^w(t) = L_w(t)$. This means that for all $i \in [2, D]$,[2] $\tilde{L}_{v_i}^{\mathrm{up}}(t) = (L_{v_{i-1}}(t) + L_{v_{i+1}}(t))/2 + \delta$, whereas $\tilde{L}_{v_0}^{\mathrm{up}}(t) = L_{v_1}(t) + \delta$ and $\tilde{L}_{v_{D+1}}^{\mathrm{up}} = L_{v_D}(t) + \delta$. Initially, the hardware clock rate of node $v_i$ is $1 + \frac{i(\vartheta-1)}{D}$. Thus, even though all nodes immediately "see" that skew is building up, they all are in fast mode in order to catch up in case they underestimate their neighbors' clock values.

Now let's see what happens to the logical clocks in this execution. While nodes are running fast, skew keeps building up, but the property that $L_{v_i}(t) = (L_{v_{i+1}}(t) - L_{v_{i-1}}(t))$ is maintained at nodes $i \in [2, D]$. In this state, $v_0$ — despite running fast — has no way of catching up to $v_1$. However, at time $\tau_0 \coloneqq \frac{\delta D}{(1+\mu)(\vartheta-1)}$ we would have that $L_{v_D}(\tau_0) = L_{v_{D-1}}(\tau_0) + \delta = \tilde{L}_{v_D}^{\mathrm{up}}(\tau_0)$ and $v_D$ would stop running fast. We set $t_0 \coloneqq \tau_0 - \varepsilon$ for some arbitrarily small $\varepsilon > 0$ and set $h_{v_D}(t) \coloneqq h_{v_{D-1}}(t)$ for all $t \geq t_0$. Thus, all nodes would remain in fast mode until the time $\tau_1 \coloneqq t_0 + \frac{\delta D}{(1+\mu)(\vartheta-1)}$ when we had $L_{v_{D-1}}(\tau_1) = \tilde{L}_{v_{D-1}}^{\mathrm{up}}(\tau_1)$. We set $t_1 \coloneqq \tau_1 - \varepsilon$ and proceed with this construction inductively. Note that, with every hop, the local skew increases by (almost) $2\delta$, as this is the additional skew that $L_{v_i}$ must build up to $L_{v_{i-1}}$ when $L_{v_{i+1}} = L_{v_i}$ in order to increase $\tilde{L}_{v_i}^{\mathrm{up}} - L_{v_i}$ by $\delta$, i.e., for $v_i$ to stop running fast. As $\varepsilon$ is arbitrarily small, we build up a local skew that is arbitrarily close to $(2D-1)\delta$. □

[2] Here, we denote $[a, b] = \{a, a+1, \ldots, b\}$.

**Remarks:**

- The algorithm is also bad in that the above execution results in a global skew of $\Omega(\delta D^2)$. Slight modifications of the algorithm can guarantee better global skew, but similar algorithms will still have large local skew.
- The argument above can be generalized to arbitrary graphs, by taking two nodes $v, w \in V$ in distance $D$ and using the function $d(x) = d(x, v) - d(x, w)$, just as in Lemma 1.5.

### Conservative Averaging

Let's be more careful. Now each node $v \in V$ computes a *lower* bound on the average of its neighbors' logical clock values:

$$\tilde{L}_v^{\text{up}}(t) = \frac{\max_{w \in N_v}\{\tilde{L}_w\} + \min_{w \in N_v}\{\tilde{L}_w\}}{2} \leq \frac{L_{N_v}^{\max} + L_{N_v}^{\min}}{2}\,.$$

The algorithm then increases the logical clock of $v$ at a rate of $h_v(t)$ if $L_v(t) > \tilde{L}_v^{\text{up}}(t)$, and a rate of $(1+\mu)h_v(t)$ otherwise. Again, the algorithm fails to achieve a small local skew.

