Clock Synchronization with Bounded Global and Local Skew (Extended Abstract) [*]

Christoph Lenzen
Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland
lenzen@tik.ee.ethz.ch

Thomas Locher
Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland
lochert@tik.ee.ethz.ch

Roger Wattenhofer
Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland
wattenhofer@tik.ee.ethz.ch

Abstract

We present a distributed clock synchronization algorithm that guarantees an exponentially improved bound of $\mathcal{O}(\log D)$ on the clock skew between neighboring nodes in any graph G of diameter D. In light of the lower bound of $\Omega(\log D/\log\log D)$, this result is almost tight. Moreover, the global clock skew between any *two nodes, particularly nodes that are not directly connected, is bounded by $\mathcal{O}(D)$, which is optimal up to a constant factor. Our algorithm further ensures that the clock values are always within a linear envelope of real time. A better bound on the accuracy with respect to real time cannot be achieved in the absence of an external timer. These results all hold in a general model where both the clock drifts and the message delays may vary arbitrarily within pre-specified bounds.*

1 Introduction

There is a wide range of tasks in distributed systems requiring its participants to maintain a common notion of time, which necessitates the use of a synchronization algorithm. In distributed systems, the participants synchronize by perpetually sending messages containing information about their current state and by applying a clock synchronization algorithm to update their clocks.

We model a distributed system as a graph $G = (V, E)$, where the nodes in V denote the participants in the system and each edge $\{u, v\} \in E$ represents a bidirectional communication link between u and v. Each node is equipped with a hardware clock with a bounded but variable drift. A *logical clock value* is computed according to the local hardware clock value and the messages received from its neighbors. Since it is reasonable to expect that events occuring at different real times also happen at different logical times, we demand that nodes increase the value of their logical clocks at least at a certain minimum rate. The goal is to minimize the skew between the logical clocks. The main difficulty lies in the fact that the nodes know neither the potentially variable *hardware clock rates* nor the *message delays*, which can also vary arbitrarily. Moreover, there is no external clock that could inform the nodes about the real time once in a while.

Naturally, one objective is to minimize the skew between any two nodes in the graph, regardless of the distance in G between them. We call the maximum worst-case skew between any two nodes in the graph the *global skew*. Apart from minimizing the global skew, it is essential for several distributed applications that the clock skew between neighboring nodes is as small as possible. One could even think of applications where the global skew is not of great concern, but any node only needs to be well synchronized with nodes in its vicinity. 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. The so-called *gradient property*, which has been introduced in [3], captures this optimization criterion. The gradient property requires that the clock skew between any two nodes v, w for which $\{v, w\} \in E$ is as small as possible whereas the logical clock values of distant nodes are allowed to deviate more. We will refer to the maximum worst-case clock skew between neighboring nodes as the *local skew*.

Ideally, an algorithm guarantees good bounds on both the global and the local skew. It has been shown that the lowest possible global skew that any algorithm can achieve is bounded by $\Omega(D)$ [1], where D denotes the diameter of the graph. As far as the local skew is concerned, it has been proven in the surprising work by Fan and Lynch [3] that a skew of $\Omega(\log D/\log\log D)$ between neighboring nodes cannot be prevented. While it is fairly easy to come up with an algorithm guaranteeing a bound of $\Theta(D)$ on the global skew, finding an algorithm with a strong gradient

[*]The full paper is available at http://dcg.ethz.ch/publications.html.

property is more challenging. So far the best known gradient clock synchronization algorithm achieves a bound of $\mathcal{O}(\sqrt{D})$ [4]. Apparently, there is still a substantial gap between this bound on the local skew and the lower bound.

We give an algorithm that guarantees a global skew that is at most roughly a factor 2 larger than the best possible bound. More importantly, the worst-case skew between neighboring nodes is bounded by $\mathcal{O}(\log D)$ in any graph of diameter D at all times, almost closing the gap between upper and lower bound. Another aspect of our model that merits attention is the accuracy of the clocks in the absence of an external source of real time.[1] Even without such a global synchronizer, it might be desirable to keep the logical clock values as close to real time as possible. Naturally, the constant bound on the deviation of the hardware clock rates from real time directly gives an upper bound on the best possible real-time approximation. We require our algorithm to respect this bound and thus to guarantee the best feasible approximation of real time, while still bounding the global and local skew.

2 Related Work

There is a large body of work on the fundamental problem of clock synchronization, studying bounds on the skew and on communication costs [5, 7, 8]. The clock synchronization algorithm by Srikanth and Toueg [11] minimizes the global skew given a certain hardware clock drift, achieving a bound of $O(D)$. In light of the lower bound of $\frac{D}{2}$ on the clock skew in graphs of diameter D [1], this result is optimal up to a constant factor. The authors further show that the accuracy of their algorithm with respect to real time is also optimal as all clocks are always within a linear envelope of real time. However, their algorithm also incurs a skew of $\Theta(D)$ between neighboring nodes in the worst case.

The gradient property has not been studied until the lower bound of $\Omega(\log D/\log\log D)$ has been proven in the remarkable work by Fan and Lynch [3]. Meier and Thiele [6] proved that this lower bound also holds for a different model in which all messages arrive instantaneously, but the communication frequency is bounded. The best known algorithm achieves a bound of $\mathcal{O}(\sqrt{D})$ on the local skew [4]. The basic idea is that any node always sets its logical clock to the maximum clock value ever received as long as the clock of no neighboring node is more than $\mathcal{O}(\sqrt{D})$ behind. If the node is "blocked" because of such a neighbor, it refuses to further raise its clock value (beyond the increase dictated by its hardware clock) until this neighbor has caught up. This neighbor might also be blocked by one of its neighbors etc., incurring long waiting times. However, as the algorithm ensures that the global skew is bounded by $\mathcal{O}(D)$, the length of such a chain of blocked nodes is bounded by $\mathcal{O}(\sqrt{D})$, implying that after $\mathcal{O}(\sqrt{D})$ time the node at the top of the chain can raise its clock value by an amount that is large enough to ensure that the local skew never exceeds $\mathcal{O}(\sqrt{D})$. In the same work, other strategies to bound the local skew are discussed, showing that plausible strategies, such as permanently minimizing the skew to all neighbors by setting the clocks to the average clock value, fail to achieve a local skew of $o(D)$.

[1]For example, nodes in an indoor sensor network cannot receive GPS signals.

In other studies, it is often assumed that the clock drift is constant or that the message delays can be approximated efficiently as they are within given bounds, allowing for straightforward distributed least-squares optimization techniques to approximate the clock values [9, 10]. Given an external signal that occasionally informs the nodes about the real time, e.g., through a GPS service, the local skew can be bounded by a small constant $\epsilon > 0$ "some of the time" [2].

We do not require any of these restrictions on the clock drifts or the message delays, nor do we depend on an external timer, yet our algorithm achieves a global skew of $\mathcal{O}(D)$ and a local skew of $\mathcal{O}(\log D)$ at all times on any graph G.

