Optimal Clock Synchronization with Bounded Rates

Christoph Lenzen, Thomas Locher, Roger Wattenhofer
{lenzen, lochert, wattenhofer}@tik.ee.ethz.ch
Computer Engineering and Networks Laboratory (TIK)
ETH Zurich, 8092 Zurich, Switzerland

TIK Report number 301

May 15, 2009

Abstract

We present a novel clock synchronization algorithm and prove that this algorithm achieves an essentially optimal upper bound on the worst-case clock skew between any two participants in any given distributed system. More importantly, the clock skew that can occur in the worst case between neighboring participants is (asymptotically) at most a factor of two larger than the best possible bound. Furthermore, the algorithm minimizes the number of messages that need to be exchanged in a given time period and also the number of bits that any node must store locally. The algorithm achieves these goals while guaranteeing that the clocks run *smoothly*, i.e., all clock rates are always within adjustable, pre-specified bounds, an essential quality of any practical clock synchronization algorithm. These results all hold in a general model where both the clock drifts and the message delays may vary arbitrarily within pre-specified bounds, and algorithms are bound to act only at discrete clock pulses.

1 Introduction

For many tasks in distributed systems it is required that the participants maintain a common notion of time. Since each participant is typically equipped with its own (hardware) clock, and all clocks advance at slightly different and potentially variable clock rates, the clocks must be *synchronized* from time to time in order to ensure that the clocks do not continually drift apart. To this end, a *clock synchronization algorithm* must be used to counterbalance the *clock skews* caused by the different clock rates. In order to detect clock skews, any clock synchronization algorithm requires that timing information is exchanged among the participants. The algorithm may trigger synchronization messages itself, or timing information may be attached to messages that are sent by other applications ("piggybacking"). Synchronizing clocks is challenging due to the variable (and uncontrollable) delay between the time when synchronization messages are sent and the time when the recipients are able to process them. The distributed system is modeled as an arbitrary connected graph $G = (V, E)$, where the nodes in V denote the participants in the system and each edge $\{v, w\} \in E$ represents a bidirectional communication link between v and w. Each node computes a *logical clock value* based on its local hardware clock value and the messages it received from its neighboring nodes. A clock synchronization algorithm strives to minimize the clock skews between these logical clocks.

The objective of this work is to provide tight upper bounds on the degree of synchronization that can be achieved, taking many parameters such as the maximum delay and the maximum clock drift rate into account.[1] A crucial aspect of clock synchronization, which so far has not received much attention, is that a practical clock synchronization algorithm must ensure that the rates of progress of all logical clock values are always within specific bounds, i.e., the clock values are not allowed to change substantially in a short time. In other words, an algorithm must guarantee that the logical clocks run smoothly at all times. However, bounding the clock rates inhibits the ability of an algorithm to react to clock skews, which have to be kept as small as possible. The worst-case clock skew that can occur between any two nodes when executing a particular algorithm $\mathcal{A}$ on a given graph G is called the *global skew*. For many distributed applications it is essential that each node is well synchronized with nodes in its vicinity, i.e., the clock skew between a node v and another node w must be small at all times if the *distance* $d(v, w)$ between v and w in G is small. This is the case if occurrences of events are only of local importance and do not bear any (immediate) significance for nodes that are not close-by.[2] More formally, the goal is to minimize the worst-case clock skew between any two neighboring nodes, which is referred to as the *local skew*. Apart from bounding the logical clock rates in order to ensure that the clock values do not change abruptly, it may further be desirable to keep the logical clock values as close to real time as possible, i.e., an algorithm should guarantee the best possible real-time approximation in the absence of an external timer.

We propose a simple algorithm that bounds the minimum and maximum progress of the logical clocks and ensures that the logical clock values always remain within a linear envelope of real time, while guaranteeing the best possible bounds on both the global and the local skew. What is more, we show that the message frequency can be kept quite low without increasing the worst-case clock differences significantly, which implies that techniques such as piggybacking can be employed. This is a viable option especially considering that we only require a few bits to be sent in each message, which can be included in (or appended to) any message sent by another application. Furthermore, we assume that the algorithm can only act at certain *clock pulses*. These results imply that the techniques in this work solve the synchronization problem asymptotically optimally with respect to several optimization criteria in a general model.

2 Related Work

There is a large body of work on the fundamental problem of synchronizing clocks in distributed systems. Most work mainly focuses on bounding the skew that may occur between any two clocks and also the communication costs that are required in order to guarantee a certain degree of synchronization (see, e.g., [7, 9, 10, 11]). If the maximum (communication) delay is normalized to 1, it has been shown that a skew of $D/2$ cannot be avoided on any graph G of diameter D [1].[3] A stronger lower bound of roughly D can be shown for clock synchronization algorithms that strive to keep all clock values within a linear envelope of real time [4]. The clock synchronization algorithm by Srikanth and Toueg [11] achieves a bound of $O(D)$ on the skew of any two clocks at all times and is thus asymptotically optimal. The authors further show that the accuracy of their algorithm

[1] The matching lower bounds are proved in [4].

[2] A prominent example is TDMA in wireless networks where nodes depend on locally well synchronized time slots.

[3] In our theorems we do not normalize the maximum delay and state all results with respect to the maximum delay and also the clock drift rate in order to show the dependency of the bounds on these parameters.

with respect to real time is also optimal as all clocks are always within a linear envelope of real time. However, their algorithm incurs a skew of $\Theta(D)$ between neighboring nodes in the worst case.

In their seminal work [2] that introduced the problem of synchronizing clocks of neighboring nodes as accurately as possible, Fan and Lynch showed that no algorithm can avoid a clock skew of $\Omega(\log_b D / \log_b \log_b D)$ between neighboring nodes, where the basis b does not depend on D. The only imposed constraint is that nodes are required to increase their clock values at a given minimum progress rate, which is quite natural given that otherwise clocks could simply be halted once critical skews occur. Subsequently, it has been shown that this bound also holds if all messages arrive instantaneously, but an adversary can determine when synchronization messages may be sent [8]. Recently, this result has been improved to $\Omega(\log_b D)$ [4], also revealing that $b = \Theta(1/\varepsilon)$, the inverse of the maximum relative clock drift.

The first algorithm guaranteeing a sublinear bound on the clock skew between neighboring nodes achieved a bound of $\mathcal{O}\left(\sqrt{\varepsilon D}\right)$ [5, 6], a result which was subsequently improved to $\mathcal{O}(\log D)$ [3]. However, the algorithm achieving the logarithmic bound has several disadvantages: Apart from being quite complicated, the algorithm has the undesirable property that the clock values can "jump" by $\mathcal{O}(\log D)$ in a single time step, and thus the clock values may not change smoothly. Moreover, both the message frequency and the size of the messages are fairly large, which prohibits techniques such as piggybacking and which may imply that the algorithm is not useful in practice. What is more, the base of the logarithm can hardly be increased if ε becomes small. Since typically $\varepsilon \ll 1$, a notable gap to the lower bound remains open. The algorithm presented in this work does not have these shortcomings.

3 Model

A distributed system is modeled as an arbitrary connected graph $G = (V, E)$ of diameter D, where nodes represent computational devices and egdes represent bidirectional communication links. Each node v can communicate with all neighboring nodes by exchanging messages. The set of v's neighbors is denoted by $\mathcal{N}_v := \{w \in V \mid \{v, w\} \in E\}$. We assume that, for any two nodes $u, w \in \mathcal{N}_v$, node v can distinguish u from w, e.g., by means of a port numbering or node identifiers, and also that all communication is reliable, i.e., messages are never lost. The time that passes from the moment a message is sent until the recipient can act upon it may be any value in the range $(0, \mathcal{T}]$. We do not allow a delay of zero since the delivery of messages takes a certain time. The upper bound $\mathcal{T}$ is referred to as the *maximum delay* in the following. The maximum delay subsumes both the maximum message delay, i.e., the maximum time during which a message can be in transit, and the time it may take at most for the recipient to process the message. This general definition of $\mathcal{T}$ allows us to define that local computations require no time. While the bound $\mathcal{T}$ is unknown to the algorithm, we assume that the nodes know an upper bound $\hat{\mathcal{T}} \in \mathcal{O}(\mathcal{T})$ on $\mathcal{T}$.

Each node v is equipped with a *hardware clock* H_v whose value at *real time* t is denoted by $H_v(t)$, i.e., $H_v : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ is a monotonically increasing function. The first node starts its clock at real time $t = 0$. An *initialization message* is then flooded through the network in order to start the clocks at the other nodes. For all $v \in V$ let $t_v \geq 0$ be the time when v is initialized. The value of the hardware clock of v is 0 until time t_v and $H_v(t) := \int_{t_v}^{t} h_v(\tau)\, d\tau$ afterwards, where $h_v(\tau)$ is the *hardware clock rate* of v at time τ. The clock rates may vary over time, but we assume that there

is a constant $0 < \varepsilon < 1$ such that the following condition holds.

$$\forall v \in V \; \forall t \geq t_v : 1 - \varepsilon \leq h_v(t) \leq 1 + \varepsilon.$$

While the exact value of ε is unknown, we assume that the nodes know an upper bound $\hat{\varepsilon}$ that is strictly smaller than one.

Typically, a computational device synchronizes its operations internally based on a *clock pulse*. Therefore, we assume that a node $v \in V$ performs computations and sends and receives messages only at such clock pulses. We define that each node v has a clock pulse at time t if $H_v(t)$ is an integer value.[4] Such a time is called a *tick event at node* v or simply a *tick* at v. Note that this definition implies that

$$\mathcal{T} \geq \frac{1}{1-\varepsilon}, \tag{1}$$

as v might have to wait arbitrarily close to $\frac{1}{1-\varepsilon}$ time until it can process a message that arrived right after the last tick event.

Additionally, each node v has a *logical clock* L_v, which is also a function $L_v : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ whose value until time t_v is 0 as well. It is desirable to keep all logical clock values within an (affine) linear envelope of real time. Therefore, we require that any algorithm fulfills the following condition, which takes the different initialization times t_v into account.

$$\forall v \in V \; \forall \text{ ticks } t \text{ at } v : \; (1-\varepsilon)t - t_v \leq L_v(t) \leq (1+\varepsilon)t. \tag{2}$$

Moreover, we demand that the logical clocks behave normally in the sense that the logical clock values may not change dramatically in a short time. Formally, there are constants $\alpha, \beta \in \mathbb{R}^+$ such that

$$\forall v \in V \; \forall \text{ ticks } t < t' \text{ at } v : \; \alpha(t'-t) \leq L_v(t') - L_v(t) \leq \beta(t'-t). \tag{3}$$

The increased (or lowered) clock rates of the logical clocks allow the nodes to correct differences between the logical clock values in the network. The difference between the values of logical clocks is called *clock skew*. Ideally, the logical clocks behave just like the hardware clocks even in the presence of clock skews, albeit with a slightly worse clock drift, i.e., $\alpha \in 1 - \mathcal{O}(\varepsilon)$ and $\beta \in 1 + \mathcal{O}(\varepsilon)$. Note that Condition (3) implies that clocks are not allowed to run backwards and thus the algorithm can only manipulate the logical clock value by increasing it.

A clock synchronization algorithm $\mathcal{A}$ executed at node v specifies how the logical clock $L_v(t)$ of node v is adapted based on its hardware clock and the information received from its neighbors up to time t in such a way that Conditions (2) and (3) are satisfied. Since the algorithm modifies the value of $L_v(\cdot)$ at discrete points in time, we have to specify the meaning of $L_v(t)$ at times where the clock value changes. If L_v is increased at time t, we define $L_v(t)$ to be the value *after* the algorithm changed it. The same definition applies to all other variables that are modified at time t.

Given a clock synchronization algorithm $\mathcal{A}$ and a (connected) graph G, an *execution* specifies the delays of all messages and also the hardware clock rates of all nodes at each point in time when $\mathcal{A}$ is executed on G. The *global* and the *local skew* are formally defined as follows:

Definition 3.1 (Global Skew) *Given a connected graph $G = (V, E)$ and a clock synchronization algorithm $\mathcal{A}$, the* global skew *is defined as the value $\sup_{\mathcal{E}, v \in V, w \in V, t} \{L_v(t) - L_w(t)\}$, where $\mathcal{E}$ is any execution of $\mathcal{A}$ on G.*

[4] Of course, only clock pulses at times $t \geq t_v$ when v has been initialized are considered.

Definition 3.2 (Local Skew) *Given a connected graph $G = (V, E)$ and a clock synchronization algorithm $\mathcal{A}$, the* local skew *is defined as the value* $\sup_{\mathcal{E}, v \in V, w \in \mathcal{N}_v, t} \{L_v(t) - L_w(t)\}$*, where $\mathcal{E}$ is any execution of $\mathcal{A}$ on G.*

Naturally, the goal of an algorithm $\mathcal{A}$ is to ensure the best possible bounds on both the global and the local skew on any graph G.

4 Algorithm

In this section, we introduce the synchronization algorithm $\mathcal{A}^{\text{opt}}$. In order to synchronize the logical clocks, any node v must perpetually send synchronization messages informing the neighboring nodes about its current clock value L_v. Node v itself adapts its clock value according to the information received from its neighbors. However, the information about the neighboring clock values is not sufficient to guarantee an optimal bound on the global skew, because the neighboring nodes might have similar clock values while the skew to nodes at greater distances may be large. Naturally, nodes may not increase their clock values over the largest received clock value as such a behavior might violate Condition (2). This problem can be solved by including an estimate of the maximum clock value in the network in each message. Although nodes might not be able to react immediately to a new estimate, they enable distant nodes to properly adapt their clock rates by forwarding the estimate to them. Hence, whenever a node v sends a message, it is of the form $\langle L_v, L_v + \Lambda_v^{\max} \rangle$, where $\Lambda_v^{\max} \geq 0$ is the estimated clock skew between v's clock value and the currently largest clock value. Similarly, we define for any $w \in \mathcal{N}_v$ that Λ_v^w is the estimated difference between v's and w's clock value from v's perspective. This variable is updated whenever v receives a new estimate L_v^w of the current clock value of w. Note that $\Lambda_v^w > 0$ ($\Lambda_v^w < 0$) indicates that v assumes that its clock is ahead (behind).

At each tick, any node $v \in V$ receives a set $\mathcal{M}$ of such messages. The set $\mathcal{M}$ may consist of any number of messages from each neighbor, as messages with different delays may be processed at the same tick. Therefore, we define the set $\mathcal{L}_w := \{L_w \mid \langle L_w, L_w + \Lambda_w^{\max} \rangle \in \mathcal{M}\} \cup \{0\}$ of all clock values received from w since the last tick event for each neighbor $w \in \mathcal{N}_v$. The element 0 is merely added to simplify the notation as otherwise the algorithm would have to test whether each $\mathcal{L}_w$ is empty. Analogously, let $\mathcal{L}_{\max} := \{L_u + \Lambda_u^{\max} \mid \langle L_u, L_u + \Lambda_u^{\max} \rangle \in \mathcal{M}\} \cup \{0\}$ denote the set of estimates of the maximum clock value received from all $u \in \mathcal{N}_v$.

The algorithm takes three parameters, ∂H, μ, and κ. The first parameter ∂H determines the *message frequency*: As we will see, each node v sends a message to all neighbors at the latest after its hardware clock H_v has advanced by ∂H. Since each node has tick events whenever its hardware clock value increases by 1 and messages can only be sent at tick events, we require that $\partial H \geq 1$. In order to determine whether the next multiple of ∂H is reached, each node v stores a variable $\tilde{H}_v$, which holds v's estimate of the largest multiple of ∂H that any hardware clock has reached yet. If there are no clock skews, each node v increases its clock value L_v by exactly 1 at each tick, but v may increase L_v by more if its clock is behind. We say that v *raises* its clock value by $R_v > 0$ if v increases its clock value by $1 + R_v$. The parameter $\mu > 0$ bounds the progress rate in that the algorithm demands that the raise R_v is always at most μ. Given the bound ε on the clock drift and the fact that the algorithm increases by at least 1 and at most $1 + \mu$ at each tick event, the amortized logical clock rates are bounded by $\alpha = 1 - \varepsilon$ and $\beta = (1 + \varepsilon)(1 + \mu)$. It may be desirable

to keep the parameter μ as small as possible.[5] The analysis of the parameters reveals that we can set the parameter μ to roughly 24ε, i.e., the precision of the clocks reduces by slightly more than one order of magnitude while clock skews are corrected. Of course, μ can be set to a larger value, which leads to a better bound on the worst-case clock skew between neighboring nodes. The role of the parameter $\kappa \in \Omega(\mathcal{T})$ will become clear in the analysis section. For the moment, the reader might think of κ as a "base unit" that $\mathcal{A}^{\text{opt}}$ uses to measure clock skew. As we will see, the parameter κ ought to be kept as small as possible. The analysis reveals that κ must be set at least to $2(1+\varepsilon)(1+\mu)\mathcal{T} + \mathcal{O}(\mu\partial H + 1/(1-\varepsilon))$, i.e., to slightly more than $2\mathcal{T}$ if $\varepsilon \ll 1 \ll \mathcal{T}$, and μ and ∂H are (sufficiently) small.

As all variables and parameters have been introduced, we can now proceed to describe the algorithm $\mathcal{A}^{\text{opt}}$ in detail. As mentioned before, any node starts the execution of the algorithm by flooding an initialization message through the entire network. We simply define that the first received synchronization message is considered the initialization message. If a node receives a synchronization message for the first time, it executes Algorithm 1.

Algorithm 1 Initialization($\mathcal{M}$)

1: $L_v := 0$; $\tilde{H}_v := \lfloor \max \mathcal{L}_{\max} \rfloor$; $\Lambda_v^{\max} := \max \mathcal{L}_{\max}$; $send := false$
2: **for** $w \in \mathcal{N}_v$ **do**
3: $\quad L_v^w := \max \mathcal{L}_w$
4: $\quad \Lambda_v^w := L_v - L_v^w$
5: Send $\langle L_v, L_v + \Lambda_v^{\max} \rangle$ to all $u \in \mathcal{N}_v$

Note that v might in general receive a set $\mathcal{M}$ of more than one synchronization message at time t_v. The clock values in these messages are used to set all variables to the appropriate initial values. In particular, the largest received clock values L_w are used to initialize L_v^w and Λ_v^w for all $w \in \mathcal{N}_v$, and the largest received estimate of the maximum clock value yields the initial values of $\tilde{H}_v$ and $\Lambda_v^{\max}$. The flag *send*, initialized to *false*, indicates whether messages have to be sent at the current tick event. Since the clocks of the neighboring nodes must be activated as well, the node sends $\langle L_v, L_v + \Lambda_v^{\max} \rangle$ anyway at time t_v (although the flag *send* is *false*).

An initialized node performs three subroutines in the given order at each tick event:

1. **UpdateVariables:** Adapt the local variables according to the information received since the last tick event.
2. **SetClock:** Decide if and by how much the clock value is raised, set the logical clock and adapt the affected local variables.
3. **SendMessage:** Send a message if necessary.

We start by discussing the first subroutine, which is given in Algorithm 2. The subroutine takes into account that L_v is increased by at least 1. For this reason, all variables are updated using $L_v + 1$ as the local clock value. In Line 1 of Algorithm 2, v determines the largest received estimate $L_{\max}$ of the maximum clock value in the network. In Line 2, v decides whether adopting $L_{\max}$ would lead to a larger estimate of the maximum clock value. If this condition is satisfied, v adjusts $\tilde{H}_v$ and $\Lambda_v^{\max}$ accordingly and sets the send flag to *true* since the new estimate needs to be

[5] Note that if $\mu \in \mathcal{O}(\varepsilon)$, the maximum clock rate is bounded by $1 + \mathcal{O}(\varepsilon)$.

Algorithm 2 UpdateVariables($\mathcal{M}$)

1: $L_{\max} := \max \mathcal{L}_{\max}$
2: **if** $L_{\max} \geq L_v + 1 + \Lambda_v^{\max}$ **then**
3: $\quad \tilde{H}_v := \lfloor L_{\max} \rfloor$
4: $\quad \Lambda_v^{\max} := L_{\max} - (L_v + 1)$
5: $\quad send := true$
6: **for** $w \in \mathcal{N}_v$ where $\max \mathcal{L}_w > L_v^w$ **do**
7: $\quad L_v^w := \max \mathcal{L}_w$
8: $\quad \Lambda_v^w := L_v + 1 - L_v^w$
9: $\Lambda_v^{\downarrow} := \max_{u \in \mathcal{N}_v} \{\Lambda_v^u\}$
10: $\Lambda_v^{\uparrow} := \max_{u \in \mathcal{N}_v} \{-\Lambda_v^u\}$

forwarded to the other nodes. This case implies that $L_{\max}$ exceeded the next larger multiple of ∂H, which is exactly $\lfloor L_{\max} \rfloor$.[6] In Lines 6-8, v recomputes Λ_v^w for all $w \in \mathcal{N}_v$ as v might have received more recent, i.e. larger, clock values. The main part of algorithm $\mathcal{A}^{\text{opt}}$, Algorithm 3, uses only the estimated clock skew to the two clocks in v's neighborhood that are ahead and behind the most. For this purpose, we introduce the variables $\Lambda_v^{\uparrow}$ and $\Lambda_v^{\downarrow}$, which are updated in Lines 9 and 10.

The goal of the subroutine *SetClock* is to determine if L_v has to be raised by a certain value R_v. The steps of this subroutine are summarized in Algorithm 3.

