Clock Synchronization with Bounded Global and Local Skew*

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

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

*An extended abstract of this work has been accepted for publication at FOCS 2008 [4].

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 property is more challenging. So far the best known gradient clock synchronization algorithm achieves a bound of $\mathcal{O}(\sqrt{D})$ [5]. 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 [6, 8, 9]. The clock synchronization algorithm by Srikanth and Toueg [12] 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 [7] 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 [5]. 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

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

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 $o(D)$.

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 [10, 11]. 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 $\forall v \in V\ \forall t : 1 - \varepsilon \leq h_v(t) \leq 1 + \varepsilon$.

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

[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.

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

[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.

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 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]

[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.

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 $\text{order}(s, \Delta_w^s) \in O_w^v$ **do**
5: $\quad \Delta_v^s := \max\{\Delta_w^s - \Gamma_v^w, \Delta_v^s\}$
6: $\Gamma_v^w := 0$
7: $R := \kappa - \Lambda_v^{\max}$
8: **for** $s \in \{s' \mid \Delta_v^{s'} > 0\}$ **do**
9: $\quad R := \max\{R, \min\{\Delta_v^s, s\kappa - \Lambda_v^{\max}\}\}$
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: $\quad \Lambda_v^u := \Lambda_v^u + R$
14: **for** $u \in \mathcal{N}_v \setminus \{w\}$ **do**
15: $\quad \Gamma_v^u := \Gamma_v^u + R$
16: **for** $s \in \{s' \mid \Delta_v^{s'} > 0\}$ **do**
17: $\quad \Delta_v^s := \max\{\Delta_v^s - R, 0\}$
18: **for** $s \in \{s' > 1 \mid \Delta_v^{s'} + \Lambda_v^w - s'\kappa + \gamma\kappa > 0\}$ **do**
19: $\quad O_v^w := O_v^w \cup \{\text{order}(s, \Delta_v^s + \Lambda_v^w - s\kappa + \gamma\kappa)\}$
20: Send $(O_v^w, L_v, L_v^{\max})$ to node w

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 $\text{order}(s, \Delta_w^s)$, 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 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.

Since our proofs are quite involved, each of them is preceded by an overview of its main concepts. Note that some statements in these outlines are oversimplified, as the outlines are designed to merely give some intuition in a compact form.

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) *Given a graph G of diameter D and $\kappa \geq 1 + 5\varepsilon$, $\mathcal{A}^{log}$ guarantees 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 Outline. First we show that if a node is far enough behind, its neighbors will not prevent it from raising its clock, since κ is large enough. Hence, it will set its clock to its estimate of the maximum clock value in the network. This value can be at most $3D$ time units old. As the largest clock estimate never increases due to a raise, it will increase by at most $3D(1 + \varepsilon)$ in this time. However, while the estimated maximum clock value is propagated through the network, it is increased at the hardware clock rate of any node where it is temporarily stored. Thus, when arriving at a "slow" node which is about to violate the bound, the estimate is at most $3D(1 + \varepsilon) - 2D(1 - \varepsilon) = (1 + 5\varepsilon)D$ smaller than the true value.

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

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

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 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)

[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.

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, $\kappa - \Lambda^w_{v_{\min}}(t')$ is strictly positive for any $w \in \mathcal{N}_{v_{\min}}$. Hence, according to Line 10 of the algorithm, $v_{\min}$ will 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 times $t \geq t_v + 3(D-1) + 1$ 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

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

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 5.11, 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*

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

Proof Outline. 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 $3\lceil C_{s-1} \rceil - (k-i) \leq 3C_{s-1}$ time units, the node within distance 0 from v_k, i.e., v_k itself, will have increased its clock sufficiently.
Proof. In the first part of the proof, we show that nodes receive sufficient demand within $3C_{s-1}$ time to ensure that they can raise their clocks in the next $3C_{s-1}$ time units. For this purpose, we consider arbitrary extensions of the path $v_0, \ldots, v_k$, i.e., paths of the form $v_0, \ldots, v_k, \ldots, v_l$.
Claim: For all times $t \geq t_0 + 3C_{s-1}$ and all $j \geq k$ we have that

$$\Delta^s_{v_j}(t) \geq L_{v_k}(t_0) - L_{v_j}(t_0) - (j-k)(s\kappa - (1-\varepsilon)) - \mathcal{I}_{v_j}(t_0, t) + \psi. \tag{7}$$