3 Model and Definitions

We consider an arbitrary connected graph $G = (V, E)$ of diameter D. Let $\mathcal{N}_v := \{w \in V \mid \{v, w\} \in E\}$ denote the set of neighbors of v. Any node v can directly communicate with all nodes $w \in \mathcal{N}_v$. We further 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. However, each message can be delayed by any value in the range $[0, \mathcal{T}]$, where the upper bound $\mathcal{T}$ is unknown to the algorithm. In the following, we use the normalized upper bound $\mathcal{T} := 1$. Moreover, we assume that local computation requires no time and therefore does not induce an additional delay. For the sake of simplicity, we further assume that all clocks start running at real time $t = 0$. It is not hard to see that such a synchronous start is not required and that the same bounds can be shown if, e.g., a "wake-up call" activating the clocks is sent through the network.

Each node v is equipped with a *hardware clock* $H_v(\cdot)$ whose value at time t is $H_v(t) := \int_0^t h_v(\tau)\, d\tau$, 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.

Condition 3.1 $\forall v \in V\ \forall t : 1 - \varepsilon \le h_v(t) \le 1 + \varepsilon.$

While the exact value of ε is unknown, we assume that the nodes know an upper bound on ε that is strictly smaller than 1.

Additionally, each node v has a *logical clock* $L_v(\cdot)$. The algorithm can only manipulate the logical clock value by increasing it, since clocks are not allowed to run backwards. Moreover, we demand that the logical clocks be increased at least at the minimum hardware clock rate, i.e., it is also not allowed to increase them too slowly.[2]

Condition 3.2 $\forall v \in V\ \forall t < t' :\ L_v(t') - L_v(t) \geq (1 - \varepsilon)(t' - t)$.

As it is further desirable to keep all logical clock values within a linear envelope of real time, the algorithm must also respect the following condition.

Condition 3.3 $\forall v \in V\ \forall t :\ |L_v(t) - t| \leq \varepsilon\, t$.

When describing the algorithm, clock values $L_v(t)$ and other variables are considered local variables, therefore we will drop the parameter t in our notation and write L_v etc. The specification of the real time will however be helpful in the analysis.

A clock synchronization algorithm $\mathcal{A}$ specifies how the logical clock $L_v(t)$ of node v at time t is adapted according to the current value of the logical clock and the information received from its neighbors up to time t. An *execution* specifies the delays of all messages and also the hardware clock rates of all nodes at each point in time. The *global* and the *local skew* are defined as follows:

Definition 3.4 (Global Skew) *Given a clock synchronization algorithm $\mathcal{A}$, the* global skew *is defined as the value* $\sup_{G,\mathcal{E},v,w\in V,\,t}\{L_v(t) - L_w(t)\}$, *where $\mathcal{E}$ is any execution of $\mathcal{A}$ on any connected graph $G = (V, E)$.*

Definition 3.5 (Local Skew) *Given a clock synchronization algorithm $\mathcal{A}$, the* local skew *is defined as the value* $\sup_{G,\mathcal{E},v,w\in\mathcal{N}_v,\,t}\{L_v(t) - L_w(t)\}$, *where $\mathcal{E}$ is any execution of $\mathcal{A}$ on any connected graph $G = (V, E)$.*

The goal is to derive an algorithm that guarantees low bounds on both the global and the local skew. We will now present an algorithm with a global and a local skew of $\mathcal{O}(D)$ and $\mathcal{O}(\log D)$, respectively, on any graph G of diameter D.

4 Algorithm

We start by briefly sketching the basic techniques used by our algorithm, which we will henceforth refer to as $\mathcal{A}^{log}$. Since the nodes must catch up with the node whose clock runs at the highest rate, nodes have to increase their clock values once they fall behind. We say that a node *raises* its clock whenever the algorithm instantaneously increases the logical clock value.[3] Similarly to the algorithm guaranteeing a bound of $\mathcal{O}(\sqrt{D})$, $\mathcal{A}^{log}$ demands that all nodes raise their clock values to the maximum value ever received, as long as there is no neighbor whose clock is a specific value κ behind. In order for our algorithm to work, κ needs to be a certain multiple of the maximum message delay $\mathcal{T}$. We will give exact bounds for κ in the subsequent section. Note that technically the nodes do not know $\mathcal{T}$, but a closer study reveals that it suffices to determine an upper bound on the maximum delay that ever occured in the network. Such an upper bound can be obtained easily and used to determine κ.[4] Thus, we assume for simplicity that the nodes know (an upper bound on) $\mathcal{T}$.

The rule that nodes do not raise their clocks once a neighbor is at least κ behind is not sufficient to guarantee a bound of $o(D)$ on the local skew, because an execution could cause $\Theta(D/\kappa)$ nodes in a row to block each other. If the clock of a node at the end of such a chain runs at a higher rate than its neighbor's, a skew of $\Theta(D/\kappa)$ can be built up. The idea of $\mathcal{A}^{log}$ is to circumvent this problem by allowing nodes to increase their tolerance towards skew in their neighborhood: Once a large skew of (roughly) $s\kappa$ or more for $s > 1$ is detected, v will demand a clock raise from its neighbors by sending so-called *orders*, which contain the amount Δ by which the recipient of the message is supposed to raise its clock, and additionally the value s. We refer to the amount Δ as the *demand* of this order and s as its *level*. Since a node might have outstanding orders on several levels, it is possible that several orders are packed into a single message. The main challenge is to ensure that these messages are handled properly. If an adversary can trick nodes, by manipulating the hardware clock rates and message delays, into raising their clocks too quickly, other nodes might experience large skew levels and thus large local clock skews. On the other hand, a large local skew can also be built up if nodes do not raise their clocks sufficiently over time.

We will now describe the algorithm $\mathcal{A}^{\log}$, which is summarized in Algorithm 1, in greater detail. Each node v starts with $L_v := 0$ and no demand on any level s. In the absence of events, i.e., in the time periods when no messages are received, L_v is increased continuously at its own hardware clock rate $h_v(\cdot)$. Synchronization messages are sent perpetually over each edge $e \in E$ in a ping-pong fashion, i.e., after sending a message to w, node v does not send another message over the edge $e = \{v, w\}$ until v receives a message from w, and vice versa. Since it is irrelevant which

[2]In fact, only $L_v(t') - L_v(t) \geq C(t' - t)$ (as in [3]) for some constant $C > 0$ is required. However, this constant merely rescales the results, which therefore remain asymptotically the same.

[3]If the logical clock increases simply at the rate of the hardware clock, this is not called a *raise*.

[4]The nodes might, e.g., keep track of the longest round-trip time ever measured using their own hardware clocks, multiply it with an upper bound for $(1 - \varepsilon)^{-1}$, and round this value up to the next power of two. The maximum result computed anywhere in the network can then be used as the (one-way) message delay. Whenever the estimated maximum delay is doubled, all local variables can be adapted accordingly.