Algorithm 3 SetClock()

1: $R_v := \sup \left\{ R \in \mathbb{R} \;\middle|\; \left\lfloor \frac{\Lambda_v^{\uparrow} - R}{\kappa} \right\rfloor \geq \left\lfloor \frac{\Lambda_v^{\downarrow} + R}{\kappa} \right\rfloor \right\}$
2: $R_v := \max \left\{ \min \left\{ \max \left\{ \kappa - \Lambda_v^{\downarrow}, R_v \right\}, \mu, \Lambda_v^{\max} \right\}, 0 \right\}$
3: $L_v := L_v + 1 + R_v$
4: $\Lambda_v^{\max} := \Lambda_v^{\max} - R_v$
5: **for** $w \in \mathcal{N}_v$ **do**
6: $\quad \Lambda_v^w := \Lambda_v^w + R_v$

In Line 1, the node computes which value R_v should take according to the locally observed clock skew. Roughly speaking, the goal of Algorithm 3 is to ensure that the clock skew to the neighbor whose clock is assumed to be behind the most and the clock skew to the neighbor with the largest estimated clock value are the same integer multiple of κ. The variable R_v attains the largest value that satisfies this constraint. More precisely, if $\Lambda_v^{\uparrow} \leq s\kappa$ and $\Lambda_v^{\downarrow} \geq s\kappa$ for some $s \in \mathbb{N}$, v does not raise its clock value, i.e., $R_v = 0$. If there is no integer s that satisfies this condition, $R_v > 0$ becomes exactly the increase of the clock value that causes the condition to hold. Line 1 of Algorithm 3 is a concise formulation of this rule.

The following simple example illustrates that this strategy is more aggressive than always setting the clock to the *average* between the smallest and the largest estimated clock value among all neighbors.[7] If $\Lambda_v^{\uparrow} = \Lambda_v^{\downarrow} = s\kappa + \frac{\kappa}{2}$ for any $s \in \mathbb{N}$, Algorithm 3 sets R_v to $\frac{\kappa}{2}$, whereas v does not increase its clock value, i.e., $R_v = 0$, when the straightforward averaging strategy is used because $\Lambda_v^{\uparrow} = \Lambda_v^{\downarrow}$.

[6] For a proof of this observation, see Lemma A.1 in Appendix A.

[7] Note that the latter fails to guarantee good bounds on the worst-case clock skews [6].

In Line 2, the final value R_v is determined. Any node is allowed to raise its clock at least κ above the smallest estimate of its neighbors' clock value. Therefore, R_v is simply set to the maximum of $\kappa - \Lambda_v^{\downarrow}$ and R_v itself. In other words, algorithm $\mathcal{A}^{\text{opt}}$ tolerates a clock skew of κ, which ensures that the node with the smallest clock value in the network is able to raise its clock value even if it may have received delayed messages from its neighbors containing smaller clock values than its own. However, v is not allowed to raise its clock by more than μ or to set its clock to a larger clock value than the (estimate of) the largest clock value in the network. This condition prevents nodes from violating Condition (2) and gives slow nodes the possibility to catch up as the fast nodes can only increase their clock values at low rates. The outer maximum simply ensures that $R_v \geq 0$. After computing R_v, L_v is set to the new value and the estimated clock differences are adapted accordingly in Lines 4-6. Note that the estimates are only changed by R_v, which means that the nodes assume that the other nodes run at the same clock rate and thus increase their clock values by 1. This technique ensures that the errors in the estimates grow slowly.

Finally, v must decide whether or not to send a message at this tick event, which is determined in Algorithm 4.

Algorithm 4 SendMessage()

1: **if** $L_v + \Lambda_v^{\max} \geq \tilde{H}_v + \partial H$ **then**
2: $\quad$ *send* := *true*
3: $\quad \tilde{H}_v := \tilde{H}_v + \partial H$
4: **if** *send* **then**
5: $\quad$ Send $\langle L_v, L_v + \Lambda_v^{\max} \rangle$ to all $u \in \mathcal{N}_v$
6: $\quad$ *send* := *false*

Node v sends a message if either the flag *send* has been set to *true* in Line 2 of Algorithm 2, i.e., v received an estimate of the maximum clock value larger than its own, or v's estimate of the maximum clock value has reached the next integer multiple of ∂H. In this case, $\tilde{H}_v$ is increased by ∂H, the flag is set to *false* again, and the message $\langle L_v, L_v + \Lambda_v^{\max} \rangle$ is sent to v's neighbors.

5 Skew Bounds

It can be shown that the largest estimate of the maximum clock value in the network increases at a rate of at most $1 + \varepsilon$. Hence, as nodes refrain from raising their clocks above this value, $\mathcal{A}^{\text{opt}}$ satisfies Condition (2).[8] We proceed by discussing the upper bounds on both the global and the local skew when algorithm $\mathcal{A}^{\text{opt}}$ is used.

First, we state the bound on the global skew when executing $\mathcal{A}^{\text{opt}}$ on any graph G.

Theorem 5.1 *The global skew of Algorithm $\mathcal{A}^{\text{opt}}$ is bounded by*

$$\mathcal{G} := (1+\varepsilon)\mathcal{T}D + \frac{2\varepsilon}{1-\varepsilon}\partial H + \frac{6}{1-\varepsilon}. \tag{4}$$

Proof. See Appendix A.2. □

[8] This is a direct consequence from Statement (*iv*) of Lemma A.1 (see Appendix A) and the fact that $\Lambda_v^{\max} \geq 0$ for all $v \in V$ at all times t.

It has been shown that no algorithm can guarantee a better bound on the global skew than $(1+\varepsilon)\mathcal{T}D$ if the algorithm has to satisfy Condition (2) unless it has an extremely accurate estimate of either the maximum delays or drifts [4]. Furthermore, even if $\hat{\mathcal{T}} = \mathcal{T}$ and $\hat{\varepsilon} = \varepsilon$, the best possible bound is $(1-\varepsilon)\mathcal{T}D$, which is only marginally better if $\varepsilon \ll 1$. This result implies that the bound on the global skew of $\mathcal{A}^{\text{opt}}$ is optimal up to a small additive term even if the message frequency is kept low, i.e., ∂H is large. The possibility to keep the message frequency low entails that piggybacking can indeed be used, provided that other applications communicate reasonably often.

Corollary 5.2 *If $\partial H \in o\left(\frac{\mathcal{T}}{\varepsilon}\right)$, the bound on the global skew of Algorithm $\mathcal{A}^{\text{opt}}$ given in Theorem 5.1 is tight up to an additive term of $o\left(\frac{\mathcal{T}}{1-\varepsilon}\right)$ and is thus optimal for $\mathcal{T}D \to \infty$.*

The proof of the bound on the local skew relies on the fact that the maximum length of a path with a given average skew decreases exponentially. This implies that the average skew on paths of length one, i.e., between neighboring nodes, is logarithmically bounded in the diameter D. In particular, we show that the network is always in a *legal state*, which is defined as follows.

Definition 5.3 (Legal State) *Given the integer constant $\sigma \geq 2$, we say that a network is in a* legal state *at time t, if and only if for all $s \in \mathbb{N}_0$ and all nodes $v, w \in V$ at distance*

$$d(v,w) \geq C_s := \frac{2\mathcal{G}}{\kappa}\sigma^{-s}$$

we have that

$$L_v(t) - L_w(t) \leq d(v,w)\left(s + \frac{1}{2}\right)\kappa.$$

Note that Theorem 5.1 shows that the legal state is never violated for any path of length at least C_0, because $L_v - L_w \leq \mathcal{G}$.

Without loss of generality, we can assume that $\frac{2\mathcal{G}}{\kappa}$ is a power of σ, since we could replace the diameter D by a (potentially non-integer) $\hat{D} > D$ such that $\log_\sigma \frac{2\hat{\mathcal{G}}}{\kappa} = \left\lceil \log_\sigma \frac{2\mathcal{G}}{\kappa} \right\rceil$, where $\hat{\mathcal{G}}$ is the bound from Theorem 5.1 with D replaced by $\hat{D}$. Thus, C_s becomes an integer value for $s \in \{0, \dots, s_{\max}\}$, where $s_{\max} := \log_\sigma \frac{2\mathcal{G}}{\kappa} \in \mathbb{N}$. This observation is helpful in that it simplifies the subsequent proofs.

A crucial quantity in the analysis of the local skew is the amount by which a node increases its logical clock value beyond the minimum rate (once it has been initialized). For this purpose, we introduce the following definition.

Definition 5.4 $\forall t_1 \leq t_2 : \mathcal{I}_v(t_1,t_2) := L_v(t_2) - L_v(t_1) - (1-\varepsilon)(t_2 - t_1)$.

Apparently, it holds that $\mathcal{I}_v(t,t) = 0$ for all t. Note that $\mathcal{I}_v(t_1,t_2)$ is the amount by which the increase of node v's clock exceeds $(1-\varepsilon)(t_2-t_1)$ in the interval $(t_1,t_2]$ as by definition $L_v(t_1)$ is the logical clock value *after* v increased its logical clock. Since clock values are only increased at tick events, the value $\mathcal{I}_v$ decreases between ticks. However, as the time between ticks is upper bounded by $\frac{1}{1-\varepsilon}$, $\mathcal{I}_v$ is always greater than -1, i.e.,

$$\forall t_1 \leq t_2 : \mathcal{I}_v(t_1,t_2) > -1. \tag{5}$$

Two lemmas are required in order to prove the main theorem. The first lemma basically states that the larger the clock skew is between two nodes v and w, the faster w can reduce it by increasing its clock value quickly.

Lemma 5.5 *Given $\xi \in \mathbb{R}$, $s \in \mathbb{N}$, and any path $v, w \in V$, suppose at time t_0 the equality*

$$L_v(t_0) - L_w(t_0) = d(v,w)\left(s - \frac{1}{2}\right)\kappa + \xi \tag{6}$$

holds. Define $\bar{t} := t_0 + \frac{\kappa C_{s-1}}{(1-\varepsilon)\mu} + \mathcal{T} + \frac{1}{1-\varepsilon}$. If the network is in a legal state at time t_0, it follows that

$$\mathcal{I}_w(t_0, t) \geq \xi \tag{7}$$

for all tick events $t \geq \bar{t}$ at w.

Proof. See Appendix A.3. □

The second lemma shows that the clock skew can only increase slowly once it reaches a certain level. More precisely, we consider the path with the largest average clock skew of length at least $\lfloor C_{s+1} \rfloor$.[9] Let w be the node with the largest and let v be the nodes with the smallest clock value among all nodes on this path. If the average clock skew on this path exceeds roughly $s\kappa$, then w's logical clock runs at the hardware clock rate, i.e., the clock skew between w and v can only grow further at a rate of at most 2ε. Moreover, the clock skew decreases if v increases its clock value quickly. In order to abbreviate the notation, the following definition is introduced.

Definition 5.6 *Given $s \in \mathbb{N}$ and $v \in V$, define for any time t*

$$\Psi_v^s(t) := \max_{w \in V} \left\{ L_w(t) - L_v(t) - s\kappa d(v,w) \right\}.$$

Lemma 5.7 *Assume that $\Psi_v^s(t) > 0$ for all $t \in (t_0, t_1)$. Then it holds for any time $t \in [t_0, t_1]$ that*

$$\Psi_v^s(t) < \Psi_v^s(t_0) + 2\varepsilon(t - t_0) - \mathcal{I}_v(t_0, t) + \frac{\kappa}{4} - 2 - \mu. \tag{8}$$

Proof. See Appendix A.4. □

The main theorem states that the local skew grows logarithmically with the diameter D of the graph.

Theorem 5.8 *Any execution of Algorithm $\mathcal{A}^{\mathrm{opt}}$ remains always in a legal state, i.e., the skew between any two nodes $v, w \in V$ is bounded by*

$$\mathcal{O}\left(\kappa \log_{\mu/\varepsilon} \frac{\mathcal{T}D}{\kappa d(v,w)}\right).$$

In particular, the local skew of Algorithm $\mathcal{A}^{\mathrm{opt}}$ is bounded by

$$\kappa\left(\left\lceil \log_\sigma \frac{2\mathcal{G}}{\kappa} \right\rceil + \frac{1}{2}\right),$$

where $\sigma = \left\lfloor \frac{\mu(1-\varepsilon)}{8\varepsilon} \right\rfloor - 1$ and $\mathcal{G}$ is the bound on the global skew from Theorem 5.1.

[9] Recall that $C_s \in \mathbb{N}$ for $s \leq s_{\max}$. Thus rounding only occurs if $s = s_{\max}$.

Proof. The exact lower bound on μ that we require in the proof is that

$$\mu \geq \frac{8\varepsilon(\sigma+1)}{1-\varepsilon} \tag{9}$$

for an arbitrary integer $\sigma \geq 2$. For a fixed parameter μ, this condition implies that the base of the logarithm in the bound is exactly $\sigma = \left\lfloor \frac{\mu(1-\varepsilon)}{8\varepsilon} \right\rfloor - 1$.

As argued before, w.l.o.g. we have that C_s is integer for $s \leq s_{\max}$. By definition, a skew of more than $d(v,w)\left(s_{\max}+\frac{1}{2}\right)\kappa$ between L_v and L_w may not occur if $d(v,w) \geq C_{s_{\max}}$, as long as the network is in a legal state. Since $C_{s_{\max}} = 1$, this means that the second statement immediately follows from the first one. Hence, for the sake of contradiction, we assume that $t_{\max} < \infty$ is the first time where the network is not in a legal state. The legal state cannot be violated for $s = 0$, as this would imply that the clock skew between two nodes exceeded $\mathcal{G}$, a contradiction to Theorem 5.1. Thus, nodes $v, w \in V$ at distance $d(v,w) \geq C_s$ exist, for some $s \in \{1, \dots, s_{\max}\}$, such that

$$L_v(t_{\max}) - L_w(t_{\max}) > d(v,w)\left(s+\frac{1}{2}\right)\kappa. \tag{10}$$

Define $t_0 := t_{\max} - \frac{\kappa C_{s-1}}{(1-\varepsilon)\mu} - \mathcal{T} - \frac{2}{1-\varepsilon} > t_{\max} - \frac{\sigma+1}{(1-\varepsilon)\mu}\kappa C_s$. If $\Psi_w^s(t') < 0$ for some $t' \in [t_0, t_{\max}]$, choose the smallest $t \in [t_0, t_{\max}]$ such that $\Psi_w^s(t') \geq 0$ for all $t' \in [t, t_{\max}]$. Such a time must exist because

$$\frac{1}{2}\kappa C_s \overset{(10)}{<} L_v(t_{\max}) - L_w(t_{\max}) - s\kappa d(v,w) \leq \Psi_w^s(t_{\max}). \tag{11}$$

Note that $\Psi_w^s(t) < 1 + \mu$ since no node increases its clock by more than $1+\mu$ at any tick event. We apply Lemma 5.7 to obtain

$$\begin{aligned} \frac{1}{2}\kappa C_s \overset{(11)}{\leq} \Psi_w^s(t_{\max}) &\overset{(8)}{<} 2\varepsilon(t_{\max}-t) - \mathcal{I}_w(t, t_{\max}) + \frac{\kappa}{4} - 2 - \mu + \Psi_w^s(t) \\ &\overset{(5)}{<} 2\varepsilon(t_{\max}-t) + \frac{1}{4}\kappa C_s. \end{aligned}$$

This leads to

$$\frac{\sigma+1}{(1-\varepsilon)\mu}\kappa C_s > t_{\max} - t_0 > t_{\max} - t > \frac{1}{8\varepsilon}\kappa C_s,$$

and thus

$$\mu < \frac{8\varepsilon(\sigma+1)}{1-\varepsilon},$$

a contradiction to Condition (9).

Hence, we have $\Psi_w^s(t) \geq 0$ for all $t \in [t_0, t_{\max}]$. Another application of Lemma 5.7 yields

$$\begin{aligned} \frac{1}{2}\kappa C_s - \Psi_w^s(t_0) &\overset{(11)}{\leq} \Psi_w^s(t_{\max}) - \Psi_w^s(t_0) \\ &\overset{(8)}{<} 2\varepsilon(t_{\max}-t_0) - \mathcal{I}_w(t_0, t_{\max}) + \frac{\kappa}{4} - 1 \\ &< 2\varepsilon\frac{\sigma+1}{(1-\varepsilon)\mu}\kappa C_s - \mathcal{I}_w(t_0, t_{\max}) + \frac{\kappa}{4} - 1. \end{aligned}$$

By rearranging the terms, we get that

$$\begin{aligned}\Psi_w^s(t_0) &> \frac{1}{2}\kappa C_s - 2\varepsilon\frac{\sigma+1}{(1-\varepsilon)\mu}\kappa C_s + \mathcal{I}_w(t_0,t_{\max}) - \frac{\kappa}{4} + 1\\ &\geq \frac{1}{4}\kappa C_s - 2\varepsilon\frac{\sigma+1}{(1-\varepsilon)\mu}\kappa C_s + \mathcal{I}_w(t_0,t_{\max}) + 1. \end{aligned} \tag{12}$$

Let $t \in \left[t_{\max} - \frac{1}{1-\varepsilon}, t_{\max}\right]$ denote the latest tick until $t_{\max}$ at w. We simultaneously use Lemma 5.5 on all nodes $v \in V$ to estimate

$$\begin{aligned}\mathcal{I}_w(t_0,t_{\max}) + 1 \overset{(5)}{\geq} \mathcal{I}_w(t_0,t) &\overset{(6)}{\geq} \max_{v\in V}\left\{L_v(t_0) - L_w(t_0) - \left(s-\frac{1}{2}\right)\kappa d(v,w)\right\}\\ &\geq \Psi_w^s(t_0)\\ &\overset{(12)}{>} \frac{1}{4}\kappa C_s - 2\varepsilon\frac{\sigma+1}{(1-\varepsilon)\mu}\kappa C_s + \mathcal{I}_w(t_0,t_{\max}) + 1.\end{aligned}$$

This implies that $2\varepsilon\frac{\sigma+1}{(1-\varepsilon)\mu}\kappa C_s > \frac{1}{4}\kappa C_s$, which again leads to the contradiction that $\mu < \frac{8\varepsilon(\sigma+1)}{(1-\varepsilon)}$. Thus, indeed the network can never leave a legal state. □

Recall that we can choose $\kappa \in \Theta(\mathcal{T})$. If $\mu \in \Theta(\varepsilon)$ and $\partial H \in \mathcal{O}(\mathcal{T}/\mu) = \mathcal{O}(\mathcal{T}/\varepsilon)$, Theorem 5.8 states that the local skew is upper bounded by $\mathcal{O}(\mathcal{T}\log D)$. Note that choosing $\mu \in \Theta(\varepsilon)$ entails that the maximum logical clock rate β is upper bounded by $1 + \mathcal{O}(\varepsilon)$. If $\alpha = 1 - \mathcal{O}(\varepsilon)$ and $\beta = 1 + \mathcal{O}(\varepsilon)$, the logical clock rates are strongly restricted. In this case, the lower bound states that no algorithm can avoid a local skew of $\Omega(\mathcal{T}\log D)$ [4], which implies that the algorithm is asymptotically optimal. Moreover, if the logical clock rate is allowed to be larger than the hardware clock rate by a constant factor, i.e., $\mu \in \Theta(1)$, and we choose $\partial H \in \mathcal{O}(\mathcal{T})$, the bound on the local skew reduces to $\mathcal{O}\left(\mathcal{T}\log_{1/\varepsilon} D\right)$. In this case, the lower bound is $\Omega\left(\mathcal{T}\log_{1/\varepsilon} D\right)$ [4], which again matches the upper bound. In both cases, we see that $\mathcal{A}^{\text{opt}}$ is asymptotically optimal. More generally, we get the following even stronger result.

Corollary 5.9 *It is possible to choose the parameters μ, ∂H, and κ such that the local skew of $\mathcal{A}^{\text{opt}}$ is at most a factor of $\mathcal{O}\left(\frac{\hat{\mathcal{T}}}{\mathcal{T}}\right)$ worse than the local skew of any synchronization algorithm. Moreover, appropriate choices exist such that this ratio tends to $\frac{2\hat{\mathcal{T}}+\mathcal{O}(1)}{\mathcal{T}}$ for $D \to \infty$ and $\hat{\varepsilon} \to 0$. Thus, if also $\hat{\mathcal{T}} = \mathcal{T}$ and $\mathcal{T} \to \infty$, $\mathcal{A}^{\text{opt}}$ provides an asymptotic approximation ratio of 2.*

6 Complexity

In this section, we discuss the costs of Algorithm $\mathcal{A}^{\text{opt}}$ with regard to several measures. For the sake of simplicity, we assume that $\hat{\varepsilon}$ is upper bounded away from 1, hence factors of $1+\varepsilon$, $1-\varepsilon$, etc. merely introduce constant factors in the bounds.

6.1 Message Complexity

An essential optimization criterion is the frequency of communication required to sustain a given quality of synchronization. If energy consumption due to communication is critical, the average of this value over time, i.e., the *amortized message frequency*, is highly relevant. Corollary A.2

immediately yields that $\mathcal{A}^{\mathrm{opt}}$ exhibits an amortized message frequency of $\Theta(\partial H^{-1})$ at each node. The bound from Theorem 5.8 suggests to choose $\partial H \in \Theta\left(\frac{\hat{\mathcal{T}}}{\mu}\right)$, which for a minimal $\mu \in \Theta(\hat{\varepsilon})$, entails that the amortized message frequency is only $\Theta\left(\frac{\hat{\varepsilon}}{\hat{\mathcal{T}}}\right)$.