**Claim 2.2.** *Consider the conservative averaging protocol on a path network of diameter $D$. Then there exists an execution $\mathcal{E}$ such that the gradient skew satisfies $\mathcal{L} \in \Omega(\delta D)$.*

*Proof Sketch.* We use the same hardware clock rates as for the aggressive strategy, except that now for each $v \in V$, $w \in N_w$, and time $t$, we rule that $\tilde{L}_w(t) = L_w(t) - \delta + \varepsilon$ for some arbitrarily small $\varepsilon > 0$. Thus, all nodes are initially in slow mode. We inductively change hardware clock speeds just before nodes would switch to fast mode, building up the exact same skews between logical clocks as in the previous execution. The only difference is that now it does not depend on $\mu$ how long this takes! □

**Remarks:**

- It seems as if we just can't do things right. Both the aggressive and the conservative strategy do not result in a proper response to the gobal distribution of clock values.
- Maybe *no* algorithm can guarantee a small local skew?

## 2.3 Lower Bound with Bounded Clock Rates

In this section, we first prove a lower bound on the worst case local skew of any GCS algorithm, assuming that each logical clock increases at a rate of at most $(1 + \mu)h_v > 1$. That is, for all $v \in V$ and $t, t' \in \mathbb{R}_0^+$ with $t < t'$, we assume $L_v(t') - L_v(t) \leq (1 + \mu)(H_v(t') - H_v(t))$.[3] We use the model of Chapter 1. Moreover, all logical clocks have a minimum rate of 1: for all $v \in V$ and $t, t' \in \mathbb{R}_0^+$ with $t < t'$, we have $L_v(t') - L_v(t) \geq t' - t$. Under these assumptions, we will prove the following theorem.

[3]Note that this assumption does not allow for algorithms that increase their clocks discontinuously. For example, the argument does not apply to the max algorithm presented in Chapter 1.

**Theorem 2.3.** *Any algorithm for the gradient clock synchronization problem with logical clock rates between 1 and $(1+\mu)h_v$ incurs a worst-case gradient skew of $\mathcal{L} \geq (u/4 - (\vartheta-1)d)\log_{\lceil\sigma\rceil} D$, where $\sigma \coloneqq \mu/(\vartheta-1)$.*

To gain some intuition, assume that $(\vartheta-1)d \ll u$, so we can ignore the former term. The basic strategy of the proof is to construct a sequence of executions $\mathcal{E}_0, \mathcal{E}_1, \ldots, \mathcal{E}_\ell$ and times $t_0 < t_1 < \cdots < t_\ell$ such that at each time $t_i$, there exist nodes $v_i, w_i$ satisfying $L_{v_i}(t_i) - L_{w_i}(t_i) \geq i\alpha u \cdot \operatorname{dist}(v_i, w_i)$, for some suitable constant $\alpha$. Our construction works up to $\ell = \Omega(\log_\sigma D)$ with $\operatorname{dist}(v_\ell, w_\ell) = 1$, which gives the desired result.

In more detail, the idea of the proof is to use the "shifting" technique of Lemma 1.5 applied $\ell$ times over closer and closer pairs of nodes. By Lemma 1.5, there is an execution $\mathcal{E}_0$ and a pair of nodes $v_0, w_0$ satisfying $\operatorname{dist}(v_0, w_0) = D$ such that time $t_0 = d + \left(\frac{u}{2(\vartheta-1)} - d\right) D$, we have $L_{v_0}(t_0) - L_{w_0}(t_0) \geq cuD$ for some constant $c > 0$. Fix a shortest path $P$ from $v_0$ to $w_0$. For any pair of nodes $v, w$ along $P$, we define the *average skew* between $v$ and $w$ at time $t$ to be $|L_v(t) - L_w(t)| \,/\, \operatorname{dist}(v, w)$. In particular, the average skew between $v_0$ and $w_0$ is at least $cu$.