Algorithm 1 Node v received the message $(O_w^v, L_w, L_w^{\max})$ from node w

```
1: $\Lambda_v^w := L_v - L_w$; $O_v^w := \emptyset$
2: $\Lambda_v^{\max} := \max_{u \in \mathcal{N}_v}\{\Lambda_v^u\}$
3: $L_v^{\max} := \max\{L_v^{\max}, L_w^{\max}\}$
4: for all order$(s, \Delta_w^s) \in O_w^v$ do
5:     $\Delta_v^s := \max\{\Delta_w^s - \Gamma_v^w, \Delta_v^s\}$
6: $\Gamma_v^w := 0$
7: $R := \kappa - \Lambda_v^{\max}$
8: for $s \in \{s' \mid \Delta_v^{s'} > 0\}$ do
9:     $R := \max\{R, \min\{\Delta_v^s, s\kappa - \Lambda_v^{\max}\}\}$
10: $R := \max\{\min\{L_v^{\max} - L_v, R\}, 0\}$
11: $L_v := L_v + R$
12: for $u \in \mathcal{N}_v$ do
13:     $\Lambda_v^u := \Lambda_v^u + R$
14: for $u \in \mathcal{N}_v \setminus \{w\}$ do
15:     $\Gamma_v^u := \Gamma_v^u + R$
16: for $s \in \{s' \mid \Delta_v^{s'} > 0\}$ do
17:     $\Delta_v^s := \max\{\Delta_v^s - R, 0\}$
18: for $s \in \{s' > 1 \mid \Delta_v^{s'} + \Lambda_v^w - s'\kappa + \gamma\kappa > 0\}$ do
19:     $O_v^w := O_v^w \cup \{$order$(s, \Delta_v^s + \Lambda_v^w - s\kappa + \gamma\kappa)\}$
20: Send $(O_v^w, L_v, L_v^{\max})$ to node $w$
```

node initiates the communication over a specific edge at the beginning, we assume that all nodes know at time $t = 0$ to which nodes they must send a message and from which neighbors they can expect to receive a message. In practice, once a node has been activated, e.g., by means of a wake-up message as described in Section 3, it initiates the synchronization process by sending a first message to all neighbors. Note that with this approach two such messages might traverse an edge $e = \{v, w\}$ (in opposite directions) at the same time. This conflict can be resolved easily, e.g., by using node identifiers: The node with the larger identifier, say node v, simply drops the message from w if it is not a response to its own message, while w correctly replies to v's message. Thus, we can assume that there is always at most one messages traversing each edge. The algorithm itself describes the steps performed when a message from a neighbor w is received. For notational convenience, we assume in the following that no two messages arrive at any given node at the same time.[5] Moreover, we assume that the total number of messages sent up to any given time t remains finite for any execution.[6]

Apart from its own logical clock value, each node v also stores the estimated clock skew Λ_v^w to each neighbor w. Note that a positive Λ_v^w indicates that the clock of w is behind and consequently a negative value indicates that w's clock is ahead. Moreover, each node v also stores an estimate $L_v^{\max}$ of the largest clock value overall, initialized to $L_v(0)$. A message received from w contains a set O_w^v of orders of the form order(s, Δ_w^s), its clock value L_w at the time when it sent the message, and also its estimate $L_w^{\max}$ of the largest clock value in the network. Similarly to its own logical clock L_v, v also increases $L_v^{\max}$ at its hardware clock rate. Thus, any node assumes that the fastest clock increases at least at the same rate as its own clock.

In the first step, v updates Λ_v^w by setting it to $L_v - L_w$ and initializes the set of orders O_v^w that v will send back to w to $\emptyset$. Subsequently, the estimate $\Lambda_v^{\max}$ of the largest clock skew to the neighbors of v that are behind is determined, and the estimate $L_v^{\max}$ is updated. Each node further stores a local variable Γ_v^w, initialized to zero, for each of its neighbors $w \in \mathcal{N}_v$. This variable indicates how much v has *raised* its logical clock since it last sent a message to w. Additionally, v stores the maximum outstanding (positive) demand Δ_v^s for each level s. When updating Δ_v^s, v has to take into account that w does not know about the clock raises performed by v since it last sent a message to w, thus exactly Γ_v^w has to be subtracted from the received demand. Afterwards, since a message will be sent to w, Γ_v^w can be reset to zero. The amount R by which node v raises its clock is determined in Lines 7-10 of the algorithm: Line 7 is the rule that v can set its clock to a value at most κ larger than the lowest known clock value in its neighborhood. If there is a positive demand on a level $s > 1$, this rule is relaxed in that L_v is allowed to be $s\kappa$ larger. However, there is no need to raise the clock by more than the ordered demand, i.e., L_v is also raised at most by Δ_v^s. Line 10 simply states that L_v is neither raised above the estimate of the largest clock value $L_v^{\max}$ nor set to any value lower than L_v. After raising L_v, the estimated clock skew Λ_v^u must be updated accordingly (Line 13) for all neighbors u. The variables Γ_v^u must also be increased by R (Line 15) for all neighbors except w, and the remaining demand for each level must be corrected by subtracting R (Line 17). Then, v has to determine how much w is supposed to raise its clock. Node w must raise its clock sufficiently such that after v is informed about this raise, node v can fulfill the demand Δ_v^s, i.e., v can raise its clock by at least Δ_v^s. Thus, the remaining Δ_v^s must be increased by the amount by which the (estimated) clock skew Λ_v^w exceeds $s\kappa$. Moreover, an additional term $\gamma\kappa$ is required in order to compensate for errors in the estimates of the neighbors' clock values. An order with this demand is then added to O_v^w (Line 19). Note that if $\Delta_v^s + \Lambda_v^w - s\kappa + \gamma\kappa$ is not positive, then w does not prevent v from raising its clock by at least Δ_v^s and thus no order has to be sent. We will see that $\gamma\kappa$ has to be only marginally larger than the maximum message delay $\mathcal{T}$ in Section 5. Since in fact more demand is sent than necessary, we must ensure that demand cannot

[5]The algorithm still works without this constraint, as messages are processed sequentially.

[6]This assumption excludes irrelevant special cases in our analysis. In Section 5.3, where we examine the space and message complexity of $\mathcal{A}^{\log}$, we propose a simple modification of the algorithm ensuring that the number of messages is bounded.

be accumulated indefinitely. For this purpose, nodes constantly reduce the demand on all levels at a real-time rate of at least 2ε. As the minimal clock rate is $1-\varepsilon$, we require that nodes reduce their stored demands at the rate $\alpha h(t)$, where $\alpha \geq \frac{2\varepsilon}{1-\varepsilon}$. After the set O_v^w has been computed, v sends the message $(O_v^w, L_v, L_v^{\max})$ to w.

5 Analysis

We first point out that $\mathcal{A}^{log}$ respects Condition 3.2, as each node increases its logical clock at least at the rate $h_v(t) \geq 1-\varepsilon$ at all times t. In Line 10 of Algorithm 1, we see that $L_v(t)$ is raised at most to $L_v^{\max}(t)$. Trivially, the largest estimate $L_v^{\max}$ for any node v cannot increase faster than the maximum hardware clock rate $1+\varepsilon$, implying that $L_v(t) \leq (1+\varepsilon)t$ for all times t. Combining this observation with the fact that the algorithm respects Condition 3.2, we get that $|L_v(t) - t| \leq \varepsilon t$ for all $v \in V$ and all times t. Hence, the algorithm also respects Condition 3.3.

Since the logical clock and all local variables are updated instantaneously when a message is processed, we need to specify how these values are interpreted at any time t: If node v raises its clock at time t, $L_v(t)$ denotes the clock value of node v after the raise. In general, if more than one value is assigned to a variable at a given time t, we define the value of this variable at time t to be the last assigned value. For example, the estimated clock skew $\Lambda_v^w(t)$ incorporates any potential change due to a clock raise of node v or a message received from node w at time t. However, there is one exception to this rule. If v sends a message to w at time t, the variable Γ_v^w, indicating how much v has raised its clock since it last sent a message to w, is reset to zero. Since we frequently need the value by which the received demand has been reduced at this time, we define that $\Gamma_v^w(t)$ holds the value of Γ_v^w *before* it is reduced to zero. We will further need the following definition:

Definition 5.1 $\forall t_1 \leq t_2 : \mathcal{I}_v(t_1,t_2) := L_v(t_2) - L_v(t_1) - (1-\varepsilon)(t_2-t_1)$.