However, at up to $\mathcal{O}\left(\frac{\mathcal{G}}{\partial H}\right)$ consecutive ticks a node v might receive values $L_{\max}$, each larger by ∂H than the previous one, which causes v to send at each such tick. Thus, the algorithm in its current form does not guarantee a non-trivial lower bound on the message frequency. The message frequency could be bounded by adding another term in the order of $\Theta(\overline{\partial H})$ to κ and forcing nodes to wait at least $\Theta(\partial H)$ time between two send events. The price of this modification is that the bound on the global skew increases by $\Theta(\varepsilon D\, \partial H)$, since the time it takes to propagate information through the whole network increases by $\mathcal{O}(D\, \partial H)$ while nodes increase the estimate on $\hat{L}^{\max}$ locally at their hardware clock rate. This results in a tunable trade-off between minimum message frequency and global skew, as increasing the former inversely affects the latter. This is, up to constant factors, obviously the best possible trade-off, since in the $\Theta(D\, \partial H)$ time a pair of nodes at distance D may have to act without updates about each other's state, $\Theta(\varepsilon D\, \partial H)$ skew can be built up by manipulating the hardware clock rates. Apparently, with this modification the algorithm can be implemented using piggybacking.

6.2 Bit Complexity

Another important property is the *bit complexity*, i.e., the maximum number of bits that must be sent at a send event. Since the same update information is sent to all neighbors at the same send event, we define that the bit complexity in our model is simply the maximum size of this message. Note that the parameter μ can be encoded as a constant times $1/n$, for a number $n \in \mathbb{N}$. The other parameters, κ and ∂H, can then be encoded as a multiple of μ, which can easily be accomplished by rounding up. Thus, any constantly bounded value can be encoded using $\mathcal{O}(\log \mu^{-1})$ bits. We will use this simple observation in the following.

In order to bound the bit complexity, we cannot send the unbounded clock values. Instead, nodes can simply communicate the progress their clocks made since they last sent a message, which requires $\log \partial H$ bits. However, since we have that $\kappa \in \Omega(\mu \partial H)$, we might as well add another $\overline{\partial H}$ to κ and discretize the sent value in steps of $\mu \partial H$. Thus, for the clock value L_v only $\mathcal{O}(\log \mu^{-1})$ bits are necessary. The estimate $L_v^{\max} + \Lambda_v^{\max}$ exceeds a multiple of ∂H by less than 1. This fraction in the range $[0, 1)$ can be encoded using $\mathcal{O}(\log \mu^{-1})$ bits. The increase of $L_v^{\max} + \Lambda_v^{\max}$, however, might be $\Theta(\mathcal{G})$, which requires $\mathcal{O}(\log \mathcal{T} D)$ bits. This can be avoided by limiting the maximum increase of $L_v^{\max} + \Lambda_v^{\max}$ a node *informs* its neighbors about in a single message to $\lceil (1+\mu) \rceil \partial H \in \mathcal{O}(\partial H)$, which can again be encoded using $\mathcal{O}(1)$ bits. If the actual value is larger, v stores the difference and informs its neighbors about the remaining increase in its subsequent messages. The intuition behind this is that if v receives an estimate of $\hat{L}^{\max}$ much larger than its own, this can only be because it has been propagated by messages with small delays. In the scenario where all messages are as slow as possible, this would not happen. Thus, the estimates of $\hat{L}^{\max}$ that slow nodes receive are still sufficiently large for Theorem 5.1 to hold. Since v sends an update of more than $(1+\mu)\partial H$, the corresponding estimate of $L_v + \Lambda_v^{\max}$ is still larger than L_v. Hence, the statement $\Lambda_w^{\max}(t) \geq \Lambda_w^v(t)$ of Lemma A.1 remains true, ensuring that the proof of Lemma 5.5 and thus also Theorem 5.8 stay correct. We conclude that Algorithm $\mathcal{A}^{\mathrm{opt}}$ can be implemented with a bit complexity of $\mathcal{O}(\log \mu^{-1}) \subseteq \mathcal{O}(\log \hat{\varepsilon}^{-1})$. Note that if all messages must contain globally unique node identifiers, $\mathcal{O}(\log |V|)$ additional bits are required.

6.3 Space Complexity

The *space complexity* of an algorithm is the maximum amount of memory it requires to run. Since the logical clock value L_v grows indefinitely, we disregard it in our analysis of the space complexity of $\mathcal{A}^{\text{opt}}$. Furthermore, we will consider the amount of memory an implementation of $\mathcal{A}^{\text{opt}}$ with a bit complexity of $\mathcal{O}(\log \mu^{-1})$ would need.[10]

Each node v must store the estimated clock skew Λ_v^w to each node $w \in \mathcal{N}_v$ and the estimated difference $\Lambda_v^{\max}$ to the maximum clock value. The value of Λ_v^w is bounded by $\mathcal{O}(\kappa \log_{\mu/\varepsilon} D)$ for all neighbors. If we round to fractions of κ again, the maximum memory requirement for these values is thus bounded by $\mathcal{O}(\Delta \log \log_{\mu/\varepsilon} D)$, where Δ denotes the maximum node degree. As $\Lambda_v^{\max}$ is bounded by $\mathcal{O}(\mathcal{T}D)$, it can be encoded using $\mathcal{O}(\log(\mathcal{T}D) + \log \mu^{-1})$ bits. Furthermore, each node must store the number of ticks since the last message from w was received when adapting Λ_v^w, since in abscence of better information v assumes that the neighbors' clocks run at the same rate as its own hardware clock. This requires at most $\mathcal{O}(\mathcal{T} + \partial H) \subseteq \mathcal{O}(\kappa)$ for each $w \in \mathcal{N}_v$. In total, these integer values require $\mathcal{O}(\Delta \log \kappa)$ bits. The last variable that needs to be stored is $\tilde{H}_v$, where $\tilde{H}_v - L_v \in \mathcal{O}(\mathcal{T}D)$, which costs $\mathcal{O}(\log(\mathcal{T}D) + \log \mu^{-1})$ bits. Overall, the space complexity is $\mathcal{O}(\log \mu^{-1} + \log D + \Delta \log \kappa + \Delta \log \log_{\mu/\varepsilon} D)$.

References

[1] S. Biaz and J. Lundelius Welch. Closed Form Bounds for Clock Synchronization Under Simple Uncertainty Assumptions. *Information Processing Letters*, 80(3):151–157, 2001.

[2] R. Fan and N. Lynch. Gradient Clock Synchronization. In *Proc. 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 320–327, 2004.

[3] C. Lenzen, T. Locher, and R. Wattenhofer. Clock Synchronization with Bounded Global and Local Skew. In *Proc. 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 500–510, 2008.

[4] C. Lenzen, T. Locher, and R. Wattenhofer. Tight Lower Bounds for Clock Synchronization. Technical Report 299, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, 2008. ftp.tik.ee.ethz.ch/pub/publications/TIK-Report-299.pdf.

[5] T. Locher. *Foundations of Aggregation and Synchronization in Distributed Systems.* PhD thesis, ETH Zurich, 2009.

[6] T. Locher and R. Wattenhofer. Oblivious Gradient Clock Synchronization. In *Proc. 20th International Symposium on Distributed Computing (DISC)*, pages 520–533, 2006.

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

[8] L. Meier and L. Thiele. Brief Announcement: Gradient Clock Synchronization in Sensor Networks. In *Proc. 24th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, page 238, 2005.

[10] Note that by putting more information into the messages, the space complexity can be reduced.

[9] R. Ostrovsky and B. Patt-Shamir. Optimal and Efficient Clock Synchronization under Drifting Clocks. In *Proc. 18th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 400–414, 1999.

[10] B. Patt-Shamir and S. Rajsbaum. A Theory of Clock Synchronization. In *Proc. 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 810–819, 1994.

[11] T. K. Srikanth and S. Toueg. Optimal Clock Synchronization. *Journal of the ACM*, 34(3):626–645, 1987.

A Proofs

A.1 General Statements

We require a few additional statements about $\mathcal{I}_v$: Since $\mathcal{I}_v(t_1, t_2)$ is the amount by which the increase of node v exceeds $(1-\varepsilon)(t_2 - t_1)$ in the interval $(t_1, t_2]$, it follows that $\mathcal{I}_v$ is *interval additive*:

$$\forall t_1 \leq t_2 \leq t_3 : \mathcal{I}_v(t_1, t_2) + \mathcal{I}_v(t_2, t_3) = \mathcal{I}_v(t_1, t_3). \tag{13}$$

Since clock values are only increased at tick events, the value $\mathcal{I}_v$ decreases between ticks. However, once the next tick occurs, the clock is increased by at least 1, implying that $\mathcal{I}_v(\cdot, t)$ is at least 0 at any tick t, i.e.,

$$\forall t_1 \leq \operatorname{tick} t_2 : \mathcal{I}_v(t_1, t_2) \geq 0. \tag{14}$$

The clock value L_v is increased by $1 + R_v(t)$ at each tick t, where $R_v(t) \in [0, \mu]$. $\mathcal{I}_v(t_1, t_2)$ is positive at a tick event t_2 if either the nodes' hardware clock rate was greater than $1-\varepsilon$ in the time interval $(t_1, t_2]$ or v raised its clock. Naturally, as any raise also increases the value of $\mathcal{I}_v$, the sum of all clock raises in the interval $(t_1, t_2]$ is a lower bound on $\mathcal{I}_v(t_1, t_2)$. However, $\mathcal{I}_v(t_1, t_2)$ can only be larger than the sum of clock raises if the hardware clock rate was greater than $1-\varepsilon$. Since the hardware clock rate is upper bounded by $1+\varepsilon$, $\mathcal{I}_v$ can increase by at most $2\varepsilon(t_2 - t_1)$ due to the hardware clock. This implies that $\mathcal{I}_v(t_1, t_2) - 2\varepsilon(t_2 - t_1)$ is at most the sum of clock raises. Formally, it holds that

$$\forall \operatorname{tick} t_1 \leq \operatorname{tick} t_2 \ : \sum_{\operatorname{tick} t \in (t_1, t_2]} R_v(t) \leq \mathcal{I}_v(t_1, t_2) \leq \sum_{\operatorname{tick} t \in (t_1, t_2]} R_v(t) + 2\varepsilon(t_2 - t_1). \tag{15}$$

Inequality 9 provides the lower bound on μ that we use in our proofs. As far as the parameter κ is concerned, it must hold that

$$\kappa \geq 2\left((1+\varepsilon)(1+\mu)\left(\mathcal{T} + \frac{1}{1-\varepsilon}\right) + 4 + \mu + \overline{\partial H}\right), \tag{16}$$

where

$$\overline{\partial H} := (2\varepsilon + \mu)\partial H \tag{17}$$

The parameter $\overline{\partial H}$ is introduced to abbreviate the notation. The intuition behind this lower bound is that κ must be large enough to compensate the inaccuracy of the known clock values of the neighboring nodes due to the maximum delay $\mathcal{T}$. Apparently, the inaccuracy is bounded by the maximum progress $(1+\varepsilon)(1+\mu)\mathcal{T}$ of the clocks. The problem that nodes have to wait for clock pulses is compensated by adding the constant terms $\frac{1}{1-\varepsilon}$ and 4. Moreover, in order to give the algorithm a chance to react to clock skews (by means of *clock raises*), the term μ is added. Obviously, the accuracy of the information about neighboring clocks deteriorates if ∂H is set to a large value. Since clock skew can be built up at a rate of at most $\mathcal{O}(\mu)$, the additional skew is bounded by $\mathcal{O}(\mu\partial H)$. Therefore, κ must further include the term $\overline{\partial H}$ as defined above. Finally, as nodes err not only with respect to their fastest, but also with respect to their slowest neighbor's clock, the bound is multiplied by 2. Throughout this paper, it is implicitly assumed that not only Condition (9) but also Condition (16) is satisfied, and all lemmas and theorems make use of this assumption.

We start by proving a few general facts about algorithm $\mathcal{A}^{\mathrm{opt}}$, which will be useful in the following.

Lemma A.1 *For any execution of algorithm $\mathcal{A}^{\text{opt}}$ the following statements hold:*

i. The function $L_v(t) + \Lambda_v^{\max}(t)$ increases by at least 1 at each tick at v.

ii. If $v \in V$ sends a message at tick t, it holds that $L_v(t) + \Lambda_v^{\max}(t) \in \partial H\mathbb{N}_0 + [0, 1)$.

iii. $\Lambda_v^{\max}(t) \geq -\Lambda_v^w(t)$ for all v and $w \in \mathcal{N}_v$ at all tick events t at v.

iv. Define $\hat{L}^{\max}(t) := \max_{v \in V}\{L_v(t) + \Lambda_v^{\max}(t)\}$. For any times $t_1 < t_2$ it holds that

$$\hat{L}^{\max}(t_2) - \hat{L}^{\max}(t_1) \leq \lceil (1 + \varepsilon)(t_2 - t_1) \rceil. \tag{18}$$

Proof. Statement (i): Let $t' < t$ denote two consecutive ticks at v. First, assume that $L_{\max} < L_v(t') + 1 + \Lambda_v^{\max}(t')$ in Line 2 of Algorithm 2 at tick t, implying that $\Lambda_v^{\max}$ remains unchanged in Algorithm 2. In this case, the clock value L_v is increased by $1 + R_v$ according to Line 3 of Algorithm 3, and R_v is subtracted from $\Lambda_v^{\max}$ in Line 4. Hence $L_v + \Lambda_v^{\max}$ increases by $1 + R_v - R_v = 1$.

If $L_{\max} \geq L_v(t') + 1 + \Lambda_v^{\max}(t')$ in Line 2 of Algorithm 2 at time t, then the algorithm computes $L_v(t) = L_v(t') + 1 + R_v(t)$ and $\Lambda_v^{\max}(t) = L_{\max} - (L_v(t') + 1) - R_v(t)$, and thus

$$\begin{aligned} L_v(t) + \Lambda_v^{\max}(t) &= L_{\max} \\ &\geq L_v(t') + \Lambda_v^{\max}(t') + 1. \end{aligned} \tag{19}$$

Statement (ii): Assume for the sake of contradiction that t is the first time where the claim is violated by a node v. Apparently, t cannot be the initialization time t_v of v, since v just forwards the largest received value when it is initialized. Likewise, the condition in Line 2 of Algorithm 2 must be false, since otherwise Equality (19) shows that the sent value is exactly the received value $L_{\max}$.

Hence, if t' denotes the latest tick at v prior to t, the previous observations yield that $L_v(t) + \Lambda_v^{\max}(t) = L_v(t') + 1 + \Lambda_v^{\max}(t')$. Moreover, since v sends a message, the condition in Line 1 of Algorithm 4 must be satisfied. $\tilde{H}_v$ can be manipulated in two ways: Either $\tilde{H}_v$ is set to $\lfloor L_{\max} \rfloor$ in Algorithm 2 when receiving a value $L_{\max}$ or $\tilde{H}_v$ is increased by ∂H in Line 3 of Algorithm 4, which implies that $\tilde{H}_v(t') \in \partial H\mathbb{N}_0$. Furthermore, it holds that $L_v(t') + \Lambda_v^{\max}(t') < \tilde{H}_v(t') + \partial H$: As $\partial H \geq 1$, this is certainly true if t' is the initialization time t_v of v. Due to Equality (19) the same holds if the condition in Line 2 of Algorithm 2 was true at t'. If neither $t' = t_v$ nor the condition was true at t', $L_v + \Lambda_v^{\max}$ increased by exactly 1. In this case, Lines 1 and 3 of Algorithm 4 ensure that $\tilde{H}_v$ has been increased by ∂H, if necessary. We conclude that at the tick t in Line 6 of Algorithm 4 we have that

$$\tilde{H}_v(t') + \partial H \leq L_v(t) + \Lambda_v^{\max}(t) < \tilde{H}_v(t') + \partial H + 1,$$

implying that $L_v(t) + \Lambda_v^{\max}(t) \in \partial H\mathbb{N}_0 + [0, 1)$, contradicting the assumption that this is not the case.

Statement (iii): Let t' again denote the latest tick before the tick t at v. If the condition in Line 2 of Algorithm 2 is true at t, Equality (19) shows that $L_{\max} = L_v(t) + \Lambda_v^{\max}(t)$. Otherwise, we have that $L_{\max} < L_v(t') + \Lambda_v^{\max}(t') + 1 = L_v(t) + \Lambda_v^{\max}(t)$ due to the proof of Statement (i). In both cases we obtain

$$L_{\max} \leq L_v(t) + \Lambda_v^{\max}(t).$$

Note that $\Lambda_v^{\max} \geq 0$, because Line 2 of Algorithm 2 ensures that the value assigned to $\Lambda_v^{\max}$ in Line 4 is not negative, while R_v is upper bounded by $\Lambda_v^{\max}$ in Line 4 of Algorithm 3. Let t_s be the

time when a neighbor $w \in \mathcal{N}_v$ sends the message $\langle L_w(t_s), L_w(t_s) + \Lambda_w^{\max}(t_s)\rangle$ containing the largest clock value $L_w(t_s)$ that v received at time $t_r \leq t$.[11] Line 8 from Algorithm 2 and Lines 3 and 4 of Algorithm 3 show that at time t_r it holds that $\Lambda_v^w(t_r) = L_v(t_r) - L_w(t_s)$. We estimate

$$\Lambda_v^{\max}(t_r) \geq L_{\max} - L_v(t_r) \geq L_w(t_s) + \Lambda_w^{\max}(t_s) - L_v(t_r) \geq L_w(t_s) - L_v(t_r) = -\Lambda_v^w(t_r).$$

Due to Statement (i), $L_v + \Lambda_v^{\max}$ increases by at least 1 at each tick t, hence $\Lambda_v^{\max}$ decreases by at most $R_v(t)$. Since Λ_v^w is increased by exactly $R_v(t)$ at each tick t according to Line 6 of Algorithm 3, it follows that

$$\Lambda_v^{\max}(t) \geq \Lambda_v^{\max}(t_r) - \sum_{\text{tick } t' \in (t_r, t]} R_v(t') \geq -\Lambda_v^w(t_r) - \sum_{\text{tick } t' \in (t_r, t]} R_v(t') = -\Lambda_v^w(t).$$

Statement (iv): Obviously, no node can receive a value $L_{\max}$ that is larger than the largest estimate of the maximum clock value at any time t. Thus, due to Equality (19), $\hat{L}^{\max}$ can only be increased by $v \in V$ at tick t if the condition in Line 2 of Algorithm 2 is false. We already observed that in this case $L_v + \Lambda_v^{\max}$ increases by exactly 1. Since any node can have at most $\lceil (1+\varepsilon)(t_2 - t_1) \rceil$ tick events in the time interval $[t_1, t_2]$, the claim follows. □

Corollary A.2 *1) Any node sends at least one message to each of its neighbors after its hardware clock advanced by ∂H since the last send event. 2) No node sends more than $1 + \frac{\hat{L}^{\max}(t)}{\partial H} \leq 1 + \frac{\lceil (1+\varepsilon)t \rceil}{\partial H}$ messages up to time t.*

The following lemma bounds the inaccuracy of the estimates Λ_v^w on the clock skew between v and w from v's perspective.

Lemma A.3 *Assume that t is a tick event at node $v \in V$ and that v has received at least one message from a neighbor $w \in \mathcal{N}_v$ until time t. Consider the first tick event $t' \geq t - \mathcal{T} - \frac{1}{1-\varepsilon}$ at w. It holds that*

$$\Lambda_v^w(t) < L_v(t) - L_w(t') + \overline{\partial H}. \tag{20}$$

Proof. Let t_s denote the time when w sent the largest clock value that v received at the latest at time t, and let $t_r \leq t$ be the time when v received this clock value. Due to Corollary A.2, if $t_s < t'$, we have that $H_w(t') - H_w(t_s) < \partial H$, as otherwise w would send a message that arrives at v at the latest at time t and contains a larger clock value than the one sent at t_s. Furthermore, $L_w(t') - L_w(t_s)$ is upper bounded by $(1+\mu)(H_w(t') - H_w(t_s))$. Consequently, it holds that

$$\begin{aligned} \mathcal{I}_w(t_s, t') &\leq (1+\mu)(H_w(t') - H_w(t_s)) - (1-\varepsilon)(t' - t_s) \\ &= \left(1 + \mu - \frac{1-\varepsilon}{1+\varepsilon}\right)(H_w(t') - H_w(t_s)) \\ &< (\mu + 2\varepsilon)\partial H \overset{(17)}{=} \overline{\partial H}. \end{aligned} \tag{21}$$

Recall that at time t_r it holds that $\Lambda_v^w(t_r) = L_v(t_r) - L_w(t_s)$. Since t' is the first tick event at time at least $t - \mathcal{T} - \frac{1}{1-\varepsilon}$, we have that $t' < t - \mathcal{T}$. The value of Λ_v^w is increased exactly by R_v at each

[11] If no such time exists, it holds that $\Lambda_v^w(t) \geq 0$, as Λ_v^w is initialized to zero and it increases monotonically as long as no messages from w arrive.

tick in the time interval $(t_r, t]$ according to Line 6 of Algorithm 3. Hence, if $t_s < t'$, we get

$$\begin{aligned}
\Lambda_v^w(t) &= \Lambda_v^w(t_r) + \sum_{\text{tick}\,\tau\in(t_r,t]} R_v(\tau) \\
&\overset{(15)}{\leq} L_v(t_r) + \mathcal{I}_v(t_r,t) - L_w(t_s) \\
&= L_v(t) - (1-\varepsilon)(t-t_r) - (L_w(t') - \mathcal{I}_w(t_s,t') - (1-\varepsilon)(t'-t_s)) \\
&\overset{(21)}{<} L_v(t) - L_w(t') + \overline{\partial H} - (1-\varepsilon)(t-t') + (1-\varepsilon)(t_r - t_s) \\
&< L_v(t) - L_w(t') + \overline{\partial H}.
\end{aligned}$$

In the last step, we simply used that $t - t' > \mathcal{T}$ and $t_r - t_s \leq \mathcal{T}$.