We extend the execution $\mathcal{E}_0$ for $t > t_0$ by setting all hardware clock rates to 1 for $t > t_0$ and all message delays to $d - u/2$ (as in the execution $\mathcal{E}$ in Lemma 1.5). By the assumptions that logical clock rates are always between 1 and $1+\mu$, for every $t > t_0$ in the extended execution, we have $L_{v_0}(t) - L_{w_0}(t) \geq cuD - \mu\cdot(t - t_0)$. That is, the average skew between $v_0$ and $w_0$ decreases at a rate of at most $\mu$. By taking $t_1 = t_0 + d + (u/2(\vartheta-1) - d)\cdot k$ for some suitably chosen $k$, there exists a pair of nodes $v_1, w_1$ in $P$ with $\operatorname{dist}(v_1, w_1) = k$ such that the average skew between $v_1$ and $w_1$ at time $t_1$ is at least $cD - \mu\cdot(t_1 - t_0)$ in the execution $\mathcal{E}_0$. We then apply the shifting technique again to the nodes $v_1$ and $w_1$ on the interval $[t_0, t_1]$. In this way we define an execution $\mathcal{E}_1$ in which the skew between $v_1$ and $w_1$ is $\Omega(uk)$ larger than the skew in $\mathcal{E}_0$. Therefore, in $\mathcal{E}_1$, the average skew bewtween $v_1$ and $w_1$ is $cu - \mu + \Omega(u)$. In the proof, we show that by choosing an appropriate $k \in \Theta(D/\sigma)$, we can ensure that the the average skew increases by a constant $c'$ (i.e., that the $\Omega(u)$ term is larger than $\mu$).

In the proof, we iterate the procedure above $\ell \in \Theta(\log D)$ times. In the $i^{\text{th}}$ iteration, we obtain a pair of nodes $v_i, w_i$ at distance $D/(2\sigma)^i$ such that the average skew between $v_i$ and $w_i$ is at least $(c + ic')\cdot u$. Thus, after $\ell = \log_{2\sigma} D$ iterations, the skew between adjacent nodes $v_\ell$ and $w_\ell$ is $\Omega(u\log_\sigma D)$, which gives the desired result.

*Proof of Theorem 2.3.* Note that the claim is vacuous if $(\vartheta-1)d \geq u/4$, so we can assume the opposite in the following. Set $b \coloneqq \lceil 2\sigma\rceil$ and $i_{\max} \coloneqq \lfloor \log_b D\rfloor$. By induction over $i \in [i_{\max}+1]$, we show that we can build up a skew of $(i+2)(u/4-(\vartheta-1)d)d(v,w)$ between nodes $v, w \in V$ in distance $d(v,w) = b^{i_{\max}-i}$ at a time $t_i$ in execution $\mathcal{E}^{(i)}$, such that after time $t_i$ all hardware clock rates are 1 and all sent messages have delays of $d - u/2$.

We anchor the induction at $i = 0$ by applying Lemma 1.5, choosing $t_0$ as in the lemma. We pick two nodes $v, w \in V$ in distance $b^{i_{\max}} \leq D$ of each other such that $L_v^{(\mathcal{E}_1)}(t_0) \geq L_w^{(\mathcal{E}_1)}(t_0)$. Now consider $\mathcal{E}_v$ for this choice of $v, w \in V$, which satisfies $H_v^{(\mathcal{E}_v)}(t_0) = H_v^{(\mathcal{E}_1)}(t_0) + (u/2 - (\vartheta-1)d)d(v,w)$ and $H_w^{(\mathcal{E}_v)}(t_0) = H_w^{(\mathcal{E}_1)}(t_0)$. By indistinguishability of the two executions and the minimum logical

clock rate of 1, we get that

$$\begin{aligned} L_v^{(\mathcal{E}_v)}(t_0) - L_w^{(\mathcal{E}_v)}(t_0) &= L_v^{(\mathcal{E}_1)}\left(t_0 + \left(\frac{u}{2} - (\vartheta-1)d\right)d(v,w)\right) - L_w^{(\mathcal{E}_1)}(t_0) \\ &\geq L_v^{(\mathcal{E}_1)}(t_0) + \left(\frac{u}{2} - (\vartheta-1)d\right)d(v,w) - L_w^{(\mathcal{E}_1)}(t_0) \\ &\geq \left(\frac{u}{2} - (\vartheta-1)d\right)d(v,w)\,. \end{aligned}$$