$\mathcal{I}_v(t_1,t_2)$ is the amount by which v increased its clock beyond the absolute minimum of $(1-\varepsilon)(t_2-t_1)$ in the time interval $[t_1,t_2]$ excluding any potential clock raise at time t_1. Note that $\mathcal{I}_v$ is interval additive, positive, and monotonic in both arguments. If node v raises its clock at time t, let $R_v(t)$ denote the amount by which v raised its clock, i.e., $R_v(t)$ holds the value of R in Line 11 of the algorithm. As mentioned before, we assume that no two messages arrive at any node at the same time, therefore it is not possible that a node v raises its clock more than once at any time t.

In Section 5.1 and Section 5.2 the exact bounds on the global and the local skew are presented, respectively, followed by a discussion of the bit and space complexity of $\mathcal{A}^{log}$ in Section 5.3.

5.1 Bound on the Global Skew

The proof of the bound on the global skew reveals that we require κ to be at least $1+5\varepsilon$. In the proof of the bound on the local skew κ must be larger, which, not surprisingly, indicates that the local skew imposes more restrictive conditions on the generally tolerated skew.

Theorem 5.2 (Bound on the Global Skew) *If $\kappa \geq 1+5\varepsilon$, $\mathcal{A}^{log}$ guarantees for any graph G of diameter D that the global skew is bounded by $(1+5\varepsilon)D$. This is at most a factor $2+10\varepsilon$ worse than the optimum. Moverover, a skew of exactly $(1+5\varepsilon)D$ can never be attained.*

Proof. Observe that for any node v we have $L_v(t) \leq L_v^{\max}(t)$ at any time t. Thus, it suffices to show that at all times the inequality $L_v^{\max}(t) - L_w(t) < (1+5\varepsilon)D$ holds for all $v, w \in V$. Denote by $L_{\max}(t) := \max_{v\in V}\{L_v^{\max}(t)\}$ the maximum of the nodes' estimates of the largest clock value and by $\mathcal{I}_{\max}(t,t') := L_{\max}(t') - L_{\max}(t) - (1-\varepsilon)(t'-t)$ its increase.[7] Since the estimate of the largest clock value is increased at most at a rate of $1+\varepsilon$, apparently it holds that $\mathcal{I}_{\max}(t,t') \leq 2\varepsilon(t'-t)$.

Assume for the sake of contradiction that at time $t_{\max}$ for some node $v_{\min} \in V$ the skew $L_{\max}(t_{\max}) - L_{v_{\min}}(t_{\max})$ reaches $(1+5\varepsilon)D$. Let $t_{\max}$ further be the first such point in time. Denote by $t' \in (t_{\max}-2, t_{\max}]$ the time when $v_{\min}$ receives the last message in the time interval $(t_{\max} - 2, t_{\max}]$.

First we show that $\Lambda_{v_{\min}}^w(t')$ for any given neighbor $w \in \mathcal{N}_{v_{\min}}$ of $v_{\min}$ cannot be too large. Denote by $t_r \in (t_{\max} - 2, t']$ the time when $v_{\min}$ received the last message from w and by $t_s \in [t_r - 1, t_r]$ the time when it was sent. We estimate

$$\begin{aligned}
\kappa - \Lambda_{v_{\min}}^w(t') &\geq 1+5\varepsilon - \Lambda_{v_{\min}}^w(t_r) - \mathcal{I}_{v_{\min}}(t_r,t') \quad (1)\\
&= 1+5\varepsilon + L_w(t_s) - L_{v_{\min}}(t_r)\\
&\quad -\mathcal{I}_{v_{\min}}(t_r,t')\\
&\geq (1+5\varepsilon)(1-D) + L_{\max}(t_s)\\
&\quad -L_{v_{\min}}(t_r) - \mathcal{I}_{v_{\min}}(t_r,t') \quad (2)\\
&\geq 4\varepsilon - (1+5\varepsilon)D + L_{\max}(t_r)\\
&\quad -L_{v_{\min}}(t_r) - \mathcal{I}_{v_{\min}}(t_r,t') \quad (3)\\
&= 4\varepsilon - \mathcal{I}_{\max}(t_r,t_{\max})\\
&\quad +\mathcal{I}_{v_{\min}}(t_r,t_{\max}) - \mathcal{I}_{v_{\min}}(t_r,t') \quad (4)\\
&\geq 4\varepsilon - \mathcal{I}_{\max}(t_r,t_{\max}) > 0.
\end{aligned}$$

Inequality (1) holds due to the assumption that $\kappa \geq 1+5\varepsilon$ and the fact that $\mathcal{I}_{v_{\min}}(t_r,t')$ upper bounds the clock raises

[7] This estimate may be greater than the maximum clock value, since a blocked node may increase a received estimate faster than the hardware clock rate increases the clock of the node with the currently largest clock value.

of node $v_{\min}$ in the interval $(t_r, t']$. By definition of $t_{\max}$ we have $L_{\max}(t_s) - L_w(t_s) \leq (1 + 5\varepsilon)D$ at time $t_s \leq t' \leq t_{\max}$, which gives Inequality (2). We immediately get Inequality (3) because $L_{\max}(t_r) - L_{\max}(t_s) \leq (1 + \varepsilon)(t_r - t_s) \leq 1 + \varepsilon$. Finally, Equality (4) holds since $L_{\max}(t_r) + \mathcal{I}_{\max}(t_r, t_{\max}) - L_{v_{\min}}(t_r) - \mathcal{I}_{v_{\min}}(t_r, t_{\max}) = L_{\max}(t_{\max}) - L_w(t_{\max}) = (1 + 5\varepsilon)D$. Thus, as $\kappa - \Lambda^w_{v_{\min}}(t')$ is strictly positive for any $w \in \mathcal{N}_{v_{\min}}$, indeed $v_{\min}$ cannot be blocked at time t'.

According to Line 10 of the algorithm, an unblocked node will always raise its clock to its estimate of the largest clock value, implying that $L_{v_{\min}}(t') = L^{\max}_{v_{\min}}(t')$.