In case $t_s > t'$ we have that $L_w(t_s) > L_w(t')$. If $t_s = t'$, then it holds that $L_w(t_s) = L_w(t')$ and also that $t_r < t$ because $t_s = t' < t - \mathcal{T}$. Thus, if $t_s \geq t'$, we get that

$$\Lambda_v^w(t) \overset{(15)}{\leq} L_v(t_r) + \mathcal{I}_v(t_r,t) - L_w(t_s) = L_v(t) - (1-\varepsilon)(t-t_r) - L_w(t_s) < L_v(t) - L_w(t').$$

□

A.2 Proof of Theorem 5.1

Define $\hat{L}^{\max}(t) := \max_{u\in V}\{L_u(t) + \Lambda_u^{\max}(t)\}$ as in Lemma A.1. Instead of proving the bound on the clock skew directly, we show that $\hat{L}^{\max}(t) - \min_{u\in V}\{L_u(t)\} \leq \mathcal{G}$ holds at any time t. Since $\Lambda_u^{\max}(t) \geq 0$, the claim follows immediately from this statement.

Assume for the sake of contradiction that the claimed bound does not hold at time $t_{\max}$, where w.l.o.g. we may choose $t_{\max}$ to be the first violation. In this case, two nodes $v, w \in V$ exist such that

$$L_v(t_{\max}) + \Lambda_v^{\max}(t_{\max}) - L_w(t_{\max}) > \mathcal{G}, \tag{22}$$

where v has the largest estimate of the maximum clock value and w has the smallest clock value in the system at time $t_{\max}$. Let $\bar{t} := t_{\max} - \frac{1}{(1-\varepsilon)\varepsilon}$. Assume there is an uninitialized node w at time $\bar{t}$, i.e., $t_{\max} - \frac{1}{(1-\varepsilon)\varepsilon} = \bar{t} < t_w \leq \mathcal{T}D$. The assumption that $t_{\max} < t_w$ leads to the contradiction that

$$\hat{L}^{\max}(t_{\max}) - L_w(t_{\max}) \overset{(18)}{<} (1+\varepsilon)\mathcal{T}D + 1 < \mathcal{G},$$

hence $t_{\max} \geq t_w$. Since it holds that $L_w(t_{\max}) = (1-\varepsilon)(t_{\max} - t_w) + \mathcal{I}_w(t_w, t_{\max})$ by definition, we get that

$$\begin{aligned}
\hat{L}^{\max}(t_{\max}) - L_w(t_{\max}) &\overset{(18)}{<} (1+\varepsilon)t_w + 2\varepsilon(t_{\max} - t_w) + 1 - \mathcal{I}_w(t_w,t) \\
&< (1+\varepsilon)\mathcal{T}D + \frac{2\varepsilon}{(1-\varepsilon)\varepsilon} + 1 - \mathcal{I}_w(t_w,t) \\
&\overset{(5)}{<} (1+\varepsilon)\mathcal{T}D + \frac{2}{1-\varepsilon} + 2 < \mathcal{G}.
\end{aligned}$$

Hence, at time $\bar{t}$ all nodes are initialized.

Let t_r denote the time when w receives the largest estimate for $\hat{L}^{\max}$ until time $\bar{t}$. Consider the largest estimate $L_{\max}$ sent not later than at time $t_r - \mathcal{T}D$ for the first time by any node. If t_s denotes

this time, it holds that $L_{\max} = \hat{L}^{\max}(t_s)$. At the latest at time $t_s + \mathcal{T}D \leq t_r$, w receives this (or a larger) estimate unless the estimate for $\hat{L}^{\max}$ of another node u on a path to w reaches the same multiple of ∂H with a smaller additive term in the range $[0, 1)$ and sends this value (Statement (ii) of Lemma A.1). In this case, u will refrain from forwarding $L_{\max}$ as it has already sent its message for this particular multiple of ∂H. However, w receives at least $\lfloor L_{\max} \rfloor$ by time $t_s + \mathcal{T}D$. If w does not adopt this estimate, then the condition in Line 2 of Algorithm 2 must be false at time $t_s + \mathcal{T}D$, i.e., $\lfloor L_{\max} \rfloor < L_v + 1 + \Lambda_v^{\max}$. Thus, we conclude that $L_w(t_s + \mathcal{T}D) + \Lambda_w^{\max}(t_s + \mathcal{T}D) \geq \lfloor \hat{L}^{\max}(t_s) \rfloor$. Morover, according to Lemma A.1, $L_v + \Lambda_v^{\max}$ increases by at least 1 at each tick. This implies

$$L_w(\bar{t}) + \Lambda_w^{\max}(\bar{t}) \geq L_w(t_r) + \Lambda_w^{\max}(t_r) + \lfloor (1-\varepsilon)(\bar{t} - t_r) \rfloor \geq \lfloor \hat{L}^{\max}(t_s) \rfloor + \lfloor (1-\varepsilon)(\bar{t} - (t_s + \mathcal{T}D)) \rfloor. \quad (23)$$

Corollary A.2 states that v_s sends at least after $\frac{\partial H}{1-\varepsilon}$ *real time* has passed since the last send event. It follows that

$$\bar{t} - (t_s + \mathcal{T}D) < \frac{\partial H}{1-\varepsilon}.$$

Let t' be the last tick at w until $t_{\max}$. We first consider the case that $\mathcal{I}_w(\bar{t}, t') \geq \Lambda_w^{\max}(\bar{t})$. We estimate

$$\begin{aligned} L_v(t_{\max}) + \Lambda_v^{\max}(t_{\max}) - L_w(t_{\max}) &\overset{(18)}{<} \hat{L}^{\max}(t_s) + \lceil (1+\varepsilon)(t_{\max} - t_s) \rceil \\ &\quad - L_w(\bar{t}) - (1-\varepsilon)(t_{\max} - \bar{t}) - \mathcal{I}_w(\bar{t}, t') - \mathcal{I}_w(t', t_{\max}) \\ &\overset{(5,23)}{<} 4 + 2\varepsilon((t_{\max} - \bar{t}) - (\bar{t} - (t_s + \mathcal{T}D))) + (1+\varepsilon)\mathcal{T}D \\ &\leq 4 + \frac{2}{1-\varepsilon} + \frac{2\varepsilon}{1-\varepsilon}\partial H + (1+\varepsilon)\mathcal{T}D < \mathcal{G}, \end{aligned}$$

a contradiction to Inequality (22).

The second case is that $\mathcal{I}_w(\bar{t}, t') < \Lambda_w^{\max}(\bar{t})$ and $\Lambda_w^u(t) < \kappa$ for all $u \in \mathcal{N}_w$ and all ticks $t \in (\bar{t}, t_{\max}]$ of w. Thus we both have

$$\Lambda_w^{\downarrow}(t) + R_w(t) = \max_{u \in \mathcal{N}_w} \{\Lambda_w^u(t)\} < \kappa$$

and

$$\begin{aligned} \Lambda_w^{\max}(t) &\geq \Lambda_w^{\max}(\bar{t}) - \sum_{\text{tick } t' \in (\bar{t}, t]} R_w(t') \\ &\overset{(15)}{\geq} \Lambda_w^{\max}(\bar{t}) - \mathcal{I}_w(\bar{t}, t) > 0. \end{aligned}$$

This means that node w is free to raise its clock by μ at each tick $t \in (\bar{t}, t_{\max}]$, as can be seen from Line 2 of Algorithm 3. Since there are at least $\lfloor (1-\varepsilon)(t_{\max} - \bar{t}) \rfloor > (1-\varepsilon)(t_{\max} - \bar{t}) - 1$ tick events in the interval $[\bar{t}, t_{\max}]$, we conclude that $\mathcal{I}_w(\bar{t}, t_{\max}) > \mu(1-\varepsilon)(t_{\max} - t) - (1+\mu)$ and bound

$$\begin{aligned} L_v(t_{\max}) + \Lambda_v^{\max}(t_{\max}) - L_w(t_{\max}) &\overset{(18)}{<} \hat{L}^{\max}(\bar{t}) + \lceil (1+\varepsilon)(t_{\max} - \bar{t}) \rceil \\ &\quad - L_w(\bar{t}) - \mathcal{I}_w(\bar{t}, t_{\max}) \\ &< \hat{L}^{\max}(\bar{t}) + (1+\varepsilon)(t_{\max} - \bar{t}) + 1 \\ &\quad - L_w(\bar{t}) - (1+\mu)(1-\varepsilon)(t_{\max} - t) + 1 + \mu \\ &\leq \mathcal{G} - \mu((1-\varepsilon)(t_{\max} - \bar{t}) - 1) + 2\varepsilon(t_{\max} - \bar{t}) + 2 \\ &= \mathcal{G} - \mu\frac{1-\varepsilon}{\varepsilon} + \frac{2}{1-\varepsilon} + 2 \overset{(9)}{<} \mathcal{G}, \end{aligned}$$

again contradicting Inequality (22).

The third and final case is that $\mathcal{I}_w(\bar{t}, t') < \Lambda_w^{\max}(\bar{t})$ and a tick $t \in (\bar{t}, t_{\max}]$ and a node $u \in \mathcal{N}_w$ exist such that $\Lambda_w^u(t) \geq \kappa$. Let t be the latest such tick and u be any neighbor for which $\Lambda_w^u(t) \geq \kappa$. Moreover, let t'_s be the time when u sent the message containing the largest clock value $L_u(t'_s)$ received by w at the latest at t, and let $t'_r \leq t$ be the time when w received it. Such a time exists, since in the $2\mathcal{T}$ time it takes at most from $t_w \leq \bar{t}$ until w receives a message from u we have $\Lambda_w^u \leq L_w$ which again is bounded by $2(1+\varepsilon)(1+\mu)\mathcal{T} < \kappa$. According to Lemma A.3, we have for the earliest tick $t' \geq t - \mathcal{T} - \frac{1}{1-\varepsilon}$ at u that

$$\begin{aligned}
\kappa \leq \Lambda_w^u(t) &\overset{(20)}{<} L_w(t) - L_u(t') + \overline{\partial H} \\
&< L_w(t) - \left(L_u(t) - (1+\varepsilon)(1+\mu)\left(\mathcal{T} - \frac{1}{1-\varepsilon}\right)\right) + \overline{\partial H} \\
&\overset{(1,16)}{<} L_w(t) - L_u(t) + \kappa - (2+\mu).
\end{aligned}$$

This implies that

$$L_w(t) - L_u(t) > 2 + \mu. \tag{24}$$

Since u has a smaller clock value than w at time t, it follows that $t < t_{\max}$ as otherwise $L_v(t_{\max}) - L_w(t_{\max})$ would not violate the claimed bound on the global skew the most. By the definition of t, it holds that $\Lambda_w^u(t') < \kappa$ for all $u \in \mathcal{N}_w$ at any tick $t' \in (t, t_{\max}]$. Thus, the observations made for the second case yield that w raises its clock by μ at any such t', i.e., $\mathcal{I}_w(t, t_{\max}) > \mu(1-\varepsilon)(t_{\max} - t) - (1+\mu)$. Since at time $t < t_{\max}$ the bound on the global skew was not violated, we get that

$$\begin{aligned}
L_v(t_{\max}) + \Lambda_v^{\max}(t_{\max}) - L_w(t_{\max}) &\overset{(18)}{<} \hat{L}^{\max}(t) + 2\varepsilon(t_{\max} - t) + 1 - L_w(t) - \mathcal{I}_w(t, t_{\max}) \\
&< L_u(t) + \mathcal{G} + 2\varepsilon(t_{\max} - t) + 1 \\
&\quad - L_w(t) - \mu(1-\varepsilon)(t_{\max} - t) + (1+\mu) \\
&\overset{(24)}{<} \mathcal{G} + (2\varepsilon - (1-\varepsilon)\mu)(t_{\max} - t) \\
&\overset{(9)}{<} \mathcal{G}.
\end{aligned}$$

Thus, all cases lead to a contradiction to Inequality (22). Hence, the bound must indeed hold, which completes the proof.

A.3 Proof of Lemma 5.5

First, we assume that all nodes in the network are initialized at time t_0. The case where some nodes are not initialized at time t_0 will be treated later. Define

$$\Xi(t) := \max_{\{\text{tick } t' \geq t \text{ at } u \in V\}} \left\{ L_v(t_0) - L_u(t_0) - d(v,u)\left(s - \frac{1}{2}\right)\kappa - \mathcal{I}_u(t_0, t') \right\}.$$

Observe that if $\Xi(t) \leq 0$ holds at any time $t \geq t_0$, we have that

$$\mathcal{I}_w(t_0, t') \geq L_v(t_0) - L_w(t_0) - d(v,w)\left(s - \frac{1}{2}\right)\kappa \geq \xi$$

for all tick events $t' \geq t$ at w. Furthermore, $\Xi(\cdot)$ is monotonically decreasing, hence showing that $\Xi(t) \leq 0$ for any $t \leq \bar{t}$ proves the lemma. We proceed by bounding $\Xi(t_0)$ and afterwards showing that $\Xi(\cdot)$ decreases at a rate of at least μ per $\frac{1}{1-\varepsilon}$ real time units as long as it remains positive.

Consider any $u \in V$. There is an integer n, such that $d(v,u) \in [C_n, C_{n-1})$, where C_{-1} is interpreted as $D+1$. The legal state conditions for C_n at time t_0 allow us to bound

$$L_v(t_0) - L_u(t_0) - d(v,u)\left(s - \frac{1}{2}\right)\kappa \leq d(v,u)(n-s+1)\kappa. \tag{25}$$

For $n < s$ the r.h.s. of this inequality is at most 0. For $n \geq s$ we can further estimate

$$\begin{aligned} d(v,u)(n-s+1)\kappa &< C_{n-1}(n-s+1)\kappa \\ &= \sigma^{s-n} C_{s-1}(n-s+1)\kappa \\ &\leq \kappa C_{s-1}, \end{aligned}$$

since $\sigma \geq 2$. Hence, we conclude that $\Xi(t_0) \leq \kappa C_{s-1}$.

Define $t_n := t_0 + \frac{n}{1-\varepsilon}$ for $n \in \mathbb{N}$ and $\bar{\mathcal{T}} := \frac{\lceil (1-\varepsilon)\mathcal{T} \rceil}{1-\varepsilon}$, i.e., $\bar{\mathcal{T}}$ is the smallest integer multiple of $\frac{1}{1-\varepsilon}$ larger than $\mathcal{T}$. We will now prove by induction that $\Xi(t_n) \leq \max\{\Xi(t_0) - (n-(1-\varepsilon)\bar{\mathcal{T}})\mu, 0\}$. Observe that the claim holds for all $n \leq (1-\varepsilon)\bar{\mathcal{T}}$ because in this case we only require that $\Xi(t_n) \leq \Xi(t_0)$, which follows from the fact that $\Xi(\cdot)$ is monotonically decreasing. Hence, it remains to show that if $n+1 > (1-\varepsilon)\bar{\mathcal{T}}$ and $0 < \Xi(t_{n'}) \leq \Xi(t_0) - (n' - (1-\varepsilon)\bar{\mathcal{T}})\mu$ for all $n' \leq n$, then we also have that $\Xi(t_{n+1}) \leq \Xi(t_0) - (n+1-(1-\varepsilon)\bar{\mathcal{T}})\mu$. Assume for the sake of contradiction that there is a node $u \in V$ such that

$$L_v(t_0) - L_u(t_0) - d(v,u)\left(s - \frac{1}{2}\right)\kappa - \mathcal{I}_u(t_0,t) > \Xi(t_0) - (n+1-(1-\varepsilon)\bar{\mathcal{T}})\mu > 0 \tag{26}$$

for some tick event $t \geq t_{n+1}$ at node u. Since the r.h.s. of Inequality (26) is positive and t is a tick at u, Inequality (14) shows that $u \neq v$. Hence, a node $u' \in \mathcal{N}_u$ with $d(v,u') = d(v,u) - 1$ exists. Let t' denote the earliest tick after $t - \mathcal{T} - \frac{1}{1-\varepsilon} \geq t_{n+1} - \bar{\mathcal{T}} - \frac{1}{1-\varepsilon} = t_{n-(1-\varepsilon)\bar{\mathcal{T}}}$ this node. We apply the induction hypothesis for $t_{n-(1-\varepsilon)\bar{\mathcal{T}}}$ to get the inequality

$$L_v(t_0) - L_{u'}(t_0) - d(v,u')\left(s - \frac{1}{2}\right)\kappa - \mathcal{I}_{u'}(t_0,t') \leq \Xi(t_0) - (n - 2(1-\varepsilon)\bar{\mathcal{T}})\mu. \tag{27}$$

By subtracting Inequality (27) from Inequality (26), we get that

$$L_{u'}(t_0) + \mathcal{I}_{u'}(t_0,t') - L_u(t_0) - \mathcal{I}_u(t_0,t) > \left(s - \frac{1}{2}\right)\kappa - \mu((1-\varepsilon)\bar{\mathcal{T}} + 1). \tag{28}$$

Since t' is the earliest tick after $t - \mathcal{T} - \frac{1}{1-\varepsilon}$, we can now use Lemma A.3 to lower bound the term $-\Lambda_u^{u'}(t)$ in order to get a bound on $\Lambda_u^{\uparrow}(t)$:

$$\begin{aligned} -\Lambda_u^{u'}(t) &\overset{(20)}{>} L_{u'}(t') - L_u(t) - \overline{\partial H} \\ &= L_{u'}(t_0) + \mathcal{I}_{u'}(t_0,t') - L_u(t_0) - \mathcal{I}_u(t_0,t) - (1-\varepsilon)(t-t') - \overline{\partial H} \\ &\overset{(28)}{>} \left(s - \frac{1}{2}\right)\kappa - \mu((1-\varepsilon)\bar{\mathcal{T}} + 1) - \overline{\partial H} - (1-\varepsilon)\bar{\mathcal{T}} \\ &\overset{(16)}{>} (s-1)\kappa. \end{aligned}$$

Note that a clock raise at time t does not change the value of $\Lambda_u^{\uparrow}$, thus $-\Lambda_u^{u'}(t) > (s-1)\kappa$ implies that

$$\Lambda_u^{\uparrow}(t) - R_u(t) > (s-1)\kappa \geq 0. \tag{29}$$

Moreover, due to Statement (*iii*) of Lemma A.1, this implies that $\Lambda_u^{\max}(t) > 0$.

For an arbitrary neighbor $u'' \in \mathcal{N}_u$, similarly to Inequality (27) it holds that

$$L_u(t_0) + \mathcal{I}_u(t_0,t) - L_{u''}(t_0) - \mathcal{I}_{u''}(t_0,t') < \left(s - \frac{1}{2}\right)\kappa + \mu((1-\varepsilon)\bar{\mathcal{T}} + 1) \tag{30}$$

where t' is defined analogously. The only term that does not switch its sign compared to Inequality (28) is $\left(s-\frac{1}{2}\right)\kappa$, since now $d(v,u'')$ might be $d(v,u)+1$. We estimate $\Lambda_u^{u''}(t)$ to obtain a bound on $\Lambda_u^{\downarrow}(t)$:

$$\begin{aligned}
\Lambda_u^{u''}(t) &< L_u(t_0) + \mathcal{I}_u(t_0,t) - L_{u''}(t_0) - \mathcal{I}_{u''}(t_0,t') + (1-\varepsilon)(t-t') + \overline{\partial H} \\
&\overset{(30)}{<} \left(s - \frac{1}{2}\right)\kappa + \mu((1-\varepsilon)\bar{\mathcal{T}} + 1) + (1-\varepsilon)\bar{\mathcal{T}} + \overline{\partial H} \\
&\overset{(16)}{<} s\kappa.
\end{aligned}$$

Thus, since this estimate holds for any neighbor of v_i, we conclude that

$$\Lambda_u^{\downarrow}(t) + R_u(t) < s\kappa. \tag{31}$$

Putting things together, we can follow from the fact that the value of $\Lambda_u^{\max}$ is reduced by $R_u(t)$ *after* the computation of $R_u(t)$ in Line 2 of Algorithm 3, that $R_u(t)$ must attain either μ or the result of the previous Line 1. The latter, however, cannot be the case, as Inequalities (29) and (31) together imply that

$$\left\lfloor \frac{\Lambda_u^{\uparrow} - R}{\kappa} \right\rfloor \geq \left\lfloor \frac{\Lambda_u^{\downarrow} + R}{\kappa} \right\rfloor$$

for some $R > R_u(t)$. Hence, u computes $R_u(t) = \mu$ at time t.