Evaluating this inequality for $j = k$ reveals that $\Delta^s_{v_k}(t) \geq \psi - \mathcal{I}_{v_j}(t_0, t)$, i.e., v_k will raise its clock to achieve $\mathcal{I}_{v_j}(t_0, t) \geq \psi$ if none of its estimated clock differences $\Lambda^{v_j}_{v_k}$ to neighbors v_j is $s\kappa$ or more. The neighbors of v_k must have demand on level s, implying that they strive to raise their clocks as well. If we assume that a neighbor v_{k+1} of v_k does not have any demand on level s at a time $t \geq t_0 + 3C_{s-1}$, i.e., $\Delta^s_{v_{k+1}}(t) = 0$, we have that $L_{v_k}(t) - L_{v_{k+1}}(t) = L_{v_k}(t_0) + \mathcal{I}_{v_k}(t_0, t) - L_{v_{k+1}}(t_0) - \mathcal{I}_{v_{k+1}}(t_0, t) \leq s\kappa - (1-\varepsilon) + \mathcal{I}_{v_k}(t_0, t) - \psi$. This implies that v_k can only observe a skew of $s\kappa$ or more to v_{k+1} at time t if $\mathcal{I}_{v_k}(t_0, t) \geq \psi$. We proceed by

proving the claim and afterwards we generalize these motivating arguments to show the bound on $\mathcal{I}_{v_k}(t_0, \bar{t}_0)$.

Choose the largest i such that

$$L_{v_i}(t_0) - L_{v_k}(t_0) \geq (k-i)(s\kappa - (1-\varepsilon)) + \psi. \tag{8}$$

Thus, we have $L_{v_j}(t_0) - L_{v_k}(t_0) < (k-j)(s\kappa - (1-\varepsilon)) + \psi$ for all $j \in \{i+1, \ldots, k-1\}$, implying that $L_{v_i}(t_0) - L_{v_{i+1}}(t_0) > s\kappa - (1-\varepsilon)$. Given that $\gamma\kappa \geq 1-\varepsilon$, according to Line 18 of the algorithm node v_i may send an order(Δ_i, s) to v_{i+1} at a time $\tilde{t}_i \leq t_0 + 2$ when the next message from v_{i+1} arrives. Note that, Line 18 of the algorithm ensures that orders are only sent for levels greater than 1. Since nodes always tolerate a skew of κ, we can think of the demand on level 1 as being infinite all the time. Thus w.l.o.g. we can assume $s \geq 2$ in this context.

Suppose for the moment that v_i sends an order(Δ_i, s) at time $\tilde{t}_i$. Let t denote the time when v_{i+1} sent the message that arrived at node v_i at time $\tilde{t}_i$. We will treat the case where no order is sent later. We have that

$$L_{v_i}(\tilde{t}_i) - \Lambda_{v_i}^{v_{i+1}}(\tilde{t}_i) = L_{v_{i+1}}(t_0) \leq L_{v_{i+1}}(\tilde{t}_i) - \mathcal{I}_{v_{i+1}}(t_0, \tilde{t}_i) \leq L_{v_{i+1}}(\tilde{t}_i) - \Gamma_{v_{i+1}}^{v_i}(\tilde{t}_i). \tag{9}$$

The assumption $\gamma\kappa \geq 1 - \varepsilon + 2(1+\varepsilon)\alpha$ yields the bound

$$\begin{aligned}
\Delta_i &\geq \Lambda_{v_i}^{v_{i+1}}(\tilde{t}_i) - (s-\gamma)\kappa \\
&\geq L_{v_i}(\tilde{t}_i) - L_{v_{i+1}}(\tilde{t}_i) + \Gamma_{v_{i+1}}^{v_i}(\tilde{t}_i) - (s-\gamma)\kappa \\
&\geq L_{v_i}(\tilde{t}_i) - L_{v_{i+1}}(\tilde{t}_i) + \Gamma_{v_{i+1}}^{v_i}(\tilde{t}_i) - (s\kappa - (1-\varepsilon)) + 2(1+\varepsilon)\alpha
\end{aligned} \tag{10}$$

on the demand Δ_i that node v_{i+1} will receive at time $t_{i+1} \leq t_0 + 3$.

More generally, suppose node v_j receives an order(s, Δ_{j-1}) at time t_j and does not reduce its demand on level s to zero until it sends the next message to node v_{j+1} at time $\tilde{t}_j \leq t_j + 2$. According to Line 5 and Line 17 of the algorithm, v_j reduces Δ_{j-1} by $\Gamma_{v_{j-1}}^{v_j}(t_j)$ and $R_{v_j}(t_j)$ and stores at least the remaining demand. Thus, the message v_j sends will contain an order(s, Δ_j) for which it holds that