Suppose a node v sends a message to a node $w \in \mathcal{N}_v$ at time t_v. This message will contain the current estimate $L^{\max}_v(t_v)$. Node w will receive this message and set its estimate $L^{\max}_w$ to at least that value not later than at time $t_v + 1$. If w has just before sent a message to one of its neighbors u, it has to wait $\tau \leq 2$ time units, before it can forward its potentially new estimate $L^{\max}_w$ to u. During this time, $L^{\max}_w$ is increased at the hardware clock rate of w. Thus, at the latest at time $t_v + \tau + 2$, u receives an estimate of at least $L^{\max}_v(t_v) + \tau(1 - \varepsilon)$. Apparently, u might also be forced to wait up to 2 time units before forwarding the new estimate. Whenever it takes $\tau < 2$ time units for a node to forward the estimate, it will arrive $2 - \tau$ time units earlier at any given destination, where it will also be increased at least at the rate $1 - \varepsilon$. Thus, repeating these arguments, at time $t \geq t_v + 3(D - 1) + 1$ any node w, and especially node $v_{\min}$, will have an estimate $L^{\max}_{v_w}$ for which it holds that $L^{\max}_{v_w}(t) \geq L^{\max}_v(t_v) + (t - t_v - D)(1 - \varepsilon)$. Since any node v sends a message to each of its neighbors at least every 2 time units, it must have sent a value $L^{\max}_v(t'_v)$ at a time $t'_v > t_{\max} - 3D$ that has been propagated (and on the way been increased by hardware clock rates) to $v_{\min}$ where it arrived at the latest at time t'. Hence, we also have that $t_0 := \min_{v \in V} t'_v > t_{\max} - 3D$. We conclude that

$$\begin{aligned}
L^{\max}_{v_{\min}}(t') &\geq L_{\max}(t_0) + (t' - t_0 - D)(1 - \varepsilon) \\
&= L_{\max}(t_{\max} - 3D) + (t' - t_{\max} + 2D) \\
&\quad (1 - \varepsilon) + \mathcal{I}_{\max}(t_{\max} - 3D, t_0) \\
&\geq L_{\max}(t_{\max}) - (1 + 5\varepsilon)D - (1 - \varepsilon) \quad (5) \\
&\quad (t_{\max} - t') + \mathcal{I}_{\max}(t_{\max} - 3D, t_0) \\
&= L_{v_{\min}}(t_{\max}) - (1 - \varepsilon)(t_{\max} - t') \\
&\quad + \mathcal{I}_{\max}(t_{\max} - 3D, t_0) \\
&\geq L_{v_{\min}}(t') + \mathcal{I}_{\max}(t_{\max} - 3D, t_0).
\end{aligned}$$

If Inequality (5) holds with equality, this implies that $L_{\max}$ increased at the rate of $1 + \varepsilon$ in the whole time interval $[t_{\max} - 3D, t_{\max}]$, thus in particular $\mathcal{I}_{\max}(t_{\max} - 3D, t_0)$ is positive. In any case, we get the contradiction $L^{\max}_{v_{\min}}(t') > L_{v_{\min}}(t') = L^{\max}_{v_{\min}}(t')$. It follows that the assumption $L_{\max}(t_{\max}) - L_{v_{\min}}(t_{\max}) = (1 + 5\varepsilon)D$ must be false, proving both that the bound on the global skew holds and that it can never be attained. The lower bound of $\frac{D}{2}$ on the global skew proven in [1] immediately yields that the bound on the global skew is optimal up to a factor of $2 + 10\varepsilon$. □

5.2 Bound on the Local Skew

A *path* is defined as a sequence of nodes $v_0, \ldots, v_k$ for which it holds that $\{v_i, v_{i+1}\} \in E$ for all $i \in \{0, \ldots, k-1\}$. Note that nodes may occur more than once in such a sequence. We will use this concept frequently in this section. 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.

More specifically, we prove that at all times the length C_s of any path with an average skew of $(s + \lambda)\kappa$ is bounded by $\beta^{-1} C_{s-1}$ for some $\lambda \in (0, 1)$ and $\beta \geq 2$. Instead of considering only adjacent levels s and $s + 1$, for any $s \in \mathbb{N}$, we also study the relation between levels s and $s + \ell$ for $\ell \geq 1$. This generalization leads to a better understanding of the algorithm and also yields improved constants in the bound on the local skew.

Definition 5.3 *Given $\beta, \lambda \in \mathbb{R}^+$ and $\ell \in \mathbb{N}_0$, 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 paths $v_0, \ldots, v_k$ of length*

$$k \geq C_s := \beta^{\ell - s} \frac{1 + 5\epsilon}{(\ell + \lambda)\kappa} D$$

we have that $L_{v_0}(t) - L_{v_k}(t) \leq k(s + \lambda)\kappa$.

In the following analysis, the statements preceding the main theorem, Theorem 5.10, assume that the network is always in a legal state.

We need to bound the time until nodes can raise their clocks in the presence of large clock skews in terms of the parameter C_s.

Lemma 5.4 *Assume that $\kappa \geq 2(1 - \varepsilon)$, $\gamma\kappa \geq 1 - \varepsilon + 2(1 + \varepsilon)\alpha$, and $\lambda + \frac{1-\varepsilon}{\kappa} \leq 1$. Given $\psi > 0$, any level $s \geq 1$, and any path $v_0, \ldots, v_k$, suppose at time t_0 the inequality*

$$L_{v_0}(t_0) - L_{v_k}(t_0) \geq ks\kappa + \psi$$

holds. Define $\bar{t}_0 := t_0 + 6C_{s-1}$. Then we have that

$$\mathcal{I}_{v_k}(t_0, \bar{t}_0) \geq \psi. \qquad (6)$$

Proof Sketch. The key observation is that the computed demands and resulting clock raises distribute the large average skew of more than $s\kappa$ on the path $v_0, ..., v_k$ over longer paths of length $l \leq C_{s-1}$, which in turn will exhibit an average skew of roughly $(s - 1)\kappa$ per node.

The proof consists of two main parts. First, we show that within $3C_{s-1}$ time units node v_k and any nodes possibly slowing down the process by preventing their neighbors from raising their clocks will have (and maintain) the necessary demand on level s such that fulfilling it ensures that they increased their clocks sufficiently. For this purpose, we analyze sequences of messages containing orders with demand on level s, which must originate from some node $v_i \in \{v_0, ..., v_{k-1}\}$ since the average skew on this path exceeds $s\kappa$.

Second, the legal state conditions bound the clock skew between v_k and nodes within distance of at most $\lceil C_{s-1} \rceil + 1$ from v_k. Since nodes maintain demand on level s, they will repeatedly raise their clocks until the demand is satisfied, unless their neighbors' clock values are too small. However, a clock difference of $s\kappa$ will be tolerated, and after at most 3 time units any neighbor will be informed about new clock values. Thus, we can show that if nodes within distance d from v_k have sufficient clock values, nodes within distance $d - 1$ will raise their clocks enough at worst 3 time units later. Using this fact inductively, after at most $\lceil 3(C_{s-1} \rceil - (k - i)) \leq 3C_{s-1}$ time units, the node within distance 0 from v_k, i.e., v_k itself, will have increased its clock sufficiently. □

This lemma basically shows that clocks are indeed increased after a certain period of time, i.e., enough demand is sent to allow nodes to raise their clock values. We now have to show that the received and stored demands cannot become too large, as otherwise nodes could be forced to raise their clocks too quickly.

The following lemma states that if a node stores a demand of Δ due to a received message, there must have been a certain clock skew in the network at some earlier time. If the network was in a legal state at that time, this clock skew is bounded due to the legal state conditions, which implies that Δ itself is bounded.