Finally, we use the assumption $\Xi(t_n) \geq \Xi(t_0) - (n - (1-\varepsilon)\bar{\mathcal{T}})\mu$, which yields for the latest tick event $t' \in [t_n, t)$ at u prior to t that

$$L_v(t_0) - L_u(t_0) - d(v,u)\left(s - \frac{1}{2}\right)\kappa - \mathcal{I}_u(t_0,t') \leq \Xi(t_0) - (n - (1-\varepsilon)\bar{\mathcal{T}})\mu$$

and thus $\mathcal{I}_u(t',t) < \mu$ due to Inequality (26). This contradicts the observation that u raises its clock by μ at time t. Hence, as long as $\Xi(t_{n+1}) > 0$, we have that $\Xi(t_{n+1}) \leq \Xi(t_0) - (n+1-(1-\varepsilon)\bar{\mathcal{T}})\mu$. This implies that $\Xi(\bar{t}\,) \leq \max\{0, \Xi(t_0) - (1-\varepsilon)\mu(\bar{t} - t_0 - \mathcal{T} - \frac{1}{1-\varepsilon})\} = 0$ as desired, since $\mathcal{T} + \frac{1}{1-\varepsilon} > \bar{\mathcal{T}}$.

In order to complete the proof, we have to cope with the case of uninitialized nodes. However, we will show that we only need to consider node $u \in V$ within distance $d(v,u) < C_{s-1}$ from v. We already observed for $d(v,u) \geq C_{s-1}$ that

$$L_v(t_0) - L_u(t_0) - d(v,u)\left(s - \frac{1}{2}\right)\kappa \overset{(25)}{\leq} 0. \tag{32}$$

Suppose at time t_0 all nodes $u \in V$ at distance $d(v,u) \leq \lceil C_{s-1} \rceil = C_{s-1}$ from v are initialized. In this case, for any such node u' at distance exactly C_{s-1} from v it holds that $L_v(t_0) - L_{u'}(t_0) -$

$d(v,u')\left(s-\frac{1}{2}\right)\kappa - \mathcal{I}_{u'}(t_0,t') \leq 0$ for all tick events $t' \geq t_0$ at node u' due to Inequality (32). Thus, replacing Inequality (27) by this estimate when bounding $\Lambda_u^{u'}$ for a neighbor $u \in \mathcal{N}_{u'}$ at distance $d(v,u) < C_{s-1}$ from v, we see that messages from u' will not prevent u from raising its clock far enough. Moreover, u is not required to raise its clock until $t_0 + \bar{\mathcal{T}} > t_0 + \mathcal{T}$ to satisfy the bound; at this time u will have received at least one message from u'. Thus, this case is already covered by the preceding arguments.

Hence, we assume that at time t_0 an uninitialized node $v_0 \in V$ at distance $d(v,v_0) \leq C_{s-1}$ exists. Consequently, v has been initialized at a time $t_v \geq t_0 - d(v,v_0)\mathcal{T}$. Thus, at time $t_0' := t_v + C_{s-1}\mathcal{T} \in (t_0, t_0 + C_{s-1}\mathcal{T}]$ all nodes $u \in V$ at distance $d(v,u) \leq C_{s-1}$ are initialized. We define

$$\Xi'(t) := \max_{\{\text{tick } t' \geq t \text{ at } u \in V \mid d(v,u) < C_{s-1}\}} \left\{ L_v(t_0) - L_u(t_0) - d(v,u)\left(s-\frac{1}{2}\right)\kappa - \mathcal{I}_u(t_0,t') \right\}$$

analogously to Ξ. The trivial estimates $L_u(t_0') \geq (1-\varepsilon)(t_0' - t_u) - 1 > (1-\varepsilon)(t_0' - t_0 - d(v,u)\mathcal{T}) - 1$ and $L_v(t_0) \leq (1+\mu)(1+\varepsilon)(t_0' - t_v)$ allow us to estimate

$$\begin{aligned}
\Xi'(t_0') &\leq \max_{\{u \in V \mid d(v,u) < C_{s-1}\}} \left\{ L_v(t_0) - L_u(t_0) - d(v,u)\left(s-\frac{1}{2}\right)\kappa - \mathcal{I}_u(t_0,t_0') \right\} \\
&\leq \max_{\{u \in V \mid d(v,u) < C_{s-1}\}} \left\{ L_v(t_0) - L_u(t_0') + (1-\varepsilon)(t_0' - t_0) - d(v,u)\frac{\kappa}{2} \right\} \\
&< \max_{\{u \in V \mid d(v,u) < C_{s-1}\}} \left\{ (1+\varepsilon)(1+\mu)(t_0 - t_v) + 1 - d(v_0,v)\left(\frac{\kappa}{2} - (1-\varepsilon)\mathcal{T}\right) \right\} \\
&\overset{(16)}{<} (1+\varepsilon)(1+\mu)((t_0' - t_v) - (t_0' - t_0)) \\
&< (1+\varepsilon)(1+\mu)\mathcal{T}C_{s-1} - \mu(1-\varepsilon)(t_0' - t_0) \\
&\overset{(16)}{<} \kappa C_{s-1} - \mu(1-\varepsilon)(t_0' - t_0).
\end{aligned}$$

At time $t_0' + \mathcal{T}$, any node at distance $d(v,u) < C_{s-1}$ will have received at least one message from each of its neighbors. Hence we can repeat the induction used on Ξ for Ξ', however, starting at time t_0' instead of t_0, as nodes are not required to raise their clocks until $t_0' + \mathcal{T}$ to achieve the bound $\Xi'(\bar{t}) \leq 0$. Since we must have $d(v,w) < C_{s-1}$ due to Inequality (32), again it follows that at all ticks $t \geq \bar{t} > t_0'$ at w we have $\mathcal{I}_w(t_0,t) \geq \xi$, which completes the proof.

A.4 Proof of Lemma 5.7

Note that if v is not initialized at time t_0, then $L_w(t_0) \leq (1+\varepsilon)(1+\mu)\mathcal{T}d(v,w) < \kappa\, d(v,w)$, implying that $\Psi_v^s(t_0) < 0$. We can thus assume that v is initialized at time t_0.

Suppose for the sake of contradiction that the statement is false for a node $w \in V$ and a time $t > t_0$, i.e.,

$$L_w(t) - L_v(t) - s\kappa d(v,w) - \frac{3}{2}\kappa C_{s+1} \geq \Psi_v^s(t_0) + 2\varepsilon(t - t_0) - \mathcal{I}_v(t_0,t) + \frac{\kappa}{4} - 2 - \mu. \tag{33}$$

Certainly we have that $w \neq v$, since the right hand side of the inequality is positive. Note that the inequality can only start to hold when a clock value changes, as $2\varepsilon(t-t_0) - \mathcal{I}_v(t_0,t)$ is monotonically increasing between ticks at v. Thus, w.l.o.g. we may assume that t is the first point in time when a violation occurs. Moreover, any change in the clock value of v also appears on the l.h.s. of the inequality, as $L_v(t)$ is subtracted. Hence, t is a tick event at w.

We show that w will not raise its clock at time t, i.e., $R_w(t) = 0$. Since $v \neq w$, a neighbor $u \in \mathcal{N}_w$ at distance $d(v,u) = d(v,w) - 1$ from v exists. If w has not yet received a message from u, w has been initialized less than $2\mathcal{T}$ time ago. In this case, it cannot have a clock value larger than $2(1+\varepsilon)(1+\mu)\mathcal{T} < \kappa \leq s\kappa$ and will certainly not violate the bound. Hence, we can assume that w already received at least one message from u in the following, and t is not the initialization time of w.

Let $t_s < t$ denote the time when u sent the largest clock value w received at a time $t_r \leq t$. We need to distinguish between the following two cases.

If $t_s \geq t_0$, Inequality (8) held at t_s, which allows us to bound

$$\begin{aligned}
\Lambda_w^u(t) &\overset{(15)}{\geq} L_w(t_r) - L_u(t_s) + \mathcal{I}_w(t_r,t) - 2\varepsilon(t-t_r) \\
&= L_w(t) - L_v(t) - (L_u(t_s) - L_v(t_s)) + \mathcal{I}_v(t_s,t) + (1-\varepsilon)(t_r - t_s) - 2\varepsilon(t-t_r) \qquad (34)\\
&> s\kappa(d(v,w) - d(v,u)) - \mathcal{I}_v(t_0,t) + \mathcal{I}_v(t_0,t_s) + \mathcal{I}_v(t_s,t) \\
&= s\kappa. \qquad (35)
\end{aligned}$$

If $t_s < t_0$, the time difference $t_0 - t_s$ is bounded as nodes send messages at least every ∂H ticks, i.e., after at most $\frac{\partial H}{1-\varepsilon}$ time, which may be delayed by $\mathcal{T}$.

$$t_0 - t_s \leq t - t_s < \frac{\partial H}{1-\varepsilon} + \mathcal{T}. \qquad (36)$$

Furthermore,

$$\begin{aligned}
L_u(t_s) - L_v(t_s) &< \Psi_v^s(t_0) + s\kappa d(v,u) - \mathcal{I}_u(t_s,t_0) + \mathcal{I}_v(t_s,t_0) \\
&\overset{(5)}{<} \Psi_v^s(t_0) + s\kappa d(v,u) + 1 + \mathcal{I}_v(t_s,t_0).
\end{aligned}$$

Inserting this bound and Inequality (33) into Inequality (34) yields that

$$\begin{aligned}
\Lambda_w^u(t) &> s\kappa(d(v,w) - d(v,u)) + \frac{\kappa}{4} - 3 - \mu - 2\varepsilon(t_0 - t_s) \\
&\overset{(36)}{>} s\kappa + \frac{\kappa}{4} - \left(3 + \mu + 2\varepsilon\left(\frac{\partial H}{1-\varepsilon} + \mathcal{T}\right)\right) \\
&\overset{(1,9)}{>} s\kappa + \frac{\kappa}{4} - \frac{\mathcal{T} + 5 + \mu + \mu\partial H + \mu\mathcal{T}}{2} \\
&\overset{(16,17)}{>} s\kappa.
\end{aligned}$$

Thus, Inequality (35) holds independently of whether $t_s \geq t_0$ or not, which implies that

$$\left\lfloor \frac{\Lambda_w^{\downarrow}(t) + R_w(t)}{\kappa} \right\rfloor \geq \left\lfloor \frac{\Lambda_w^u(t)}{\kappa} \right\rfloor \geq s. \qquad (37)$$

In addition, any neighbor $u \in \mathcal{N}_w$ of w is within distance $d(v,u) \leq d(v,w) + 1$ from v. In case w received a message from u until t, analogously we get that

$$-\Lambda_w^u(t) < s\kappa,$$

as all terms but $s\kappa$ in the previous estimates switch signs, because $d(v,w)-d(v,u)$ may also become -1 now.

If w did not receive a message from u until t, we have that $-\Lambda_w^u(t) \leq 0 < s\kappa$. Thus, as the bound is independent of the choice of the neighbor u, we get that

$$\left\lfloor \frac{\Lambda_w^{\uparrow}(t) - R_w(t)}{\kappa} \right\rfloor = \left\lfloor \frac{\max_{u\in\mathcal{N}_w}\{-\Lambda_w^u(t)\}}{\kappa} \right\rfloor \leq s-1.$$

Note that this estimate and Inequality (37) also hold for some value $R < R_w(t)$, as Λ_w^u satisfies strict inequalities in both cases. Consequently, it must hold that R_w evaluates to a smaller value than $R_w(t)$ in Line 1 of Algorithm 3 at time t. Inequality (35) further entails that $\kappa-\Lambda_w^{\downarrow}(t) < (1-s)\kappa \leq 0$, thus it follows from Line 2 of Algorithm 3 that $R_w(t) = 0$. Hence, w increases its clock only by 1 at time t as claimed.

Let $t' \geq t - \frac{1}{1+\varepsilon}$ denote the latest tick event before t at node w. If $t' \geq t_0$, Inequality (8) held for node w, implying that

$$\begin{aligned} L_w(t) - L_v(t) &= L_w(t') - L_v(t') + \mathcal{I}_w(t',t) - \mathcal{I}_v(t',t) \\ &\overset{(15)}{<} L_w(t') - L_v(t') + 2\varepsilon(t-t') - \mathcal{I}_v(t',t) \\ &\overset{(8)}{\leq} \Psi_v^s(t_0) + s\kappa d(v,w) + 2\varepsilon(t-t_0) - \mathcal{I}_v(t_0,t) + \frac{\kappa}{4} - 2 - \mu, \end{aligned}$$

contradicting the assumption that the bound is violated at time t. However, if $t' < t_0$, we can use the trivial estimate $L_w(t) - L_w(t_0) = L_w(t) - L_w(t') \leq 1+\mu$ to show that in case of $t' < t_0$ it must hold that

$$\Psi_v^s(t_0) = \Psi_v^s(t) + \mathcal{I}_v(t_0,t) - \mathcal{I}_w(t_0,t) \overset{(33)}{>} \Psi_v^s(t_0) + 2\varepsilon(t-t_0) + \frac{\kappa}{4} - 3 - 2\mu > \Psi_v^s(t_0),$$

as $\frac{\kappa}{2} \overset{(16)}{>} (1+\mu)(\mathcal{T}+1) + 4 + \mu + \overline{\partial H} \overset{(1,17)}{>} 6 + 4\mu$. Since all cases lead to a contradiction, indeed Inequality (8) can never be violated.

Tight Lower Bounds for Clock Synchronization

Christoph Lenzen, Thomas Locher, Roger Wattenhofer
{lenzen, lochert, wattenhofer}@tik.ee.ethz.ch
Computer Engineering and Networks Laboratory (TIK)
ETH Zurich, 8092 Zurich, Switzerland

TIK Report number 299

May 15, 2009

Abstract

Clock synchronization is a fundamental and well-studied problem in distributed computing. In 2004, however, Fan and Lynch opened an intriguing and challenging chapter of this topic by raising the question what bounds can be achieved on the skew between clocks of nodes that are close to each other in a large network. Surprisingly, they proved that the skew between neighbors must grow as $\Omega(\log D/\log\log D)$, where D is the diameter of the network. We improve this result in several ways: First, we remove the $\log\log D$ factor and thereby close the gap between the upper bound of $\mathcal{O}(\log D)$ and the lower bound. Second, we prove that the base of the logarithm depends inversely on the relative clock drift. Third, we show that the base of the logarithm becomes a constant if we severely restrict the clock rates in order to ensure that the clocks run smoothly at all times. These results are accompanied by a stronger lower bound on the clock skew between distant nodes if the nodes are required to keep their clock values as close to real time as possible. The respective upper bounds reveal that all our presented results are optimal up to small constants.

1 Introduction & Related Work

Lower bounds are the bee's knees of distributed computing. While lower bounds for the non-distributed random access machine (RAM) often need to constrain potential algorithms in unnatural ways,[1] researchers in distributed computing can argue about lower bounds quite generally by exploiting the limitations imposed by *local knowledge* [8]. Such *indistinguishability* arguments come in different flavors, but they essentially all boil down to the same concepts. Hence they are well understood, and often not overly surprising to experts in the field.

Five years ago, in a seminal paper, Fan and Lynch [3] did surprise the experts by presenting a non-trivial and unexpected lower bound in clock synchronization. They proved that direct neighbors cannot synchronize their clocks arbitrarily well, instead the quality of synchronization depends on the diameter of the network! Their result holds in a seemingly practical setting, where nodes have quite accurate hardware clocks, and nodes are able to communicate with their neighbors by exchanging messages. These messages are sometimes a bit faster, sometimes a bit

[1] For example, lower bound arguments for sorting are often restricted to algorithms that can only compare and exchange elements.

slower, but the message exchange is reliable, i.e., messages are never lost, and the message delay is bounded.[2] More specifically, Fan and Lynch showed that the clock skew between two neighbors can be $\Omega(\log D / \log\log D)$, where D denotes the diameter of the network, if all the other parameters (clock drift, message jitter) are constant. This result is astonishing because it seems reasonable at first glance that the accuracy of information on other clock values should determine the quality of synchronization.

It was later shown that a large class of natural algorithms exhibits an exponentially worse clock skew between neighboring nodes in the worst case [6],[3] and only last year a rather intricate algorithm was proposed that almost matches the lower bound discovered by Fan and Lynch, guaranteeing an upper bound of $O(\log D)$ on the clock skew between neighbors [4].

Since clock synchronization is arguably an important cornerstone in distributed computing (see, e.g., [1, 2, 7, 10, 11, 12] and references therein), one wants to close the remaining doubly logarithmic gap. This is exactly the first result of this paper. By improving the Fan/Lynch lower bound to $\Omega(\log D)$, we get a tight bound on the worst-case clock skew between neighboring nodes. In addition, we try to get an understanding of how other parameters, in particular the maximum possible clock drift, affect the lower bound. As we will show, the base of the logarithm depends inversely on the relative clock drift. Since in real-world systems the inverse of the clock drift is generally larger than the diameter of the network, the lower bound becomes rather tame. Additionally, we show that the base of the logarithm gradually shifts to a constant if the clocks are required to run more and more smoothly, i.e., if the clock rates must not change abruptly in a short time. As proved in [5], these bounds are optimal up to a factor of two. As one might expect, imposing such tight, but quite natural restrictions on the behavior of the clocks also leads to a larger worst-case clock skew between *any* two nodes in the network. We prove that no algorithm can avoid a clock skew of approximately D under these conditions; this lower bound is roughly a factor of 2 stronger than the bound of $D/2$ if the behavior of the clocks is not restricted [2]. The algorithm given in [5] matches our stronger bound up to a small additive term.

2 Model

We model a distributed system as an arbitrary connected graph $G = (V, E)$ of diameter D, where nodes represent computational devices and edges represent communication links. The message message delays are bounded, but they may vary: The time that passes until a recipient can act upon the sent information may be any value in the range $[0, \mathcal{T}]$. Our bounds neither rely on the size or frequency of messages, nor on the amount of local computations that can be performed.

Each node v is equipped with a *hardware clock* H_v whose value at *real time* t is denoted by $H_v(t)$, i.e., $H_v : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ is a strictly monotonically increasing function. For simplicity, all nodes start their clocks synchronously at time $t = 0$. The value of the hardware clock of v is $H_v(t) := \int_0^t h_v(\tau)\, d\tau$ afterwards, where $h_v(\tau)$ is the *hardware clock rate* of v at time τ. The clock rates can vary over time, but there is a constant $0 < \varepsilon < 1$ such that the following condition holds.

$$\forall v \in V\ \forall t \geq t_v : 1 - \varepsilon \leq h_v(t) \leq 1 + \varepsilon.$$

[2] We will elaborate on the exact model later, in Section 2. Indeed, the same techniques also work in slightly different models: As shown in [9] the lower bound also holds if all messages arrive instantaneously, but an adversary can determine when synchronization messages may be sent.

[3] Indeed, the worst-case clock skew between neighboring nodes of all previously known algorithms was exponentially worse than the lower bound as originally the only goal was to minimize the clock skew between *any* two nodes.

Based on its hardware clock and the received information, each node v computes a *logical clock* L_v, which is also a function $L_v : \mathbb{R}_0^+ \to \mathbb{R}_0^+$. Since L_v is expected to emulate a clock, we require that it guarantees a constant minimum progress rate of $\alpha \in \Omega(1)$. Moreover, it might be desirable that clocks run smoothly, i.e., the clock rates are upper bounded by some value $\beta > \alpha$. Formally, there are constants $\alpha, \beta > 0$ such that

$$\forall v \in V \; \forall t < t' : \; \alpha(t' - t) \leq L_v(t') - L_v(t) \leq \beta(t' - t). \tag{1}$$

We allow that $\beta = \infty$, which simply means that an algorithm may not guarantee any upper bound on the clock rate.

Since a computational device typically synchronizes its operations internally based on a *clock pulse*, we will mostly assume that a node $v \in V$ performs computations and sends and receives messages only at such clock pulses. Each node v has a clock pulse at time t if $H_v(t)$ is an integer value. We call such a time a *tick event at node* v or simply a *tick* at v. Naturally, in this case Condition (1) is relaxed in the sense that it only needs to hold at ticks t and t' (and not necessarily between tick events). We will refer to such an algorithm as a *discrete* clock synchronization algorithm, as opposed to a *continuous* clock synchronization algorithm which may act at any time. Note that if message delays are (slightly) adjusted such that messages are received only at ticks, the requirement that nodes act only at discrete times is no restriction because no additional information becomes available in between. Thus, the obtained lower bounds are indeed stronger and drectly carry over to the continuous model.

Given an algorithm $\mathcal{A}$, an *execution* specifies the delays of all messages and also the hardware clock rates of all nodes at each point in time when $\mathcal{A}$ is executed on a given graph G. The *global* and the *local skew* are defined as follows:

Definition 2.1 (Global Skew) *Given a connected graph $G = (V, E)$ and a clock synchronization algorithm $\mathcal{A}$, the* global skew *is defined as the value*

$$\sup_{\mathcal{E}, v \in V, w \in V, t} \{L_v(t) - L_w(t)\},$$

where $\mathcal{E}$ is any execution of $\mathcal{A}$ on G.

Definition 2.2 (Local Skew) *Given a connected graph $G = (V, E)$ and a clock synchronization algorithm $\mathcal{A}$, the* local skew *is defined as the value*

$$\sup_{\mathcal{E}, v \in V, w \in \mathcal{N}_v, t} \{L_v(t) - L_w(t)\},$$

where $\mathcal{E}$ is any execution of $\mathcal{A}$ on G.