We obtain $\mathcal{E}^{(0)}$ by changing all hardware clock rates in $\mathcal{E}_v$ to 1 at time $t_0$ and all message delays of messages sent at or after time $t_0$ to $d - u/2$. As this does not affect the logical clock values at time $t_0$ — $\mathcal{E}^{(0)}$ is indistinguishable from $\mathcal{E}_v$ at $x \in V$ until local time $H_x^{(\mathcal{E}^{(0)})}(t_0)$ — this shows the claim for $i = 0$.

For the induction step from $i$ to $i+1$, let $v, w \in V$, $\mathcal{E}^{(i)}$, and $t_i$ be given by the induction hypothesis, i.e.,

$$L_v^{(\mathcal{E}^{(i)})}(t_i) - L_w^{(\mathcal{E}^{(i)})}(t_i) \geq (i+2)\left(\frac{u}{4} - (\vartheta-1)d\right)d(v,w)\,,$$

and from time $t_i$ on all hardware clock rates are 1 and sent messages have delay $d - u/2$. Note that the latter conditions mean that $\mathcal{E}^{(i)}$ behaves exactly like $\mathcal{E}_1$ from Lemma 1.5 from time $t_i$ on, except that some messages sent at times $t < t_i$ may arrive during $[t_i, t_i + d)$. Hence, if we apply the same modifications to $\mathcal{E}^{(i)}$ as to $\mathcal{E}_1$, but starting from time $t_i + d$ instead of time 0, we will show that for any $v', w' \in V$, construct an execution $\mathcal{E}_{v'}$ indistinguishable from $\mathcal{E}^{(i)}$, where

- for all $x \in V$ and $t \geq t_i$, $H_x^{(\mathcal{E}^{(i)})}(t) = H_x^{(\mathcal{E}^{(i)})}(t_i) + t - t_i$,
- $H_{v'}^{(\mathcal{E}_{v'})}(t) = H_{v'}^{(\mathcal{E}^{(i)})}(t) + d(v',w')(u/2 - (\vartheta-1)d)$ for all times $t \geq t_i + d + (u/(2(\vartheta-1)) - d)d(v',w')$, and
- $H_{w'}^{(\mathcal{E}_{v'})}(t) = H_{w'}^{(\mathcal{E}^{(i)})}(t_i) + t - t_i$ for all $t \geq t_i$.

Consider the logical clock values of $v$ and $w$ in $\mathcal{E}^{(i)}$ at time

$$t_{i+1} \coloneqq t_i + d + \left(\frac{u}{2(\vartheta-1)} - d\right)\frac{d(v,w)}{b}\,.$$

Recall that $\frac{\mathrm{d}}{\mathrm{dt}}L_v(t) \geq h_v(t) \geq 1$ and $l_w(t) \leq (1+\mu)h_w(t)$ at all times $t$. As $h_w^{(\mathcal{E}^{(i)})}(t) = 1$ at times $t \geq t_i$, we obtain

$$L_v^{(\mathcal{E}^{(i)})}(t_{i+1}) - L_w^{(\mathcal{E}^{(i)})}(t_{i+1}) \geq L_v^{(\mathcal{E}^{(i)})}(t_i) - L_w^{(\mathcal{E}^{(i)})}(t_i) - \mu(t_{i+1} - t_i)\,. \tag{2.2}$$