Lemma 5.5 *Assume that $(1 - \varepsilon)\alpha \geq 2\varepsilon$. Suppose node v_0 receives an* order(s, Δ_1) *at time t_0 that increases the stored demand on level s to Δ. Then a path $v_0, \ldots, v_k$ exists such that for a time $t' < t_0$ we have*

$$\Delta \leq L_{v_k}(t') - L_{v_0}(t') - ((s - \gamma)\kappa - 1 - 3\varepsilon)k - \mathcal{I}_{v_0}(t', t_0). \quad (7)$$

Proof Sketch. We analyze a sequence of messages containing orders with demand on level s: Starting at node v_0, we determine the node v_1 that sent the last message causing v_0 to increase the demand it stores on level s, and at node v_1 we repeat this procedure leading to a node v_2 etc. This way, we retrace the sequence of messages responsible for the demand received by v_0, leading to a node v_k which does not store any demand on level s. Similar computations as in Lemma 5.4 then prove the bound on the received demand. □

In the next lemma, we improve the previous result in the sense that we control the demand Δ stored at a node v_0 after receiving an order(s, Δ_1) solely in terms of the parameter C_{s-1} and $\mathcal{I}_{v_0}$. Basically, we state that if v_0 raised its clock by a large amount in a preceding time period, it must subsequently receive less demand.

Lemma 5.6 *Assume that $(1 - \varepsilon)\alpha \geq 2\varepsilon$, $\lambda + \gamma + \frac{1+3\varepsilon}{\kappa} \leq 1$, and also that $\beta \geq 2$. Suppose a node v_0 receives an* order(s, Δ_1) *at time t_0. For any $0 \leq t \leq t_0$, the demand Δ that v_0 will store on level s due to this order is bounded by*

$$\Delta \leq \kappa C_{s-1} - \mathcal{I}_{v_0}(t, t_0) + 2\varepsilon(t_0 - t).$$

Proof Sketch. The fact that an average clock skew of $(s - 1 + \lambda)\kappa$ cannot be exceeded by more than κC_{s-1} on any path is of central importance to the proof. We can deduce this fact from the legal state conditions. Furthermore, we need the following observation: If v_0 increased its clock quickly in a certain time interval, which apparently reduced the skew on the path, the skew could have increased in the same interval at a maximum rate of only 2ε. This follows from the aforementioned fact that an average skew of $(s - 1 + \lambda)\kappa$ cannot be exceeded by more than κC_{s-1} and from Lemma 5.5, which basically states that v_0 receives less demand if it increased its clock quickly in a previous period of time. After establishing these intermediate results, we easily obtain the claimed inequality from a second application of Lemma 5.5. □

As a direct consequence of Lemma 5.5 and Lemma 5.6, we get that nodes will have no demand if they increased their clocks by a specific amount in a preceding time period.

Corollary 5.7 *Suppose we have $\mathcal{I}_v(t, t') > \kappa C_{s-1} + 2\varepsilon(t' - t)$, where $v \in V$ and $t' \geq t$, and the prerequisites of Lemma 5.6 are met. In this case it holds that (1) v has no demand stored on level s or higher at time t', and (2) v does not raise its clock at time t' to a value that is more than $(s - 1)\kappa$ larger than the clock value of any of its neighbors.*

Next we summarize the conditions that must be fulfilled for us to be able to prove the claimed upper bound of $\mathcal{O}(\log D)$ on the local skew of $\mathcal{A}^{\log}$. Note that Conditions (11)-(15) are required in the main theorem itself.

Condition 5.8 *We call the parameters $\alpha, \gamma, \kappa \in \mathbb{R}^+$ of algorithm $\mathcal{A}^{\log}$ to be admissible for a given $\varepsilon > 0$, if constants $\beta \geq 2$, $\lambda \in (0, 1)$, $\ell, m \in \mathbb{N}$, and $c > 0$ exist such that the* *inequalities*

$$
\begin{aligned}
\alpha &\geq \frac{2\varepsilon}{1-\varepsilon} && (8)\\
\gamma\kappa &\geq 1-\varepsilon+2(1+\varepsilon)\alpha && (9)\\
\lambda+\gamma+\frac{1+3\varepsilon}{\kappa} &\leq 1 && (10)\\
\left(\beta^{-1}+\beta^{2-m}\right)(1+\lambda) &\leq \lambda-\frac{1}{2c} && (11)\\
\left(1-\lambda+\frac{1}{c}\right)\beta &\leq \ell && (12)\\
\beta^{m} &\leq \frac{\kappa}{24\varepsilon} && (13)\\
\beta^{\ell+1} &\leq \frac{\kappa}{24\varepsilon c} && (14)\\
\log_\beta\left(\frac{1+5\varepsilon}{(\ell+\lambda)\kappa}D\right) &> m-l-1 && (15)
\end{aligned}
$$

hold. We will say a set of constants solving this system is an admissible choice of constants for the given parameters.

Remark 5.9 *If $\varepsilon \leq 10^{-4}$ and $D \geq 10^3$, setting $\alpha := 3\cdot 10^{-4}$, $\gamma := \frac{3}{14}$ and $\kappa := 5$ is an admissible choice of parameters.*[8] *A corresponding admissable choice of constants is given by $\lambda := \frac{4}{7}$, $\beta := 4$, $\ell := 2$ $m := 5$, and $c := 14$.*

Now we are in the position to state our main result.

Theorem 5.10 *Given an admissible choice of parameters and constants, the local skew of $\mathcal{A}^{\log}$ is bounded by*

$$\kappa\left(\left\lceil\log_\beta\left(\frac{1+5\varepsilon}{(\ell+\lambda)\kappa}D\right)\right\rceil+\ell-m+2\right)\in\mathcal{O}(\log D).$$

Proof. Since we are given an admissible choice of parameters, all lemmas are applicable if the network is in a legal state. First, we show that a clock skew of $s_{\max}\kappa$ or more, where $s_{\max} := \left\lceil\log_\beta \frac{1+5\varepsilon}{(\ell+\lambda)\kappa}D\right\rceil+\ell-m+2$, cannot occur between neighboring nodes as long as the network is in a legal state. Note that $s_{\max}$ is at least 3 due to Condition (15).

Suppose that $\mathcal{A}^{\log}$ is always in a legal state until time $t_{\max}$ and suppose, for the sake of contradiction, that there are two nodes v and w such that $L_v(t)-L_w(t)\geq s_{\max}\kappa$ at time $t<t_{\max}$. Assume we have

$$L_v(t_0)-L_w(t_0)=\left(s_{\max}-\frac{1}{2}\right)\kappa$$

at a time $t_0 < t_{\max}$. Since at $t=0$ all clock values are zero, the necessary skew must somehow be introduced into the system.[9] Note that this cannot happen due to a clock raise, since there is no demand on level $s\geq s_{\max}$ in the network at time t_0 and hence nodes do not raise their clocks to a value greater than $(s_{\max}-1)\kappa$. Thus, this clock skew can only be built up by means of a larger hardware clock rate, implying that there is a time t_0 when the skew reaches exactly $\left(s_{\max}-\frac{1}{2}\right)\kappa$. The absence of demand on level $s_{\max}$ further entails that a clock skew of $L_v(t)-L_w(t)\geq s_{\max}\kappa$ can only be reached if the hardware clock rate of v's clock is larger than the rate of w's clock for a sufficiently long period of time. Since the difference between the hardware clock rates is upper bounded by 2ε, it takes at least $\frac{1}{2\varepsilon}\frac{\kappa}{2}=\frac{\kappa}{4\varepsilon}$ time to reach a skew of $s_{\max}\kappa$. However, as $L_v(t_0)-L_w(t_0)=(s_{\max}-1)\kappa+\frac{\kappa}{2}$, node w increases its clock within $6C_{s_{\max}-2}\leq 6\beta^m$ time by at least $\frac{\kappa}{2}$ above the minimum according to Lemma 5.4. Given that $6\beta^m\leq\frac{\kappa}{4\varepsilon}$, due to Condition (13), the skew is reduced at least as fast as it is built up, implying that a skew of $s_{\max}\kappa$ cannot be reached. Thus, we conclude that the claimed bound on the local skew cannot be violated as long as the network is in a legal state.