All our bounds are proved using indistinguishability type of arguments. Concretely, we construct different executions for any given synchronization algorithm $\mathcal{A}$ and any graph G so that at least one of the executions causes large clock skews. Given two executions $\mathcal{E}$ and $\bar{\mathcal{E}}$ of an algorithm $\mathcal{A}$ on a graph G, let $H_v^{\mathcal{E}}(t)$ and $H_v^{\bar{\mathcal{E}}}(t)$ denote the hardware clock values of v at time t in $\mathcal{E}$ and $\bar{\mathcal{E}}$, respectively. The corresponding logical clock values are denoted by $L_v^{\mathcal{E}}(t)$ and $L_v^{\bar{\mathcal{E}}}(t)$. The following definition formalizes the notion of indistinguishable executions.

Definition 2.3 (Indistinguishability of Executions) *We call $\mathcal{E}$ and $\bar{\mathcal{E}}$* indistinguishable at node v until hardware clock time H, *if v observes the same message pattern with respect to its*

local time H_v in both $\mathcal{E}$ and $\bar{\mathcal{E}}$ until its hardware clock reaches H. More precisely, if v receives a message at a time t_r when $H_v^{\mathcal{E}}(t_r) \leq H$ in $\mathcal{E}$, it receives an identical message in $\bar{\mathcal{E}}$ at the time $\bar{t}_r$ where $H_v^{\bar{\mathcal{E}}}(\bar{t}_r) = H_v^{\mathcal{E}}(t_r)$, and vice versa. Note that in this situation v acts the same in $\mathcal{E}$ and $\bar{\mathcal{E}}$ until local time H, i.e., if $H_v^{\mathcal{E}}(t) = H_v^{\bar{\mathcal{E}}}(\bar{t}) \leq H$, it follows that $L_v^{\mathcal{E}}(t) = L_v^{\bar{\mathcal{E}}}(t)$ and v sends the same set of messages at times t and $\bar{t}$ in $\mathcal{E}$ and $\bar{\mathcal{E}}$, respectively.

If it is clear from the context, we may omit the specification of the execution in the notation and write, e.g., $H_v(t)$ instead of $H_v^{\mathcal{E}}(t)$.

The proofs of the bounds on the local skew use incrementally constructed executions that enforce large clock skews. This concept is captured by the following definition.

Definition 2.4 (Extended Executions) *Given an execution $\mathcal{E}$ running until time t, we can extend it by specifying hardware clock rates and message delays in a time interval $[t, t']$. We will refer to this extension as an execution $\mathcal{E}'$ running from time t until time t'. Note that $\mathcal{E}'$ inherits the state of the system at time t, i.e., the state of all nodes and any "pending" messages sent in $\mathcal{E}$ which did not reach their destination until time t.*

3 Lower Bounds

In order to establish our bounds on the local skew, we need some technical statements. The common theme of all arguments is that it is possible to indistinguishably introduce up to $\mathcal{O}(d(v,w)\mathcal{T})$ hardware clock skew between two nodes v and w in $\mathcal{O}\left(\varepsilon^{-1}d(v,w)\mathcal{T}\right)$ time. Since clocks are required to run at least at the rate α, this observation enables us to gradually build up skew on paths of decreasing length.

3.1 Helper Statements

The first stated lemma illustrates the reason why it must take a certain time to reduce clock skews even if nodes are aware of a certain skew. It shows that one can modify any member of the following (quite general) class of executions such that the hardware clock value of a specific node is imperceptibly changed by $\Theta(\mathcal{T})$.

Definition 3.1 (Framed Executions) *Given $\varphi \in \left[0, \frac{1}{2}\right]$, a* φ-framed *execution is any execution such that the hardware clock rates always lie in the interval $[1, 1+\varepsilon]$ and all message delays are in the range $[\varphi\mathcal{T}, (1-\varphi)\mathcal{T}]$.*

Lemma 3.2 *Fix any clock synchronization algorithm and any graph. Let a φ-framed execution $\mathcal{E}$, a time $t \geq \varphi\mathcal{T}/\varepsilon$, and a node $v \in V$ be given, and define that $t' := t - \varphi\mathcal{T}/(1+\varepsilon)$. We can indistinguishably modify $\mathcal{E}$ into an execution $\bar{\mathcal{E}}$ for which it holds that $L_v^{\bar{\mathcal{E}}}(t) = L_v^{\mathcal{E}}(t')$ and $L_w^{\bar{\mathcal{E}}}(t) = L_w^{\mathcal{E}}(t)$ for any node $w \in V \setminus \{v\}$.*

Proof. We change $\mathcal{E}$ to $\bar{\mathcal{E}}$ by reducing the hardware clock rate of node v by ε in the time interval $\left[0, \left(H_v^{\mathcal{E}}(t) - H_v^{\mathcal{E}}(t')\right)/\varepsilon\right]$ and by modifying all delays in such a way that indistinguishability is maintained. Certainly, we have that $H_v^{\mathcal{E}}(t) - H_v^{\mathcal{E}}(t') \leq (1+\varepsilon)(t'-t) = \varphi\mathcal{T}$, implying that $t \geq \left(H_v^{\mathcal{E}}(t) - H_v^{\mathcal{E}}(t')\right)/\varepsilon$. Therefore it holds that $H_v^{\bar{\mathcal{E}}}(t) = H_v^{\mathcal{E}}(t')$. Apparently, we also have that $H_w^{\bar{\mathcal{E}}}(t) = H_w^{\mathcal{E}}(t)$ for any other node $w \neq v$. The indistinguishability of $\mathcal{E}$ and $\bar{\mathcal{E}}$ then implies the corresponding statement on the logical clock values.

It remains to show that $\bar{\mathcal{E}}$ is a valid execution. By construction we have that

$$H_v^{\mathcal{E}}(t'' - \varphi\mathcal{T}) \leq H_v^{\mathcal{E}}(t'') - \varphi\mathcal{T} \leq H_v^{\bar{\mathcal{E}}}(t'') \leq H_v^{\mathcal{E}}(t'')$$

at any time t'', implying that delays change by at most $\varphi\mathcal{T}$. Since $\mathcal{E}$ is a φ-framed execution, all delays in $\bar{\mathcal{E}}$ thus lie within the legitimate range of $[0, \mathcal{T}]$. Similarly, all hardware clock rates in $\mathcal{E}$ are at least 1, meaning that in $\bar{\mathcal{E}}$ all clock rates are in the interval $[1-\varepsilon, 1+\varepsilon]$ as required. □

The lower bound of Fan and Lynch exploits that if any node increases its clock at an (average) rate of at least b over $\Omega(\mathcal{T})$ time in a framed execution, this lemma can be used to construct an execution resulting in a clock skew of $\Omega(\mathcal{T}b)$ between two neighbors. Furthermore, an extended execution can be constructed where the clock skew between two neighbors becomes $\Omega(\mathcal{T}\log_b D)$. If these two results are combined, we get a lower bound of $\Omega(\mathcal{T}b+\mathcal{T}\log_b D) \subset \Omega(\mathcal{T}(\log D/\log\log D))$. In order to get rid of the $\log\log D$ term, we need more sophisticated techniques.

Next, we extend the scope to nodes at arbitrary distances, but in return confine the considered executions.

Lemma 3.3 *Fix any clock synchronization algorithm, any graph, an arbitrary pair of nodes $v, w \in V$ and some $\varphi \in [0, 1/(2(1+\varepsilon))]$. Given a φ-framed execution $\mathcal{E}_0$ that ends at a time $t_{\mathcal{E}_0}$, this execution can be extended by a φ-framed execution $\mathcal{E} = \mathcal{E}(\mathcal{E}_0, v, w, \varphi)$ with the following property. For any pair of nodes $v', w' \in V$ on a shortest path from v to w such that $d(v,v') < d(v,w')$ and for any time $t_{\mathcal{E}} \geq t_{\mathcal{E}_0} + (1+\varepsilon)(1-2(1+\varepsilon)\varphi)d(v',w')\mathcal{T}/\varepsilon$, $\mathcal{E}$ can be modified into the φ-framed execution $\bar{\mathcal{E}} = \bar{\mathcal{E}}(\mathcal{E}, v', w')$ such that at time $t_{\bar{\mathcal{E}}} := t_{\mathcal{E}} - (1-2(1+\varepsilon)\varphi)d(v',w')\mathcal{T}$ we have that $L_{v'}^{\bar{\mathcal{E}}}(t_{\bar{\mathcal{E}}}) = L_{v'}^{\mathcal{E}}(t_{\mathcal{E}})$ and $L_{w'}^{\bar{\mathcal{E}}}(t_{\bar{\mathcal{E}}}) = L_{w'}^{\mathcal{E}}(t_{\bar{\mathcal{E}}})$.*

Proof. Define $\Phi_v^w(u) := d(w,u) - d(v,u)$. In execution $\mathcal{E}$, there are no clock drifts, i.e., all hardware clock rates are always 1, and message delays from node $u_s \in V$ to $u_r \in \mathcal{N}_{u_s}$ are $(1+\varepsilon)\varphi\mathcal{T}$ if $\Phi_v^w(u_s) \geq \Phi_v^w(u_r)$ and $(1-(1+\varepsilon)\varphi)\mathcal{T}$ otherwise. If possible, the delays of any messages sent in $\mathcal{E}_0$ that have not yet arrived are the same, whereas messages already delayed by more are received immediately at time $t_{\mathcal{E}_0}$.

Set $t' := t_{\bar{\mathcal{E}}} - (1-2(1+\varepsilon)\varphi)d(v',w')\mathcal{T}/\varepsilon \geq t_{\mathcal{E}_0}$. Execution $\bar{\mathcal{E}}$ is defined as follows: In execution $\bar{\mathcal{E}}$, the hardware clock rate of any node $u \in V$ is

$$h_u(t) := \begin{cases} \max\left\{\min\left\{1+\varepsilon - \frac{\Phi_v^w(v') - \Phi_v^w(u)}{2d(v',w')}\varepsilon, 1+\varepsilon\right\}, 1\right\} & \text{if } t \in [t', t_{\bar{\mathcal{E}}}] \\ 1 & \text{else} \end{cases}$$

Due to the prerequisites that $d(v,v') < d(v,w')$ and v' and w' lie on a shortest path from v to w, we have that $\Phi_v^w(v') - \Phi_v^w(w') = 2d(v',w')$. Therefore, w' has a clock rate of 1 at any time, i.e., $H_{w'}^{\bar{\mathcal{E}}}(t_{\bar{\mathcal{E}}}) = H_{w'}^{\mathcal{E}}(t_{\bar{\mathcal{E}}})$. As v' has clock rate $1+\varepsilon$ for exactly $t_{\bar{\mathcal{E}}} - t' = (1-2(1+\varepsilon)\varphi)d(v',w')\mathcal{T}/\varepsilon$ time in $\bar{\mathcal{E}}$, we have that $H_{v'}^{\bar{\mathcal{E}}}(t_{\bar{\mathcal{E}}}) = H_{v'}^{\mathcal{E}}(t_{\mathcal{E}})$. The message delays are adjusted in such a way that $\mathcal{E}$ and $\bar{\mathcal{E}}$ are indistinguishable at any node $u \in V$. Hence, as both executions inherit the same state of the network from the preceding execution $\mathcal{E}_0$, the statements $L_{v'}^{\bar{\mathcal{E}}}(t_{\bar{\mathcal{E}}}) = L_{v'}^{\mathcal{E}}(t_{\mathcal{E}})$ and $L_{w'}^{\bar{\mathcal{E}}}(t_{\bar{\mathcal{E}}}) = L_{w'}^{\mathcal{E}}(t_{\bar{\mathcal{E}}})$ are a direct consequence of the indistinguishability of $\mathcal{E}$ and $\bar{\mathcal{E}}$.

In order to finish the proof it remains to show that $\bar{\mathcal{E}}$ is indeed a φ-framed execution, i.e., all hardware clock rates are in the range $[1, 1+\varepsilon]$ and message delays are in the range $[\varphi\mathcal{T}, (1-\varphi)\mathcal{T}]$. The hardware clock rate of each node at any point in time is between 1 and $1+\varepsilon$ and thus always in the legal range. Hence, we have to show that all messages received by some node $u \in V$ arrive after at least $\varphi\mathcal{T}$ and at most $(1-\varphi)\mathcal{T}$ time. Any message arriving immediately at time $t_{\mathcal{E}_0}$ cannot

violate these bounds because $\mathcal{E}_0$ is a φ-framed execution. Given a message sent from a node $u_s \in V$ to a node $u_r \in \mathcal{N}_{u_s}$ that arrives later than $t_{\mathcal{E}_0}$ in $\mathcal{E}$, let t_s and t_r denote the times when the message is sent and received, respectively, in execution $\mathcal{E}$ (or $\mathcal{E}_0$), and let $\bar{t}_s$ and $\bar{t}_r$ be the corresponding times in execution $\bar{\mathcal{E}}$ (or, again, $\mathcal{E}_0$).

Starting at time t', the differences between the hardware clock values of neighbors gradually shift in $\bar{\mathcal{E}}$ compared to $\mathcal{E}$, until these shifts become maximal at time $t_{\bar{\mathcal{E}}}$. Inserting the definitions, we see that at this time we have that

$$\begin{aligned} & H^{\bar{\mathcal{E}}}_{u_r}(t_{\bar{\mathcal{E}}}) - H^{\mathcal{E}}_{u_r}(t_{\bar{\mathcal{E}}}) - \left(H^{\bar{\mathcal{E}}}_{u_s}(t_{\bar{\mathcal{E}}}) - H^{\mathcal{E}}_{u_s}(t_{\bar{\mathcal{E}}})\right) \\ = & \begin{cases} \frac{\Phi^w_v(u_r)-\Phi^w_v(u_s)}{2}(1-2(1+\varepsilon)\varphi)\mathcal{T} & \text{if } \Phi^w_v(u_s), \Phi^w_v(u_r) \in [0, 2d(v',w')] \\ 0 & \text{else.} \end{cases} \end{aligned}$$

Since before t' and after $t_{\mathcal{E}}$ all clock rates are 1 in both executions, this means that at any time t it holds that

$$H^{\bar{\mathcal{E}}}_{u_r}(t) - H^{\mathcal{E}}_{u_r}(t) - \left(H^{\bar{\mathcal{E}}}_{u_s}(t) - H^{\mathcal{E}}_{u_s}(t)\right) \in \begin{cases} [-(1-2(1+\varepsilon)\varphi)\mathcal{T}, 0] & \text{if } \Phi^w_v(u_s) \geq \Phi^w_v(u_r) \\ [0, (1-2(1+\varepsilon)\varphi)\mathcal{T}] & \text{else.} \end{cases}$$

Thus, we see that the message delays $t_r - t_s$ are defined in a way ensuring that

$$t_r - t_s - \left(H^{\bar{\mathcal{E}}}_{u_r}(t) - H^{\mathcal{E}}_{u_r}(t) - \left(H^{\bar{\mathcal{E}}}_{u_s}(t) - H^{\mathcal{E}}_{u_s}(t)\right)\right) \in [(1+\varepsilon)\varphi\mathcal{T}, (1-(1+\varepsilon)\varphi)\mathcal{T}]\,. \tag{2}$$

Moreover, as all clock rates are always in the interval $[1, 1+\varepsilon]$, we have that

$$H^{\bar{\mathcal{E}}}_{u_r}(\bar{t}_r) - H^{\mathcal{E}}_{u_r}(\bar{t}_r) - \left(H^{\bar{\mathcal{E}}}_{u_r}(\bar{t}_s) - H^{\mathcal{E}}_{u_r}(\bar{t}_s)\right) \in [0, \varepsilon(\bar{t}_r - \bar{t}_s)]. \tag{3}$$

Given these relations, we can bound $\bar{t}_r - \bar{t}_s$. We compute

$$\begin{aligned} \bar{t}_r - \bar{t}_s &= \bar{t}_r - t_r - (\bar{t}_s - t_s) + (t_r - t_s) \\ &= H^{\mathcal{E}}_{u_r}(\bar{t}_r) - H^{\mathcal{E}}_{u_r}(t_r) - \left(H^{\mathcal{E}}_{u_s}(\bar{t}_s) - H^{\mathcal{E}}_{u_s}(t_s)\right) + (t_r - t_s) \\ &= H^{\mathcal{E}}_{u_r}(\bar{t}_r) - H^{\bar{\mathcal{E}}}_{u_r}(\bar{t}_r) - \left(H^{\mathcal{E}}_{u_s}(\bar{t}_s) - H^{\bar{\mathcal{E}}}_{u_s}(\bar{t}_s)\right) + (t_r - t_s) \\ &= -\left(H^{\bar{\mathcal{E}}}_{u_r}(\bar{t}_r) - H^{\mathcal{E}}_{u_r}(\bar{t}_r) - \left(H^{\bar{\mathcal{E}}}_{u_r}(\bar{t}_s) - H^{\mathcal{E}}_{u_r}(\bar{t}_s)\right)\right) \\ &\quad + (t_r - t_s) - \left(H^{\bar{\mathcal{E}}}_{u_r}(\bar{t}_s) - H^{\mathcal{E}}_{u_r}(\bar{t}_s) - \left(H^{\bar{\mathcal{E}}}_{u_s}(\bar{t}_s) - H^{\mathcal{E}}_{u_s}(\bar{t}_s)\right)\right). \end{aligned}$$

Inserting Bound (2) and Bound (3) into this equation, we obtain

$$\varphi\mathcal{T} \leq \bar{t}_r - \bar{t}_s \leq (1-(1+\varepsilon)\varphi)\mathcal{T} \leq (1-\varphi)\mathcal{T},$$

which completes the proof. □

Basically, this lemma shows that in $d(v,w)\mathcal{T}$ time we can add $\varepsilon d(v,w)\mathcal{T}$ hardware clock skew between any two nodes at distance $\varepsilon d(v,w)$ that lie on a shortest path from v to w. This is the core argument in the proof of both Theorem 3.5 and Theorem 3.6.

This lemma paves the way for the key lemma that is required to prove Theorem 3.5. It trades a quicker decrease of path lengths for a larger increase of the average clock skew on certain paths if the nodes increase their clock values quickly.

Lemma 3.4 *Fix any clock synchronization algorithm and any graph of diameter $D \geq 1/\varepsilon$. Assume that $1/\varepsilon$ is an integer (in particular, $\varepsilon \leq 1/2$), and let $X := \lceil 12 \log(8/\varepsilon) \rceil$. Set $\varphi_\varepsilon := \varepsilon/(2(1+\varepsilon))$ and $\zeta := 1 - \varepsilon - 12/X \geq 1/4$. Let a φ_ε-framed execution $\mathcal{E}_0$ ending at time $t_{\mathcal{E}_0} \geq \mathcal{T}/2$ be given such that a shortest path $p := v_0, \ldots, v_k$ with $L_{v_0}(t_{\mathcal{E}_0}) - L_{v_k}(t_{\mathcal{E}_0}) \geq \lambda \alpha \mathcal{T} k$ for some $\lambda \in \mathbb{R}$ exists, where X/ε^n divides $d(v_0, v_k) = k$ for some integer $n \geq 2 + \log_{1/\varepsilon} \log_{1/\varepsilon} D$.*

In this case, either $\mathcal{E}_0$ can be extended by a φ_ε-framed execution $\bar{\mathcal{E}}$ running from $t_{\mathcal{E}_0}$ until some time $t_{\bar{\mathcal{E}}}$, such that two nodes $v, w \in V$ exist for which

$$L_v(t_{\bar{\mathcal{E}}}) - L_w(t_{\bar{\mathcal{E}}}) \geq (\lambda + m\zeta)\, \alpha \mathcal{T} d(v, w)$$

and $d(v, w) = \varepsilon^m k/X$ for some $m \in \{1, \ldots, n\}$, or two neighbors $v, w \in V$, an execution, and some time t exist such that $L_v(t) - L_w(t) \geq \alpha \mathcal{T} \log_{1/\varepsilon} D$.