Recall that $d(v,w) = b^{i_{\max}-i}$ and that $b = \lceil 2\sigma \rceil$. We split up a shortest path from $v$ to $w$ in $b$ subpaths of length $b^{i_{\max}-(i+1)}$. By the pidgeon hole principle, at least one of these paths must exhibit at least a $1/b$ fraction of the skew between $v$ and $w$, i.e., there are $v', w' \in V$ with $d(v',w') = b^{i_{\max}-(i+1)} = d(v,w)/b$ so

that

$$\begin{aligned}
&L_{v'}^{(\mathcal{E}^{(i)})}(t_{i+1}) - L_{w'}^{(\mathcal{E}^{(i)})}(t_{i+1}) \\
&\geq \frac{L_{v}^{(\mathcal{E}^{(i)})}(t_{i+1}) - L_{w}^{(\mathcal{E}^{(i)})}(t_{i+1})}{b} \\
&\geq \frac{L_{v}^{(\mathcal{E}^{(i)})}(t_{i}) - L_{w}^{(\mathcal{E}^{(i)})}(t_{i}) - \mu(t_{i+1}-t_i)}{b} \\
&= \frac{L_{v}^{(\mathcal{E}^{(i)})}(t_{i}) - L_{w}^{(\mathcal{E}^{(i)})}(t_{i}) - \mu(d + (u/(2(\vartheta-1)) - d)d(v',w'))}{b} \\
&\geq \frac{L_{v}^{(\mathcal{E}^{(i)})}(t_{i}) - L_{w}^{(\mathcal{E}^{(i)})}(t_{i}) - \mu u d(v',w')/(2(\vartheta-1))}{b} \\
&\geq \frac{L_{v}^{(\mathcal{E}^{(i)})}(t_{i}) - L_{w}^{(\mathcal{E}^{(i)})}(t_{i})}{b} - \frac{\mu}{2\sigma(\vartheta-1)} \cdot \frac{u}{2} \cdot d(v',w') \\
&= \frac{L_{v}^{(\mathcal{E}^{(i)})}(t_{i}) - L_{w}^{(\mathcal{E}^{(i)})}(t_{i})}{b} - \frac{u}{4} \cdot d(v',w') \\
&\geq \frac{(i+2)(u/4 - (\vartheta-1)d)d(v,w)}{b} - \frac{u}{4} \cdot d(v',w') \\
&= \left((i+2)\left(\frac{u}{4} - (\vartheta-1)d\right) - \frac{u}{4}\right) d(v',w')\,.
\end{aligned}$$

In other words, as the average skew on a shortest path from $v$ to $w$ did not decrease by more than $u/4$, there most be some subpath of length $d(v,w)/b$ with at least the same average skew. Now we sneak in additional skew by advancing the (hardware and thus also logical) clock of $v'$ using the indistinguishable execution $\mathcal{E}_{v'}$:

$$\begin{aligned}
&L_{v'}^{(\mathcal{E}_v)}(t_{i+1}) - L_{w'}^{(\mathcal{E}_v)}(t_{i+1}) \\
&= L_{v'}^{(\mathcal{E}^{(i)})}\left(t_{i+1} + \left(\frac{u}{2} - (\vartheta-1)d\right) d(v',w')\right) - L_{w'}^{(\mathcal{E}^{(i)})}(t_{i+1}) \\
&\geq L_{v'}^{(\mathcal{E}^{(i)})}(t_{i+1}) + \left(\frac{u}{2} - (\vartheta-1)d\right) d(v',w') - L_{w'}^{(\mathcal{E}^{(i)})}(t_{i+1}) \\
&\geq (i+3)\left(\frac{u}{4} - (\vartheta-1)d\right) d(v',w')\,.
\end{aligned}$$

This completes the induction. Plugging in $i = i_{\max}$ and noting that $\log b = \log\lceil 2\sigma\rceil \leq 1 + \log\lceil\sigma\rceil$, we get an execution in which two nodes at distance $b^0 = 1$ exhibit a skew of at least

$$\begin{aligned}
(i_{\max}+2)\left(\frac{u}{4} - (\vartheta-1)d\right) &\geq \left(\frac{u}{4} - (\vartheta-1)d\right)(1 + \log_b D) \\
&\geq \left(\frac{u}{4} - (\vartheta-1)d\right)\log_{\lceil\sigma\rceil} D\,.
\end{aligned}$$

□

**Remarks:**

- It is somewhat "bad form" to adapt Lemma 1.5 on the fly, as we did in the proof. However, the alternative of carefully defining partial executions, how to stitch them together, and proving indistinguishability results in this setting would mean to crack a nut with a sledgehammer.