It remains to show that the network always remains in a legal state. Again, for the sake of contradiction, we assume that this not the case. Let $t_{\max}$ be the infimum of all real times where the network is not in a legal state. Since the global skew is bounded by $(1+5\varepsilon)D$ according to Theorem 5.2, a skew of $L_{v_0}(t)-L_{v_k}(t)>k(s+\lambda)\kappa$ on any path $v_0,\ldots,v_k$ implies that $k<\frac{1+5\varepsilon}{(s+\lambda)\kappa}D$. By Condition (11), we can bound $\beta^{\ell-s}\geq\left(\frac{1+\lambda}{\lambda}\right)^{\ell-s}\geq\frac{\ell+\lambda}{s+\lambda}$ for $s\in\{0,\ldots,\ell\}$. Thus, we have that $k<\frac{1+5\varepsilon}{(s+\lambda)\kappa}D\leq\beta^{\ell-s}\frac{1+5\varepsilon}{(\ell+\lambda)\kappa}D=C_s$, implying that the legal state conditions will never be violated on any level $s\in\{0,\ldots,\ell\}$. Hence it follows that a path $v_0,\ldots,v_k$ must exist where $L_{v_0}(t_{\max})-L_{v_k}(t_{\max})>k(s+\lambda)\kappa$ for some $s>\ell$. According to the preceding paragraph, a violation cannot occur on a level $s\geq s_{\max}$ first, so we also have $s<s_{\max}$. Since the path violates the legal state condition, we get the bound $k\geq C_s\geq C_{s_{\max}-1}\geq\beta^{m-2}$.

Let t_0 denote the supremum of all times $t<t_{\max}$ where we had

$$L_{v_i}(t)-L_{v_k}(t)\leq(k-i)(s-\ell)\kappa+\frac{\kappa}{2c}k \qquad (16)$$

for all $i\in\{0,\ldots,k-1\}$. We claim that for $t\geq t_0$ and $i\in\{0,\ldots,k-\lceil C_{s+1}\rceil\}$ we have

$$L_{v_i}(t)-L_{v_k}(t)\leq(k-i)s\kappa+\left(\lambda-\frac{1}{2c}\right)\kappa k+2\varepsilon(t-t_0), \qquad (17)$$

[8]A hardware clock with a clock drift of $\varepsilon=10^{-4}$ loses roughly ten seconds per day.

[9]If a wake-up message is used to initiate the synchronization algorithm, a node might increase its clock by at most $1+\varepsilon$ before its neighbors start their clocks. Since this "initial clock skew" is smaller than $\left(s_{\max}-\frac{1}{2}\right)\kappa$, the same reasoning applies.

which we will prove by induction. We start the induction at node v_i, where $i = k - \lceil C_{s+1} \rceil$. Due to the legal state conditions for $s+1$ we can bound $L_{v_i}(t) - L_{v_k}(t) \le (k - i)(s+1+\lambda)\kappa$. Condition (11) and the observation that $C_s \ge \beta^{m-2}$ allow us to bound

$$\begin{aligned}(k-i)(1+\lambda) &< \left(\beta^{-1}C_s + 1\right)(1+\lambda)\\ &\le \left(\beta^{-1} + \beta^{-m+2}\right)(1+\lambda)C_s\\ &\le \left(\lambda - \frac{1}{2c}\right)C_s.\\ &\le \left(\lambda - \frac{1}{2c}\right)k.\end{aligned}$$

Thus, $L_{v_i}(t) - L_{v_k}(t) \le (k-i)s\kappa + \left(\lambda - \frac{1}{2c}\right)\kappa k$ holds for $i = k - \lceil C_{s+1} \rceil$.

Now assume that Inequality (17) is true for some node v_i, where $i \le k - \lceil C_{s+1} \rceil$. We will now show that in this case the bound must also hold for node v_{i-1}. Assume for the sake of contradiction that there is a time $t' \ge t_0$ when

$$\begin{aligned}L_{v_{i-1}}(t') - L_{v_k}(t') &> (k-(i-1))s\kappa\\ &\quad + \left(\lambda - \frac{1}{2c}\right)\kappa k + 2\varepsilon(t' - t_0).\end{aligned}$$

Note that Inequality (17) cannot be violated by means of a fast hardware clock, as the r.h.s. of the inequality increases at a rate of 2ε, thus compensating for any contribution of hardware clock rates. Hence, v_{i-1} must have raised its clock at a specific time leading to the first violation of the bound, which w.l.o.g. we may assume to be t'.

Since Inequality (16) might be violated at time t_0, we have to consider a time t shortly before t_0. If we choose $t_0 - t > 0$ sufficiently small, by definition of t_0 and Condition (12) we get that

$$\begin{aligned}\mathcal{I}_{v_{i-1}}(t,t') &> (k-(i-1))\ell\kappa + \left(\lambda - \frac{1}{c}\right)\kappa C_s\\ &\quad + 2\varepsilon(t'-t) + \mathcal{I}_{v_k}(t,t') \qquad (18)\\ &\ge \left(\beta^{-1}\ell + \lambda - \frac{1}{c}\right)\kappa C_s + 2\varepsilon(t'-t)\\ &\ge \kappa C_s + 2\varepsilon(t'-t).\end{aligned}$$

Since v_{i-1} raises its clock at time t' and given the bound on $\mathcal{I}_{v_{i-1}}(t,t')$, applying Corollary 5.7 for level $s+1$ yields that v_{i-1} does not raise its clock to a value greater than $L_{v_i}(t') + s\kappa$, implying that

$$\begin{aligned}L_{v_i}(t') - L_{v_k}(t') &\ge L_{v_{i-1}}(t') - L_{v_k}(t') - s\kappa\\ &> (k-i)s\kappa + \left(\lambda - \frac{1}{2c}\right)\kappa k\\ &\quad + 2\varepsilon(t'-t_0),\end{aligned}$$

contradicting the assumption that Inequality (17) holds for node v_i at all times $t \ge t_0$. Hence, the claim is true for node v_{i-1} if it is true for v_i, which completes the proof of the claim.