Proof. We extend $\mathcal{E}_0$ by the execution $\mathcal{E} = \mathcal{E}(\mathcal{E}_0, v_0, v_k, \varphi_\varepsilon)$ from Lemma 3.3. Define for $m \in \{1, \ldots, n\}$ that $t_m := t_{\mathcal{E}_0} + \varepsilon^{m-1}(1-\varepsilon)\mathcal{T}k/X$. We make a case differentiation. First, assume that we have $L^{\mathcal{E}}_{v_0}(t_1) - L^{\mathcal{E}}_{v_i}(t_1) \geq (\lambda - 12/X)\, \alpha \mathcal{T} i$ for some $i \geq \varepsilon k/X$. Hence, there must be two nodes $v, w \in p$ at distance $d(v, w) = \varepsilon k/X$ such that

$$L^{\mathcal{E}}_v(t_1) - L^{\mathcal{E}}_w(t_1) \geq \left(\lambda - \frac{12}{X}\right) \alpha \mathcal{T} d(v, w) \tag{4}$$

and $d(v_0, v) < d(v_0, w)$. Define $t_{\mathcal{E}} := t_1 + (1-\varepsilon) d(v, w) \mathcal{T} = t_{\mathcal{E}_0} + (1+\varepsilon)(1-\varepsilon) d(v, w) \mathcal{T}/\varepsilon$ as the time when $\mathcal{E}$ ends. Observe that $1 - \varepsilon = 1 - 2(1+\varepsilon)\varphi_\varepsilon$. Thus, due to Lemma 3.3, we can modify $\mathcal{E}$ into the φ_ε-framed execution $\bar{\mathcal{E}} = \bar{\mathcal{E}}(\mathcal{E}, v, w, \varphi_\varepsilon)$ such that $L^{\bar{\mathcal{E}}}_v(t_1) = L^{\mathcal{E}}_v(t_{\mathcal{E}})$ and $L^{\bar{\mathcal{E}}}_w(t_1) = L^{\mathcal{E}}_w(t_1)$. It follows that

$$\begin{aligned} L^{\bar{\mathcal{E}}}_v(t_1) - L^{\bar{\mathcal{E}}}_w(t_1) &= L^{\mathcal{E}}_v(t_{\mathcal{E}}) - L^{\mathcal{E}}_v(t_1) + (L^{\mathcal{E}}_v(t_1) - L^{\mathcal{E}}_w(t_1)) \\ &\overset{(1,4)}{\geq} \alpha(t_{\mathcal{E}} - t_1) + \left(\lambda - \frac{12}{X}\right) \alpha \mathcal{T} d(v, w) \\ &= (\lambda + \zeta)\, \alpha \mathcal{T} d(v, w). \end{aligned}$$

Second, assume that a pair of nodes $v, w \in p$ at distance $d(v, w) = \varepsilon^m k/X$, where $m \in \{2, \ldots, n\}$, and a time $t_{\mathcal{E}} \geq t_m + (1-\varepsilon) d(v, w) \mathcal{T}$ exist such that the inequality

$$L^{\mathcal{E}}_v(t_{\mathcal{E}}) - L^{\mathcal{E}}_w(t_{\bar{\mathcal{E}}}) \geq (\lambda + m\zeta)\, \alpha \mathcal{T} d(v, w) \tag{5}$$

holds, where $t_{\bar{\mathcal{E}}} := t_{\mathcal{E}} - \varepsilon^m (1-\varepsilon)\mathcal{T}k/X = t_{\mathcal{E}} - (1-\varepsilon) d(v, w) \mathcal{T}$. Since $t_{\mathcal{E}}$ is sufficiently large, Lemma 3.3 states that if $\mathcal{E}$ ends at time $t_{\mathcal{E}}$, it can be changed into the φ_ε-framed execution $\bar{\mathcal{E}} = \bar{\mathcal{E}}(\mathcal{E}, v, w, \varphi_\varepsilon)$ where

$$L^{\bar{\mathcal{E}}}_v(t_{\bar{\mathcal{E}}}) - L^{\bar{\mathcal{E}}}_w(t_{\bar{\mathcal{E}}}) = L^{\mathcal{E}}_v(t_{\mathcal{E}}) - L^{\mathcal{E}}_w(t_{\bar{\mathcal{E}}}) \geq (\lambda + m\zeta)\, \alpha \mathcal{T} d(v, w).$$

Third and last, assume that none of the former is true. Consider the following set of pairs of times and nodes $\left\{ \left(t^j, v^j\right) \mid j \in \{0, \ldots, j_{\max} := (1-\varepsilon)(n-1)/\varepsilon^2 \} \right\}$. Define $v^0 := v_{i(0)}$, where $i(0) := k - \left(1 - \varepsilon^{n-1}\right) k/X$. Let $m_j := 2 + \lfloor \varepsilon^2 j/(1-\varepsilon) \rfloor$ and $i_{m_j} := i(0) + (1-\varepsilon)k/\left(\varepsilon^2 X\right) \sum_{m=2}^{m_j - 1} \varepsilon^m$. Set

$$\begin{aligned} t^j &:= t_{m_j - 1} - \left(j - \frac{(1-\varepsilon)(m_j - 2)}{\varepsilon^2}\right) \frac{\varepsilon^{m_j}(1-\varepsilon)\mathcal{T}k}{X} \\ &\geq t_{m_{j_{\max}} - 1} = t_n > t_{\mathcal{E}_0} \end{aligned}$$

and $v^j := v_{i(j)}$, where $i(j) := i_{m_j-1} + \big(j - (1-\varepsilon)(m_j-2)/\left(\varepsilon^2 X\right)\big)\,\varepsilon^{m_j} k \le i(j_{\max}) = i_{n+1} = k$. These cumbersome choices of t^j and v^j ensure that, as the second case does not apply, we have

$$L^{\mathcal{E}}_{v^j}\left(t^j\right) - L^{\mathcal{E}}_{v^{j+1}}\left(t^{j+1}\right) \overset{(5)}{<} (\lambda + m_j\zeta)\,\alpha\mathcal{T}\,d\big(v^j, v^{j+1}\big)$$

for all $j \in \{0, \dots, j_{\max}-1\}$, because $d\big(v^j, v^{j+1}\big) = i(j+1) - i(j) = \varepsilon^{m_j}k/X$, $t^{j+1} \ge t_{m_j}$, and $t^j - t^{j+1} = (1-\varepsilon)d\big(v^j, v^{j+1}\big)\,\mathcal{T}$. Observe that $v^{j_{\max}} = v_k$, $t^0 = t_1$ and $t^{j_{\max}} = t_n$, and also that the v^j are well-defined because $m_j \le n$ for all but $j_{\max}$, i.e., $\varepsilon^{m_j}k/X$ is an integer for all $j \in \{0, \dots, j_{\max}-1\}$.

Summing up over all $j < j_{\max}$, we get the bound

$$\begin{aligned}
L^{\mathcal{E}}_{v^0}(t_1) - L^{\mathcal{E}}_k(t_n) &= \sum_{j=0}^{j_{\max}-1} \left(L^{\mathcal{E}}_{v^j}\left(t^j\right) - L^{\mathcal{E}}_{v^{j+1}}\left(t^{j+1}\right)\right) \\
&\overset{(5)}{<} \sum_{m=2}^{n} \sum_{\{j \,|\, m_j = m\}} (\lambda + m_j\zeta)\,\alpha\mathcal{T}\,d\big(v^j, v^{j+1}\big) \\
&= \lambda\alpha\mathcal{T}\,d\big(v^0, v_k\big) + \frac{\zeta\alpha\mathcal{T}k}{X} \sum_{m=2}^{n} \sum_{l=1}^{(1-\varepsilon)/\varepsilon^2} m\varepsilon^m \\
&< \lambda\alpha\mathcal{T}\,d\big(v^0, v_k\big) + (1-\varepsilon)\frac{\zeta\alpha\mathcal{T}k}{X} \sum_{m=0}^{\infty} (m+2)\varepsilon^m \\
&< \lambda\alpha\mathcal{T}\,d\big(v^0, v_k\big) + (1-\varepsilon)^2\frac{\alpha\mathcal{T}k}{X}\frac{2}{(1-\varepsilon)^2} \\
&= \lambda\alpha\mathcal{T}\,d\big(v^0, v_k\big) + \frac{2\alpha\mathcal{T}k}{X}. \qquad (6)
\end{aligned}$$

As the first case does not apply and $d\big(v_0, v^0\big) > (1 - 1/X)k > \varepsilon k/X$, we have that $L^{\mathcal{E}}_{v_0}(t_1) - L^{\mathcal{E}}_{v^0}(t_1) < (\lambda - 12/X)\alpha\mathcal{T}k$. We obtain

$$\begin{aligned}
L^{\mathcal{E}}_{v_k}(t_n) - L^{\mathcal{E}}_{v_k}(t_{\mathcal{E}_0}) &= L^{\mathcal{E}}_{v_k}(t_n) - L^{\mathcal{E}}_{v^0}(t_1) + \big(L^{\mathcal{E}}_{v^0}(t_1) - L^{\mathcal{E}}_{v^0}(t_{\mathcal{E}_0})\big) \\
&\quad + \big(L^{\mathcal{E}}_{v^0}(t_{\mathcal{E}_0}) - L^{\mathcal{E}}_{v_0}(t_{\mathcal{E}_0})\big) + \big(L^{\mathcal{E}}_{v_0}(t_{\mathcal{E}_0}) - L^{\mathcal{E}}_{v_k}(t_{\mathcal{E}_0})\big) \\
&\overset{(6)}{>} \left(-\lambda d\big(v^0, v_k\big) - \frac{2k}{X} - \left(\lambda - \frac{12}{X}\right) d\big(v_0, v^0\big) + \lambda d\big(v_0, v^{j_{\max}}\big)\right)\alpha\mathcal{T} \\
&> \left(12\left(1 - \frac{1}{X}\right) - 2\right)\frac{\alpha\mathcal{T}k}{X} \\
&\overset{X>12}{>} \frac{9\alpha\mathcal{T}k}{X} \\
&\ge \frac{9\alpha(t_n - t_{\mathcal{E}_0})}{\varepsilon}\log_{1/\varepsilon} D,
\end{aligned}$$

where we used that $t_n - t_{\mathcal{E}_0} = \varepsilon^{n-1}(1-\varepsilon)\mathcal{T}k/X \le \varepsilon\mathcal{T}/(X\log_{1/\varepsilon} D)$ since $n \ge 2 + \log_{1/\varepsilon}\log_{1/\varepsilon} D$. Thus, as we also have $t_n - t_{\mathcal{E}_0} > (1-\varepsilon)\mathcal{T}/\varepsilon \ge \mathcal{T} > \varphi_\varepsilon\mathcal{T}$ because X/ε^n divides k and $\varepsilon \le 1/2$, there must be times $t \in [t_{\mathcal{E}_0} + \varphi_\varepsilon\mathcal{T}/(1+\varepsilon), t_n]$ and $t' := t - \varphi_\varepsilon\mathcal{T}$ such that

$$L^{\mathcal{E}}_{v_k}(t) - L^{\mathcal{E}}_{v_k}(t') \ge \frac{9\alpha\varphi_\varepsilon\mathcal{T}}{(1+\varepsilon)\varepsilon}\log_{1/\varepsilon} D \overset{\varepsilon\le 1/2}{\ge} 2\alpha\mathcal{T}\log_{1/\varepsilon} D.$$

As $\mathcal{E}_0$ extended by $\mathcal{E}$ meets the prerequisites of Lemma 3.2 for $t > t_{\mathcal{E}_0} \geq \mathcal{T}/2 = (1+\varepsilon)\varphi_\varepsilon \mathcal{T}/\varepsilon$, an execution $\bar{\mathcal{E}}$ exists such that for any $u \in \mathcal{N}_{v^{j_{\max}}}$ the relation

$$L_{v_k}^{\bar{\mathcal{E}}}(t) - L_u^{\bar{\mathcal{E}}}(t) = L_{v_k}^{\mathcal{E}}(t') - L_u^{\mathcal{E}}(t) = L_{v_k}^{\mathcal{E}}(t') - L_{v_k}^{\mathcal{E}}(t) + L_{v_k}^{\mathcal{E}}(t) - L_u^{\mathcal{E}}(t)$$

holds. Thus, in one of the two executions, a skew of $\alpha\mathcal{T}\log_{1/\varepsilon} D$ can be observed between $v^{j_{\max}}$ and u, which concludes the case differentiation and also the proof. □

3.2 Main results

Our first result states that no continuous clock synchronization algorithm, regardless of the upper bound on the clock rates β, is able to avoid a clock skew of $\Omega\left(\alpha\mathcal{T}\log_{1/\varepsilon} D\right)$ between neighboring nodes. This result differs from the result in [3] in that the $\log\log D$ denominator is removed and the dependence on α, $\mathcal{T}$, and in particular ε is made explicit.

Theorem 3.5 *No clock synchronization algorithm can achieve a better bound on the local skew than*

$$\Omega\left(\alpha\mathcal{T}\left(1+\log_{1/\varepsilon} D\right)\right)$$

on any graph of diameter D. Furthermore, for any $\delta > 0$ and some specific diameters D and maximum drift rates ε, the local skew exceeds

$$(1-\delta)\alpha\mathcal{T}\log_{1/\varepsilon} D.$$

Proof. If $D \leq (1/\varepsilon)^c$ for any constant c, the claimed bound reduces to $\Omega(\alpha\mathcal{T})$. Such a clock skew can easily be enforced between two neighbors as shown in the proof of Theorem 3.6.

Define $\varepsilon' := 1/\lceil 1/\varepsilon \rceil > \varepsilon/2$, i.e., $1/\varepsilon'$ is an integer. Throughout this proof, we will use the notation of Lemma 3.4, however, with ε replaced by ε'. Set $b := \lceil 12\log(8/\varepsilon')\rceil/\varepsilon' = X/\varepsilon'$. As noted above, we may assume that D is sufficiently large such that $\log_{1/\varepsilon'}\log_{1/\varepsilon'} D$ is defined and we have that

$$\frac{\lfloor \log_b D \rfloor}{2} \geq 1 + \lceil \log_{1/\varepsilon'}\log_{1/\varepsilon'} D\rceil \in o(\log_b D).$$

Therefore, when setting $D_0 := (1/\varepsilon')^{1+\lceil \log_{1/\varepsilon'}\log_{1/\varepsilon'} D\rceil}$ we have that $D_0 \leq (1/\varepsilon')^{\lfloor \log_b D\rfloor/2} \leq \sqrt{D}$, implying

$$\ell := \lfloor \log_b D/D_0 \rfloor \geq \frac{\lfloor \log_b D\rfloor}{2}.$$

We state the following induction hypothesis. Assume that for $i \in \{0,\ldots,\ell-1\}$ a $\varphi_{\varepsilon'}$-framed execution $\mathcal{E}_i$ ending at a time $t_i \geq \mathcal{T}/2$ and two nodes $v_i, w_i \in V$ at distance $d(v_i,w_i) \geq b^{\ell-i}D_0$ exist, such that

$$L_{v_i}^{\mathcal{E}_i}(t_i) - L_{w_i}^{\mathcal{E}_i}(t_i) \geq i\zeta\alpha\mathcal{T}d(v_i,w_i). \tag{7}$$

We claim that in this case either the same is true for $i+m$, where $m \in \mathbb{N}$, or an execution exists where the clock skew between two neighbors becomes $\alpha\mathcal{T}\log_{1/\varepsilon'} D$ at some time.

To start the induction, we define $\mathcal{E}_0$ to be the 1/2-framed execution ending at time $t_0 := \mathcal{T}/2$ where all delays are $\mathcal{T}/2$. Apparently, at time t_0 we have two nodes $v_0, w_0 \in V$ within distance $d(v_0,w_0) = b^l D_0 \leq D$ from each other such that $L_{v_0}^{\mathcal{E}_0}(t_0) - L_{w_0}^{\mathcal{E}_0}(t_0) \geq 0$.

Now assume that Inequality (7) holds for some $i \in \{0,\ldots,\ell-1\}$. Because $b^{l-i}D_0$ is an integer multiple of X/ε^n for $n = 2 + \lceil \log_{1/\varepsilon'}\log_{1/\varepsilon'} D\rceil$, the nodes v_i, w_i and the execution $\mathcal{E}_i$ ending at

time $t_i \geq \mathcal{T}/2$ meet the requirements of Lemma 3.4. Hence, either we immediately get some execution and two neighbors exhibiting a skew of $\alpha\mathcal{T}\log_{1/\varepsilon'} D$ at some time, or for some $m \in \mathbb{N}$ a $\varphi_{\varepsilon'}$-framed execution $\mathcal{E}_{i+m}$, two nodes $v, w \in V$ at distance $d(v,w) = (\varepsilon')^m d(v_i, w_i)/X$ and a time $t_{i+m} \geq t_i \geq \mathcal{T}/2$ exist when

$$L_v^{\mathcal{E}_{i+m}}(t_i) - L_w^{\mathcal{E}_{i+m}}(t_{i+m}) \geq (i+m)\zeta\alpha\mathcal{T}d(v,w).$$

Thus, there must also be two nodes $v_{i+m}, w_{i+m} \in V$ at distance $d(v,w)/X^{m-1} = d(v_i,w_i)/b^m = b^{l-i-m}D_0$ satisfying Inequality (7) for $i+m$, i.e., the induction step succeeds.

We conclude that either an execution exists in which a skew of $\alpha\mathcal{T}\log_{1/\varepsilon} D$ occurs, or Inequality (7) holds for an $i \geq \ell \in \Omega(\log_{1/\varepsilon} D)$ in some execution. In the latter case, however, the same inequality is also true for a pair of neighboring nodes, implying that there is a clock skew of at least $\ell\zeta\alpha\mathcal{T} \in \Omega(\alpha\mathcal{T}\log_{1/\varepsilon} D)$ between two neighbors. Finally, for $\varepsilon \to 0$ we have $1/\varepsilon' - 1/\varepsilon \to 0$ and $\zeta \to 1$, and for $D \to \infty$ we get $((\log_{1/\varepsilon} D) - \ell)/\ell \to 1$. Therefore, for any $\delta > 0$, appropriate choices of ε and D imply a local skew of at least $(1-\delta)\alpha\mathcal{T}\log_{1/\varepsilon} D$. □

This theorem implies that there is no benefit if a continuous algorithm does not constrain the maximum increase of the clock values.[4] Therefore, we proceed by giving a lower bound on the local skew depending on $\beta < \infty$. This lower bound becomes stronger than the previous bound for $\beta - \alpha \in \mathcal{O}(1)$.

Theorem 3.6 *For any discrete clock synchronization algorithm and any graph of diameter D the local skew is lower bounded by*

$$\alpha\left(\left(\frac{1}{2} + \left(1 - \frac{1}{2\log(2/\varepsilon)}\right)\lfloor\log_b D\rfloor\right)\mathcal{T} - 5\right) \in \Omega\left(\alpha\mathcal{T}\left(1 + \log_{(\beta-\alpha)/(\alpha\varepsilon)} D\right)\right),$$

where $b := \left\lceil \frac{2(\beta-\alpha)(1+\varepsilon)\log(2/\varepsilon)}{\alpha\varepsilon} \right\rceil$.

Proof. Observe that any node must increase its logical clock by at least α and at most β at each of its tick events, since it does not know its own hardware clock rate, yet must obey Condition (1). Thus we have that $(1-\varepsilon)\beta \geq (1+\varepsilon)\alpha$, as otherwise infinite skew between neighbors could be built up by just manipulating the hardware clock rates. This observation implies that

$$b > \frac{2(1+\varepsilon)(\beta-\alpha)}{\alpha\varepsilon} \geq 4. \tag{8}$$

We will apply Lemma 3.3 repeatedly to produce the claimed skew between two nodes by concatenating executions whose durations decrease by the factor b. The distance between the considered pairs of nodes will also decrease by the same factor, which limits the number of steps to $\lfloor\log_b D\rfloor$.

To this end, we fix $v_0, w_0 \in V$ to be two nodes at distance $d(v_0,w_0) = b^{\lfloor\log_b D\rfloor}$. Using the notation of Lemma 3.3, we define $\mathcal{E}_0 := \mathcal{E}(\emptyset, v_0, w_0, 0)$, where $\emptyset$ means that no execution precedes $\mathcal{E}_0$. If $\mathcal{E}_0$ ends at the time $t_{\mathcal{E}_0} := (1+\varepsilon)d(v_0,w_0)\mathcal{T}/\varepsilon$, due to the lemma it can be modified into the indistinguishable execution $\bar{\mathcal{E}}_0 := \bar{\mathcal{E}}(\mathcal{E}_0, v_0, w_0)$, such that $L_{v_0}^{\bar{\mathcal{E}}_0}(t_{\bar{\mathcal{E}}_0}) = L_{v_0}^{\mathcal{E}_0}(t_{\mathcal{E}_0})$ and $L_{w_0}^{\bar{\mathcal{E}}_0}(t_{\bar{\mathcal{E}}_0}) = L_{w_0}^{\mathcal{E}_0}(t_{\bar{\mathcal{E}}_0})$. Since the minimum clock rate is α due to Condition (1), we further have that $L_{v_0}^{\mathcal{E}_0}(t_{\mathcal{E}_0}) -$

[4] The next theorem and its proof give a strong intuition that discrete algorithms are not superior to continuous ones, i.e., the same statement applies to them.

$L_{v_0}^{\mathcal{E}_0}(t_{\bar{\mathcal{E}}_0}) > \alpha(t_{\mathcal{E}_0} - t_{\bar{\mathcal{E}}_0} - 1) = \alpha(d(v_0, w_0)\mathcal{T} - 1)$. Note that we have to substract α because the condition is only applicable at ticks at node w_0. Thus, we have that

$$\begin{aligned} L_{v_0}^{\bar{\mathcal{E}}_0}(t_{\bar{\mathcal{E}}_0}) - L_{w_0}^{\bar{\mathcal{E}}_0}(t_{\bar{\mathcal{E}}_0}) &= L_{v_0}^{\mathcal{E}_0}(t_{\mathcal{E}_0}) - L_{w_0}^{\mathcal{E}_0}(t_{\bar{\mathcal{E}}_0}) \\ &> L_{v_0}^{\mathcal{E}_0}(t_{\bar{\mathcal{E}}_0}) - L_{w_0}^{\mathcal{E}_0}(t_{\bar{\mathcal{E}}_0}) + \alpha(d(v_0, w_0)\mathcal{T} - 1), \end{aligned}$$

which implies that in one of the executions the skew must exceed $\frac{\alpha}{2}(d(v_0, w_0)\mathcal{T} - 1)$. Assume w.l.o.g. that the case

$$L_{v_0}^{\bar{\mathcal{E}}_0}(t_{\bar{\mathcal{E}}_0}) - L_{w_0}^{\bar{\mathcal{E}}_0}(t_{\bar{\mathcal{E}}_0}) > \frac{\alpha}{2}(d(v_0, w_0)\mathcal{T} - 1)$$

applies.