- By making the base of the logarithm larger (i.e., making paths shorter more quickly), we can reduce the "loss" of skew in each step. Thus, we get a skew of $u/2 - (\vartheta - 1)d - \varepsilon$ per iteration, at the cost of reducing the number of iterations by a factor of $\log \sigma / (\log \sigma - \log \varepsilon^{-1})$. As typically $\sigma \gg 1$, this means that we gain roughly a factor of 2.

- We can gain another factor of 2 by introducing skew more carefully. If we constract $\mathcal{E}_1$ so that messages "in direction of $w$" have delay (roughly) $d-u$ and messages "in direction of $v$" have delay $d$, we can hide $u$ skew per hop. We favored the simpler construction to avoid additional bookkeeping.

- Overall, if $(\vartheta - 1)d \ll u$, $\sigma \gg 1$, and $\log_\sigma D \gg 1$, we can show a lower bound of $(u - \varepsilon) \log_\sigma D$ for some small $\varepsilon > 0$.

- What if $(\vartheta - 1)d$ is comparable to $u$ or even larger? As for a lower bound construction we can always pretend that clock drifts are actually smaller, e.g., $\vartheta' \coloneqq \min\{\vartheta, 1 + u/(4d)\}$, the lower bound does not get weaker if the hardware clocks get worse. On the other hand, we will see that larger $\vartheta$ is not really an issue (up to a "one-time" additive term of $\mathcal{O}((\vartheta - 1)d)$), as we can then bounce messages back and forth between nodes to keep track of time with greater accuracy than the "base clocks" permit.

- What about unbounded clock rates?

## 2.4 Lower Bound with Arbitrary Clock Rates

It can be shown that clock rates $l_v(t) \in \omega(1)$ do not help. That is, if $(\vartheta - 1)d < u/4$, we have that $\mathcal{L} \in \Omega(u \log_{1/(\vartheta-1)} D)$. However, the only (currently known) proof for this is tedious, to the point where it conveys little insight regarding what's going on. Hence, we will settle for a (much) simpler argument by Fan and Lynch showing a slightly weaker lower bound, followed by some intution as to why the stronger result is true as well.

We need a technical lemma stating that, provided that we leave some slack in terms of clock drifts and message delays, we can introduce $\Omega(u)$ hardware clock skew between any pair of neighbors in an indistinguishable manner. As this follows from repetition of previous arguments, we skip the proof.

**Lemma 2.4.** *Let $\mathcal{E}$ be any execution in which hardware clock rates are at most $1 + (\vartheta - 1)/2$ and message delays are in the range $(d - 3u/4, d - u/4)$. Then, for any $\{v, w\} \in E$ and sufficiently large times $t$, there is an indistinguishable execution $\mathcal{E}_v$ such that $L_v^{(\mathcal{E}_v)}(t) = L_v^{(\mathcal{E})}(t + u/4)$ and $L_w^{(\mathcal{E}_v)}(t) = L_w^{(\mathcal{E})}(t)$.*

*Proof Sketch.* The general idea is to use the remaining slack of $u/2$ to hide the additional skew, and the slack in the clock rates to introduce it. We can do this as slowly as needed, just as in the proof of Lemma 1.5. Again, we can choose the clock rates according to the function $d(x)$ defined in Lemma 1.5; as $v$ and $w$ are neighbors here, it can only take on values of $-1$, 0, or 1. □

This is all we need to generalize our lower bound to arbitrarily large logical clock rates.