Inserting $i = 0$ and $t = t_{\max}$ into Inequality (17), we conclude that

$$\begin{aligned}(s+\lambda)\kappa k &\le L_{v_0}(t_{\max}) - L_{v_k}(t_{\max})\\ &\le \left(s + \lambda - \frac{1}{2c}\right)\kappa k + 2\varepsilon(t_{\max} - t_0), \quad (19)\end{aligned}$$

implying that $t_{\max} - t_0 \ge \frac{\kappa}{4\varepsilon c}k \ge \frac{\kappa}{4\varepsilon c}C_s$. As Inequality (16) is violated at time t_0, Lemma 5.4 and Condition (14) yield that for time

$$t_1 := t_0 + 6C_{s-\ell-1} = t_0 + 6\beta^{\ell+1}C_s \le t_0 + \frac{\kappa}{4c\varepsilon}C_s \le t_{\max}$$

it holds that $\mathcal{I}_{v_k}(t_0,t_1) \ge \frac{\kappa}{2c}k \ge \frac{\kappa}{2c}C_s \ge 2\varepsilon(t_1 - t_0)$, showing that $t_{\max} > t_1$. Furthermore, we can now improve Inequality (17) for times $t \ge t_1$ to

$$L_{v_i}(t) - L_{v_k}(t) \le (k-i)s\kappa + \left(\lambda - \frac{1}{2c}\right)\kappa k + 2\varepsilon(t - t_1),$$

which is proved analogously. The only necessary adjustment is that we have to substract the term $2\varepsilon(t_1 - t_0)$ from the r.h.s. of Inequality (18), which is compensated for by the lower bound on $\mathcal{I}_{v_k}(t_0,t_1)$. Thus Inequality (19) sharpens to

$$\begin{aligned}(s+\lambda)\kappa k &\le L_{v_0}(t_{\max}) - L_{v_k}(t_{\max})\\ &\le \left(s + \lambda - \frac{1}{2c}\right)\kappa k + 2\varepsilon(t_{\max} - t_1).\end{aligned}$$

Hence it will take at least another $\frac{\kappa}{4c\varepsilon}C_s$ time units to reach a skew of $(s+\lambda)\kappa k$, implying that $t_{\max} \ge t_2 := t_1 + 6C_{s-\ell-1}$. Since, by definition of t_0, Inequality (16) is also violated at time $t_1 > t_0$, Lemma 5.4 yields that again $\mathcal{I}_{v_k}(t_1,t_2) \ge \frac{\kappa}{2c}C_s \ge 2\varepsilon(t_2 - t_1)$. We may repeat this argument indefinitely, leading to the contradiction $t_{\max} = \infty$. In other words, the network remains in a legal state at any time, which completes the proof. □

Remark 5.11 *We will now briefly discuss the special case of a network with a bounded diameter D, where $D < \frac{\kappa^2}{24\varepsilon}$. Set $\ell = 1$, $\lambda = 5\varepsilon$, $m := \lceil \log_\beta \frac{D}{\kappa} \rceil + 1$ and choose $\beta > 1$ small enough such that $D \le \frac{\kappa^2}{24\beta^2\varepsilon}$. Hence, Condition (13) is fulfilled. Note that Lemma 5.4 holds for $s = 1$ even without Conditions (9) and (10), since one might think of nodes as always having infinite demand on level 1. In this case, the proof of Theorem 5.2 reveals that a clock skew of 2κ cannot be reached and thus no order ever needs to be sent, implying that the remaining conditions can be dropped as well. We conclude that for any $\kappa \ge 2$ the local skew can be bounded by 2κ provided that $D \in \mathcal{O}\left(\frac{\kappa^2}{\varepsilon}\right)$. Furthermore, we directly get the simple algorithm guaranteeing a local skew of $\mathcal{O}(\sqrt{D})$ by choosing $\kappa \in \mathcal{O}(\sqrt{D})$.*

5.3 Bit and Space Complexity

We define the bit complexity of a clock synchronization algorithm to be B if any node sends at most B bits in 1 time unit. Neighboring nodes might exchange any number of messages when executing $\mathcal{A}^{log}$ in no time if the message delay is zero, implying that no bound on the bit complexity can be shown. However, this situation can be avoided by ensuring that any node v waits until at least, say, $1/2$ time units measured using its own hardware clock have passed since it last sent a message to w, before v processes the message received from w. Note that this modification does not change any proven bounds, as the nodes are never forced to wait if the message delays are larger than $\frac{1}{2(1-\varepsilon)}$ time units. Using this simple modification, we get that only one message for each neighbor can be sent every $\frac{1}{2(1+\varepsilon)}$ time units. Each message contains the current demand on all skew levels for which v has a positive demand. Let $\delta_{\max} := \max_{w \in V}\{|\mathcal{N}_w|\}$ denote the maximum degree of the graph G. Since the maximum skew level is bounded by $\mathcal{O}(\log D)$, as proven in Section 5.2, and given that any single demand is clearly bounded by $\mathcal{O}(D)$ and can thus be encoded using $\mathcal{O}(\log D)$ bits, we get that the bit complexity of $\mathcal{A}^{log}$ is bounded by $\mathcal{O}(\delta_{\max} \log^2 D)$.

The space complexity is simply the maximum number of bits each node v must store at any point in time. Since L_v is necessarily unbounded in our model, we will disregard it in this context. Apart from the local variables Γ_v^w and the estimated clock skews Λ_v^w between the clocks of v and w, each node only stores the demand Δ_v^s, if it is positive, for each skew level s. Given the bound on the local skew, we get that the values of Γ_v^w and Λ_v^w are both bounded by $\mathcal{O}(\log D)$ for each neighbor, i.e., these values in total require $\mathcal{O}(\delta_{\max} \log\log D)$ bits. As each Δ_v^s is certainly upper bounded by the global skew and s is upper bounded by $\mathcal{O}(\log D)$, storing the demands costs $\mathcal{O}(\log^2 D)$ bits. Overall, we get that the space complexity of $\mathcal{A}^{log}$ is bounded by $\mathcal{O}(\delta_{\max} \log\log D + \log^2 D)$.

Remark 5.12 *Note that nodes must both store and send real-valued data. In order to achieve the claimed bounds on the bit and space complexity, all data can only be stored with finite precision. However, if the discretization error is bounded by $\mathcal{O}(\varepsilon)$, rounding errors can also be bounded by $\mathcal{O}(\varepsilon)$. Thus all bounds still apply when ε is replaced by some appropriate $\tilde{\varepsilon} \in \mathcal{O}(\varepsilon)$.*

6 Conclusion

The presented clock synchronization algorithm is the first algorithm to guarantee a worst-case local skew of $\mathcal{O}(\log D)$, breaking the $\mathcal{O}(\sqrt{D})$ barrier. At the same time, the worst-case global skew is at most roughly a factor 2 larger than the optimum. Moreover, the algorithm reveals that a small (constant) clock skew can be guaranteed in all practical scenarios, as the network, and particularly its diameter, would have to be exceedingly large before any node could ever reach a skew of 2κ. From a theoretical point of view, the problem of minimizing the local skew is not yet solved completely, since a small gap between the new upper bound of $\mathcal{O}(\log D)$ and the lower bound of $\Omega(\log D / \log\log D)$ remains. In the paper presenting the lower bound, Fan and Lynch conjecture that $\Omega(\log D)$ is the true lower bound. We believe and—given the considerable amount of effort put into devising and analyzing the algorithm—hope that this conjecture is true.

References

[1] S. Biaz and J. L. Welch. Closed Form Bounds for Clock Synchronization Under Simple Uncertainty Assumptions. *Inf. Process. Lett.*, 80(3):151–157, 2001.

[2] R. Fan, I. Chakraborty, and N. Lynch. Clock Synchronization for Wireless Networks. In *Proc. 8th International Conference on Principles of Distributed Systems (OPODIS)*, pages 400–414, 2004.

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

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

[6] 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.

[7] 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.

[8] 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.

[9] M. Sichitiu and C. Veerarittiphan. Simple, Accurate Time Synchronization for Wireless Sensor Networks. In *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, 2003.

[10] R. Solis, V. Borkar, and P. R. Kumar. A New Distributed Time Synchronization Protocol for Multihop Wireless Networks. In *Proc. 45th IEEE Conference on Decision and Control (CDC)*, pages 2734–2739, 2006.

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