We proceed with an induction. Define $i_{\max} := \lfloor \log_b D \rfloor$ and $\eta := \frac{\mathcal{T}}{2\log(2/\varepsilon)}$. Assume that after the ith step, where $i \in \{0, \dots, i_{\max} - 1\}$, the following holds: We have a pair of nodes $v_i, w_i \in V$ at distance $d(v_i, w_i) = b^{i_{\max} - i}$ and an execution $\bar{\mathcal{E}}_i$ ending at time $t_{\bar{\mathcal{E}}_i}$ such that

$$L_{v_i}^{\bar{\mathcal{E}}_i}(t_{\bar{\mathcal{E}}_i}) - L_{w_i}^{\bar{\mathcal{E}}_i}(t_{\bar{\mathcal{E}}_i}) > \alpha d(v_i, w_i) \left(\frac{\mathcal{T}}{2} + i(\mathcal{T} - \eta) - 3\sum_{j=1}^{i} b^{-i_{\max}+j} \right) - \alpha. \tag{9}$$

Then, we can extend $\bar{\mathcal{E}}_i$ by an execution $\bar{\mathcal{E}}_{i+1}$ ending at time $t_{\bar{\mathcal{E}}_{i+1}}$ such that there is a pair of nodes $v_{i+1}, w_{i+1} \in V$ at distance $d(v_{i+1}, w_{i+1}) = b^{i_{\max} - (i+1)}$ that satisfies Inequality (9) with i replaced by $i + 1$.

We already proved that for $i = 0$ an execution satisfying Inequality (9) exists, thus it remains to conduct the induction step. Again, we apply Lemma 3.3, this time to extend $\bar{\mathcal{E}}_i$, truncated at time $t_{\bar{\mathcal{E}}_i}$, by the execution $\mathcal{E}_{i+1} := \mathcal{E}(t_{\bar{\mathcal{E}}_i}, v_i, w_i, 0)$. Define $t_{\mathcal{E}_{i+1}} := t_{\bar{\mathcal{E}}_i} + (1 + \varepsilon)d(v_i, w_i)\mathcal{T}/(\varepsilon b)$ and $t_{\bar{\mathcal{E}}_{i+1}} := t_{\bar{\mathcal{E}}_i} + d(v_i, w_i)\mathcal{T}/(\varepsilon b)$. Due to Condition (1), which bounds the minimum and maximum clock rates, it holds that

$$\begin{aligned} L_{v_i}^{\mathcal{E}_{i+1}}(t_{\bar{\mathcal{E}}_{i+1}}) - L_{w_i}^{\mathcal{E}_{i+1}}(t_{\bar{\mathcal{E}}_{i+1}}) &> L_{v_i}^{\mathcal{E}_i}(t_{\bar{\mathcal{E}}_i}) + \alpha(t_{\bar{\mathcal{E}}_{i+1}} - t_{\bar{\mathcal{E}}_i} - 1) - L_{w_i}^{\mathcal{E}_i}(t_{\bar{\mathcal{E}}_i}) - \beta(t_{\bar{\mathcal{E}}_{i+1}} - t_{\bar{\mathcal{E}}_i} - 1) \\ &\geq L_{v_i}^{\mathcal{E}_i}(t_{\bar{\mathcal{E}}_i}) - L_{w_i}^{\mathcal{E}_i}(t_{\bar{\mathcal{E}}_i}) - (\beta - \alpha)\left(\frac{1}{\varepsilon b} d(v_i, w_i)\mathcal{T} + 1 \right) - 2\alpha \\ &> L_{v_i}^{\mathcal{E}_i}(t_{\bar{\mathcal{E}}_i}) - L_{w_i}^{\mathcal{E}_i}(t_{\bar{\mathcal{E}}_i}) - (\beta - \alpha)\frac{1 + \varepsilon}{\varepsilon b} d(v_i, w_i)\mathcal{T} - 2\alpha \\ &\geq L_{v_i}^{\mathcal{E}_i}(t_{\bar{\mathcal{E}}_i}) - L_{w_i}^{\mathcal{E}_i}(t_{\bar{\mathcal{E}}_i}) - \frac{\alpha}{2\log(2/\varepsilon)} d(v_i, w_i)\mathcal{T} - 2\alpha \\ &\overset{(9)}{>} \alpha d(v_i, w_i) \left(\frac{\mathcal{T}}{2} + i(\mathcal{T} - \eta) - \eta - 3\sum_{j=1}^{i} b^{-i_{\max}+j} \right) - 3\alpha. \end{aligned}$$

Thus, by the pidgeon hole principle, there is a pair of nodes $v_{i+1}, w_{i+1} \in V$ on a shortest path from v_i to w_i at distance $d(v_{i+1}, w_{i+1}) = b^{-1} d(v_i, w_i) = b^{i_{\max} - (i+1)}$ such that $d(v_i, v_{i+1}) < d(v_i, w_{i+1})$ and

$$L_{v_{i+1}}^{\mathcal{E}_{i+1}}(t_{\bar{\mathcal{E}}_{i+1}}) - L_{w_{i+1}}^{\mathcal{E}_{i+1}}(t_{\bar{\mathcal{E}}_{i+1}}) > \alpha d(v_{i+1}, w_{i+1}) \left(\frac{\mathcal{T}}{2} + i(\mathcal{T} - \eta) - \eta - 3\sum_{j=1}^{i+1} b^{-i_{\max}+j} \right). \tag{10}$$

According to Lemma 3.3, there is an indistinguishable execution $\bar{\mathcal{E}}_{i+1} := \bar{\mathcal{E}}(\mathcal{E}_{i+1}, v_{i+1}, w_{i+1})$ for which it holds that $L_{v_{i+1}}^{\bar{\mathcal{E}}_{i+1}}(t_{\bar{\mathcal{E}}_{i+1}}) = L_{v_{i+1}}^{\mathcal{E}_{i+1}}(t_{\mathcal{E}_{i+1}})$ and $L_{w_{i+1}}^{\bar{\mathcal{E}}_{i+1}}(t_{\bar{\mathcal{E}}_{i+1}}) = L_{w_{i+1}}^{\mathcal{E}_{i+1}}(t_{\bar{\mathcal{E}}_{i+1}})$. We conclude that

$$\begin{aligned}
L_{v_{i+1}}^{\bar{\mathcal{E}}_{i+1}}(t_{\bar{\mathcal{E}}_{i+1}}) - L_{w_{i+1}}^{\bar{\mathcal{E}}_{i+1}}(t_{\bar{\mathcal{E}}_{i+1}}) &= L_{v_{i+1}}^{\mathcal{E}_{i+1}}(t_{\mathcal{E}_{i+1}}) - L_{v_{i+1}}^{\mathcal{E}_{i+1}}(t_{\bar{\mathcal{E}}_{i+1}}) + \left(L_{v_{i+1}}^{\mathcal{E}_{i+1}}(t_{\bar{\mathcal{E}}_{i+1}}) - L_{w_{i+1}}^{\mathcal{E}_{i+1}}(t_{\bar{\mathcal{E}}_{i+1}})\right) \\
&\overset{(1,10)}{>} \alpha d(v_{i+1}, w_{i+1}) \left(\frac{\mathcal{T}}{2} + i(\mathcal{T}-\eta) - \eta - 3\sum_{j=1}^{i+1} b^{-i_{\max}+j}\right) \\
&\quad + \alpha\left(t_{\bar{\mathcal{E}}_{i+1}} - t_{\bar{\mathcal{E}}_{i+1}} - 1\right) \\
&= \alpha d(v_{i+1}, w_{i+1}) \left(\frac{\mathcal{T}}{2} + (i+1)(\mathcal{T}-\eta) - 3\sum_{j=1}^{i+1} b^{-i_{\max}+j}\right) - \alpha.
\end{aligned}$$

Hence, the induction steps succeeds. We obtain the claimed bound of

$$\begin{aligned}
L_{v_{i_{\max}}}^{\bar{\mathcal{E}}_{i_{\max}}}(t_{\bar{\mathcal{E}}_{i_{\max}}}) - L_{w_{i_{\max}}}^{\bar{\mathcal{E}}_{i_{\max}}}(t_{\bar{\mathcal{E}}_{i_{\max}}}) &\overset{(9)}{>} \alpha\left(\frac{\mathcal{T}}{2} + i_{\max}(\mathcal{T}-\eta) - 3\sum_{j=1}^{i_{\max}} b^{-i_{\max}+j}\right) - \alpha \\
&> \alpha\left(\frac{\mathcal{T}}{2} + i_{\max}(\mathcal{T}-\eta) - \frac{3}{1-1/b}\right) - \alpha \\
&\overset{(8)}{>} \alpha\left(\left(\frac{1}{2} + i_{\max}\left(1 - \frac{1}{2\log(2/\varepsilon)}\right)\right)\mathcal{T} - 5\right)
\end{aligned}$$

on the skew between the neighbors $v_{i_{\max}}$ and $w_{i_{\max}}$ in the execution $\bar{\mathcal{E}}_{i_{\max}}$ at time $t_{\bar{\mathcal{E}}_{i_{\max}}}$. □

If we demand that the logical clocks run roughly at the same rates as the hardware clocks, e.g., $\alpha = 1 - \mathcal{O}(\varepsilon)$ and $\beta = 1 + \mathcal{O}(\varepsilon)$, we get that $b \in \mathcal{O}(1)$ and thus a lower bound of $\Omega(\mathcal{T} \log D)$. if we allow a logical clock rate that is a constant times larger than real time, i.e., $\beta - \alpha \in \Theta(1)$, the lower bound reduces to $\Omega\left(\mathcal{T} \log_{1/\varepsilon} D\right)$. The constants in this bound and Theorem 3.5 are almost optimal, as the algorithm given in [5] yields the asymptotic approximation ratio of 2 for $\varepsilon \to 0$ and $\mathcal{T} \to \infty$ if $\mathcal{T}$ and ε are known.

We remark that both proofs reveal that large skews occur for a significant period of time, i.e., for any $s \in [0, 1]$ we have that for $\Theta\left(\mathcal{T} D^{1-s}\right)$ time $L_v - L_w \geq \Theta\left(s\alpha\mathcal{T} \log_b D d(v, w)\right)$ between some nodes v, w at distance $d(v, w) = \Theta\left(D^s\right)$. For $s = 1$ we also obtain a lower bound on the global skew comparable to the $\mathcal{T}D/2$ bound shown in [2].

This bound of $\mathcal{T}D/2$ on the global skew can be improved to roughly $\mathcal{T}D$ if one takes into account that it is natural to demand that all clock values are as close to real time as possible. More precisely, we may require that

$$\forall v \in V \; \forall \text{ticks } t \text{ at } v: \; (1-\varepsilon)t \leq L_v(t) \leq (1+\varepsilon)t. \tag{11}$$

It is not hard to see that this condition can easily be satisfied, while a better better approximation to real time is impossible unless external timing information is available. Interestingly, this seemingly trivial condition strengthens the lower bound on the global skew by roughly a factor of 2, regardless of the permitted minimum and maximum clock rates α and β.

Theorem 3.7 *Assume a clock synchronization algorithm $\mathcal{A}$ is equipped with initial parameters $c_1, c_2 \in (0,1]$, $\hat{\varepsilon} \in (0,1)$, and $\hat{\mathcal{T}} \in \mathbb{R}^+$ such that $c_1\hat{\mathcal{T}} \leq \mathcal{T} \leq \hat{\mathcal{T}}$ and $c_2\hat{\varepsilon} \leq \varepsilon \leq \hat{\varepsilon}$. Define $\varrho := \min\{\varepsilon, (1 - c_2\hat{\varepsilon} - c_1)/c_1\} \in [-\varepsilon, \varepsilon]$. Any such algorithm $\mathcal{A}$ obeying Condition (11) cannot avoid a global skew of at least*

$$(1 + \varrho)\mathcal{T}D$$

on any graph G of diameter D.

Proof. For the sake of simplicity, we formally allow relative clock drifts of $\varepsilon + \delta\varepsilon$, where $\delta\varepsilon$ is infinitesimally small.[5]

Let $v_0, v_D \in V$ be any two nodes at distance D. Furthermore, define that $\mathcal{T} := c_1\hat{\mathcal{T}}$, $\varepsilon' := c_2\hat{\varepsilon}$, and $\mathcal{T}' := \frac{1+\varrho}{1-\varepsilon'}\mathcal{T}$. Since $\varrho \geq -\varepsilon'$, we have that

$$c_1\hat{\mathcal{T}} = \mathcal{T} \leq \mathcal{T}' \leq \hat{\mathcal{T}}$$
$$c_2\hat{\varepsilon} = \varepsilon' \leq \varepsilon \leq \hat{\varepsilon}.$$

Thus, it is possible that $\mathcal{T}'$ is the real maximum delay and ε' is the real maximum clock drift because both values lie in the legal range according to the definition of c_1 and c_2. Assume that the maximum delay is in fact $\mathcal{T}'$ and the maximum clock drift is ε'. Consider the following two executions:

$\mathcal{E}_1$: The hardware clock rates of all clocks are $1 - \varepsilon'$ at all times. The message delays are always $\mathcal{T}'$ from any node $v \in V$ to any node $w \in \mathcal{N}_v$ if $d(v_0, w) = d(v_0, v) - 1$, and 0 otherwise.

$\mathcal{E}_2$: The hardware clock rates of all clocks are $1 + \varepsilon'$ at all times. The message delays are $\frac{(1-\varepsilon')}{1+\varepsilon'}\mathcal{T}'$ from node $v \in V$ to node $w \in \mathcal{N}_v$ if $d(v_0, w) = d(v_0, v) - 1$, and 0 otherwise.

Execution $\mathcal{E}_1$ and $\mathcal{E}_2$ are obviously legal executions as both the message delays and the clock drifts are within the legal bounds. Furthermore, $\mathcal{E}_1$ and $\mathcal{E}_2$ are indistinguishable: In execution $\mathcal{E}_1$, if a node v sends a message to w at local time H_v, w receives this message at a time t when $H_w(t) = H_v + (1 - \varepsilon')\mathcal{T}'$ if $d(v_0, w) = d(v_0, v) - 1$ and $H_w(t) = H_v$ otherwise. Since the clock rates are faster by a factor of $(1+\varepsilon')/(1-\varepsilon')$ and the message delay of any message that is sent to a node that is closer to v_0 is reduced by the same factor, the nodes receive and send the same messages at the same hardware clock times in execution $\mathcal{E}_2$.

Thus, in *both* executions nodes cannot increase their logical clock at a rate lower than their hardware clock rate, as otherwise Condition (11) would be violated in execution $\mathcal{E}_1$. Likewise, they cannot increase logical clocks faster than hardware clocks because Condition (11) would be violated in execution $\mathcal{E}_2$. Hence, in both executions it must hold that $L_v(t) = H_v(t)$ at all ticks t at v.

Assume now that $\mathcal{T}$ and ε are the correct upper bounds on the maximum delay and the maximum clock drift, respectively. Consider the following execution:

$\mathcal{E}_3$: The hardware clock rate of $v \in V$ is $1 + \varrho + \frac{D-d(v_0,v)}{D}\delta\varepsilon$, where $0 < \delta\varepsilon \ll |\varrho|$ is infinitesimally small. At time $t_0 := \frac{(1+\varrho)\mathcal{T}D}{\delta\varepsilon}$ all hardware clock rates are switched to $1 + \varrho$. If a node v sends a message at hardware clock time H_v, the message delay is adjusted in such a way that it is received at time t when $H_w(t) = H_v + (1 - \varepsilon')\mathcal{T}'$ if $d(v_0, w) = d(v_0, v) - 1$ and $H_w(t) = H_v$ otherwise.

[5]The same result could be obtained, e.g., by replacing ε by $\varepsilon - \delta\varepsilon$ and proving a bound of $(1 + \varrho - \mathcal{O}(\delta\varepsilon))\mathcal{T}D$.

Note that execution $\mathcal{E}_1$ and $\mathcal{E}_3$, and hence also $\mathcal{E}_2$ and $\mathcal{E}_3$, are indistinguishable at each node $v \in V$ by construction. It remains to verify that $\mathcal{E}_3$ is a legal execution. Since $\varrho \in [-\varepsilon, \varepsilon]$ and a clock drift of $\varepsilon + \delta\varepsilon$ is allowed, the clock drifts of all clocks are in the legal range. As far as the message delays are concerned, we have at all times $t \leq t_0$ that $H_w(t) - H_v(t) = \frac{\delta\varepsilon}{D} t \in [0, (1+\varrho)\mathcal{T}] = [0, (1-\varepsilon')\mathcal{T}']$ if $d(v_0, w) = d(v_0, v) - 1$. First, consider a message sent from v to w. If $H_w(t) - H_v(t) = (1-\varepsilon')\mathcal{T}'$, then the message delay is set to zero, which ensures that w "sees" exactly a difference of $(1-\varepsilon')\mathcal{T}'$. If $H_w(t) - H_v(t) = 0$, the message must be delayed. However, since the hardware clock rate of each node is at least $1+\varrho$, it takes at most $\frac{(1-\varepsilon')\mathcal{T}'}{1+\varrho} = \mathcal{T}$ time for w to reach the hardware clock value $H_v + (1-\varepsilon')\mathcal{T}'$. Thus in case of $d(v_0, w) = d(v_0, v) - 1$ the message delays are always in the range $[0, \mathcal{T}]$. If w sends a message to v, the same arguments apply, but in this case we need that the message delay is set to zero if $H_w(t) - H_v(t) = 0$ and at most $\frac{(1-\varepsilon')\mathcal{T}'}{1+\varrho} = \mathcal{T}$ if $H_w(t) - H_v(t) = (1-\varepsilon')\mathcal{T}'$. Note that if $d(v_0, w) = d(v_0, v)$, then $H_v(t) = H_w(t)$ as in the other two executions, and the message delay remains zero. Finally, the message delays remain in the range $[0, \mathcal{T}]$ at any time $t > t_0$, because all clocks run at the same rate, i.e., the differences between the hardware clock values do not change.

Since the nodes cannot distinguish between any of the three executions, it follows that $L_v(t) = H_v(t)$ for all nodes $v \in V$ and all ticks t also in $\mathcal{E}_3$. The skew between the hardware clocks H_{v_0} and H_{v_D} in execution $\mathcal{E}_3$ at any time $t \geq t_0$ is

$$t_0 \frac{d(v_0, v_D) - d(v_D, v_D)}{D} \delta\varepsilon = (1+\varrho)\mathcal{T}D.$$

Hence, at any tick $t \geq t_0$ at v_0 we have that $L_{v_0}^{\mathcal{E}_3}(t) - L_{v_D}^{\mathcal{E}_3}(t) = (1+\varrho)\mathcal{T}D$, which proves the stated lower bound on the global skew of $\mathcal{A}$. □

The obtained result slightly depends on how accurate the estimates of the maximum delay and the maximum drift rate are. We believe that this is not an artifact of the proof, but rather an algorithm aware of ε and $\mathcal{T}$ might deliberately choose to keep its logical clock values as close as possible to $(1-\varepsilon)t$ to avoid large skews. The estimates of $\mathcal{T}$ and ε must be extremely accurate, however, if a better bound than $(1+\varepsilon)\mathcal{T}D$ ought to be guaranteed, and a global skew of $(1-\varepsilon)\mathcal{T}D$ cannot be prevented in any situation.

Corollary 3.8 *No clock synchronization algorithm without knowledge of a lower bound on ε can avoid a global skew of $\mathcal{T}D$. No clock synchronization algorithm without knowledge of bounds on $\mathcal{T}$ stronger than $\mathcal{T} \in \left[\frac{1-\varepsilon}{1+\varepsilon}\hat{\mathcal{T}}, \hat{\mathcal{T}}\right]$ can achieve a better bound on the global skew than $(1+\varepsilon)\mathcal{T}D$.*

In [5] it is shown that $(1+\varepsilon)\mathcal{T}D$ is indeed tight up to a negligible additive term.

References

[1] H. Attiya, A. Herzberg, and S. Rajsbaum. Optimal Clock Synchronization under Different Delay Assumptions. *SIAM Journal on Computing*, 25(2):369–389, 1996.

[2] S. Biaz and J. Lundelius Welch. Closed Form Bounds for Clock Synchronization Under Simple Uncertainty Assumptions. *Information Processing Letters*, 80(3):151–157, 2001.

[3] R. Fan and N. Lynch. Gradient Clock Synchronization. In *Proc. 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 320–327, 2004.

[4] C. Lenzen, T. Locher, and R. Wattenhofer. Clock Synchronization with Bounded Global and Local Skew. In *Proc. 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 500–510, 2008.

[5] C. Lenzen, T. Locher, and R. Wattenhofer. Optimal Clock Synchronization with Bounded Clock Rates. Technical Report 301, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, 2008.

[6] T. Locher and R. Wattenhofer. Oblivious Gradient Clock Synchronization. In *Proc. 20th International Symposium on Distributed Computing (DISC)*, pages 520–533, 2006.

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

[8] N. Lynch. A Hundred Impossibility Proofs for Distributed Computing. In *Proc. 8th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 1–28, 1989.

[9] L. Meier and L. Thiele. Brief Announcement: Gradient Clock Synchronization in Sensor Networks. In *Proc. 24th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, page 238, 2005.

[10] R. Ostrovsky and B. Patt-Shamir. Optimal and Efficient Clock Synchronization under Drifting Clocks. In *Proc. 18th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 400–414, 1999.

[11] B. Patt-Shamir and S. Rajsbaum. A Theory of Clock Synchronization. In *Proc. 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 810–819, 1994.

[12] T. K. Srikanth and S. Toueg. Optimal Clock Synchronization. *Journal of the ACM*, 34(3):626–645, 1987.