**Theorem 2.5.** *Assume that $\vartheta \leq 2$. Any algorithm for the gradient clock synchronization problem with logical clock rates of at least 1 incurs a worst-case gradient skew of*

$$\mathcal{L} \in \Omega\left(\left(\frac{u}{4} - (\vartheta - 1)d\right) \log_{(\log D)/(\vartheta-1)} D\right).$$

*Proof.* Set $u' \coloneqq u/2$, $d' \coloneqq d - u/4$, and $\vartheta' \coloneqq 1 + (\vartheta - 1)/2$. We perform the exact same construction as in Theorem 2.3, with three modifications. First, $u$, $d$, and $\vartheta$ are replaced by $u'$, $d'$, and $\vartheta'$. Second, before starting the construction, we wait for sufficiently long so that Lemma 2.4 is applicable to all times when we actually "work," i.e., we let the algorithm run for the required time with hardware clock rates of 1 and message delays of $d' - u'/2$. Third, we assume that $\mu = \log_{1/(\vartheta-1)} D$ in the construction; if ever we attempt to use this (assumed) bound on the clock rates in an inequality and it does not hold, the construction fails.

Now two things can happen. The first is that the construction succeeds. Note that we may assume that $u'/4 > (\vartheta' - 1)d'$, as otherwise $u/4 < (\vartheta - 1)d$, i.e., nothing is to show. Thus, the construction shows a lower bound of

$$\begin{aligned}\left(\frac{u'}{4} - (\vartheta' - 1)d'\right) \log_{\lceil\sigma\rceil} D &> \left(\frac{u}{8} - \frac{(\vartheta - 1)d}{2}\right) \log_{\lceil\mu/(\vartheta'-1)\rceil} D \\ &\in \Omega\left(\left(\frac{u}{4} - (\vartheta - 1)d\right) \log_{\mu/(\vartheta-1)} D\right).\end{aligned}$$

As

$$\begin{aligned}\log_{\mu/(\vartheta-1)} D &= \frac{\log D}{\log \mu - \log(\vartheta - 1)} \\ &= \frac{\log D}{\log(\log D - \log(\vartheta - 1)) - \log(\vartheta - 1)} \\ &\in \Omega\left(\frac{\log D}{\log\log D - \log(\vartheta - 1)}\right) \\ &= \Omega\left(\log_{(\log D)/(\vartheta-1)} D\right),\end{aligned}$$

the claim follows in this case.

On the other hand, if the construction fails, there is an index $i < i_{\max}$ for which (2.2) does not hold — this is the only place where we make use of the fact that logical clocks do not run faster than rate $\mu$. Thus,

$$L_w^{(\mathcal{E}^{(i)})}(t_{i+1}) - L_w^{(\mathcal{E}^{(i)})}(t_i) > \mu(t_{i+1} - t_i)$$

for some $i < i_{\max}$. Recall that in the construction, $d(v, w) = b^{i_{\max}-i} \geq b$ and

$$t_{i+1} - t_i = d + \left(\frac{u}{2(\vartheta - 1)} - d\right) \frac{d(v, w)}{b} > \frac{u}{2(\vartheta - 1)} - d > \frac{u}{4(\vartheta - 1)} \geq \frac{u}{4}.$$

Hence, there must be a time $t \geq t_i$ so that

$$L_w^{(\mathcal{E}^{(i)})}\left(t + \frac{u}{4}\right) - L_w^{(\mathcal{E}^{(i)})}(t) > \frac{\mu u}{4}.$$

Let $x \in N_w$ be arbitrary. By Lemma 2.4, we can construct an execution $\mathcal{E}_w$ so that

$$L_w^{(\mathcal{E}_w)}(t) = L_w^{(\mathcal{E}^{(i)})}\left(t + \frac{u}{4}\right) > L_w^{(\mathcal{E}^{(i)})}(t) + \frac{\mu u}{4}$$

and $L_x^{(\mathcal{E}_w)}(t) = L_x^{(\mathcal{E}^{(i)})}(t)$. Thus, in at least one of the executions, the local skew exceeds

$$\frac{\mu u}{8} = \frac{u}{8} \log_{1/(\vartheta-1)} D\,. \qquad \square$$

We conclude this chapter with the promised intuition regarding the influence of $D$ on the base of the logarithm. Consider a path of length $k$ with a skew of exactly $\alpha$ per hop, for a total of $\alpha k$ between its endpoints. Now suppose that an algorithm cleverly uses a large logical clock rate, perfectly reducing the skew at the same rate between any pair of neighbors. Consider the point in time when the skew has been reduced to, say, $\alpha - u/8$ per hop. The node in the middle of the path has increased its logical clock at half the rate of the endpoint that's catching up — and the nodes in between have been even faster! Denoting this rate by $r$, slipping in hardware clock skew at rate $\vartheta - 1$ means adding logical clock skew at rate at least $r(\vartheta - 1)/2$. So, even if it takes factor $r$ less time to reduce the skew to, say $\alpha - u/8$ per hop than it would for $\mu = 1$, it also takes factor $r/2$ less time to build up additional skew. We would end up with the same result!

**Remarks:**

- Unfortunately, molding this idea into a proof is challenging, and the result is not pretty.
- The $D$ in the base of the logarithm is of little importance unless clocks are of poor quality. A standard quartz oscillator guarantees that $\vartheta - 1 \leq 10^{-5}$. Even a gigantic diameter of $10^5$ would not affect the bound by more than a factor 2 for such clocks!
- The assumption that $\vartheta \leq 2$ in Theorem 2.5 is an artifact of the proof. However, hardware clocks that are this inaccurate hardly deserve the name "clock," so this corner case is not of interest.
- Don't fall into the trap of forgetting that relaxing the model enables better solutions! For instance, if it is not important that clocks make progress at all times (or most of the time), constant local skew can be achieved (buzzword: $\alpha$-synchronizer)!
- The elephant in the room is the large gap between the best algorithms we have seen so far (whose local skew is not very different from their global skew) and the lower bounds we established today, which are exponentially smaller as a function of $D$.
- This was the state of the art after Lynch and Welch introduced the problem and presented the lower bound, confounding the research community. If this gap doesn't picque your curiosity, this lecture series is most likely not meant for you.

## Bibliographic Notes

Gradient clock synchronization was introduced by Fan and Lynch [FL06], who show a lower bound of $\Omega(\log(uD)/\log\log(uD))$ on the local skew. Some researchers found this result rather counter-intuitive, and it triggered a line of research seeking to resolve the question what precisely can be achieved. Meier and Thiele show that essentially the same lower bound arises from bounded communication rates, without uncertainty (i.e., $u = 0$) [**?** ]. Theorem 2.3 follows [LLW10], which also tightens the lower bound for unbounded clock rates by removing the $D$ from the base of the logarithm. In the dynamic setting, one can show bounds on how quickly an edge can be incorporated into the subgraph of edges that satisfy the skew bounds, and asymptotic optimality can be achieved simultaneously with other guarantees [KLO11, KLLO10].

## Bibliography

[FL06] Rui Fan and Nancy Lynch. Gradient Clock Synchronization. *Distributed Computing*, 18(4):255–266, 2006.

[KLLO10] Fabian Kuhn, Christoph Lenzen, Thomas Locher, and Rotem Oshman. Optimal Gradient Clock Synchronization in Dynamic Networks. *CoRR*, abs/1005.2894, 2010.

[KLO11] Fabian Kuhn, Thomas Locher, and Rotem Oshman. Gradient Clock Synchronization in Dynamic Networks. *Theory Comput. Syst.*, 49(4):781–816, 2011.

[LLW10] Christoph Lenzen, Thomas Locher, and Roger Wattenhofer. Tight Bounds for Clock Synchronization. *J. ACM*, 57(2):8:1–8:42, 2010.