$$\begin{aligned}
\Delta_j &\geq \Delta_{j-1} - \Gamma_{v_j}^{v_{j-1}}(t_j) - R_{v_j}(t_j) - \mathcal{I}_{v_j}(t_j, \tilde{t}_j) - \alpha \int_{t_j}^{\tilde{t}_j} h_{v_j}(\tau) d\tau \\
&\quad + \Lambda_j^{j+1}(\tilde{t}_j) - (s-\gamma)\kappa && (11) \\
&\geq \Delta_{j-1} - \Gamma_{v_j}^{v_{j-1}}(t_j) - R_{v_j}(t_j) - \mathcal{I}_{v_j}(t_j, \tilde{t}_j) - 2(1+\varepsilon)\alpha \\
&\quad + L_{v_j}(\tilde{t}_j) - L_{v_{j+1}}(\tilde{t}_j) + \Gamma_{v_{j+1}}^{v_j}(\tilde{t}_j) - (s-\gamma)\kappa && (12) \\
&= \Delta_{j-1} - \Gamma_{v_j}^{v_{j-1}}(t_j) - R_{v_j}(t_j) - \mathcal{I}_{v_j}(t_j, \tilde{t}_j) - 2(1+\varepsilon)\alpha \\
&\quad + L_{v_j}(\tilde{t}_i) - L_{v_{j+1}}(\tilde{t}_i) + \mathcal{I}_{v_j}(\tilde{t}_i, \tilde{t}_j) - \mathcal{I}_{v_{j+1}}(\tilde{t}_i, \tilde{t}_j) + \Gamma_{v_{j+1}}^{v_j}(\tilde{t}_j) - (s-\gamma)\kappa \\
&\geq \Delta_{j-1} - \Gamma_{v_j}^{v_{j-1}}(\tilde{t}_{j-1}) - \mathcal{I}_{v_j}(\tilde{t}_{j-1}, t_j) - \mathcal{I}_{v_j}(t_j, \tilde{t}_j) - 2(1+\varepsilon)\alpha \\
&\quad + L_{v_j}(\tilde{t}_i) - L_{v_{j+1}}(\tilde{t}_i) + \mathcal{I}_{v_j}(\tilde{t}_i, \tilde{t}_j) - \mathcal{I}_{v_{j+1}}(\tilde{t}_i, \tilde{t}_j) + \Gamma_{v_{j+1}}^{v_j}(\tilde{t}_j) - (s-\gamma)\kappa && (13) \\
&= \Delta_{j-1} + \mathcal{I}_{v_j}(\tilde{t}_i, \tilde{t}_j) - \mathcal{I}_{v_j}(\tilde{t}_{j-1}, t_j) - \mathcal{I}_{v_j}(t_j, \tilde{t}_j) - \mathcal{I}_{v_{j+1}}(\tilde{t}_i, \tilde{t}_j) \\
&\quad + L_{v_j}(\tilde{t}_i) - L_{v_{j+1}}(\tilde{t}_i) - \Gamma_{v_j}^{v_{j-1}}(\tilde{t}_{j-1}) + \Gamma_{v_{j+1}}^{v_j}(\tilde{t}_j) - (s-\gamma)\kappa - 2(1+\varepsilon)\alpha \\
&= \Delta_{j-1} + \mathcal{I}_{v_j}(\tilde{t}_i, \tilde{t}_{j-1}) - \mathcal{I}_{v_{j+1}}(\tilde{t}_i, \tilde{t}_j) + L_{v_j}(\tilde{t}_i) - L_{v_{j+1}}(\tilde{t}_i) \\
&\quad - \Gamma_{v_j}^{v_{j-1}}(\tilde{t}_{j-1}) + \Gamma_{v_{j+1}}^{v_j}(\tilde{t}_j) - (s-\gamma)\kappa - 2(1+\varepsilon)\alpha \\
&\geq \Delta_{j-1} + \mathcal{I}_{v_j}(\tilde{t}_i, \tilde{t}_{j-1}) - \mathcal{I}_{v_{j+1}}(\tilde{t}_i, \tilde{t}_j) + L_{v_j}(\tilde{t}_i) - L_{v_{j+1}}(\tilde{t}_i) \\
&\quad - \Gamma_{v_j}^{v_{j-1}}(\tilde{t}_{j-1}) + \Gamma_{v_{j+1}}^{v_j}(\tilde{t}_j) - (s\kappa - (1-\varepsilon)). && (14)
\end{aligned}$$

As for any two times $t < t'$ the demand stored by some node v will be decreased by at most $\mathcal{I}_v(t,t') + \alpha \int_t^{t'} h_v(\tau)d\tau$ between t and t', Inequality (11) holds. Inequality (9) together with the fact that the hardware clock rate is upper bounded by $1+\varepsilon$ lead to Inequality (12). Furthermore, we have

$$\Gamma_{v_j}^{v_{j-1}}(t_j) \leq \Gamma_{v_j}^{v_{j-1}}(\tilde{t}_{j-1}) + \mathcal{I}_{v_j}(\tilde{t}_{j-1}, t_j) - R_{v_j}(t_j), \tag{15}$$

since the total increase $\mathcal{I}_{v_j}(\tilde{t}_{j-1}, t_j)$ is an upper bound on the amount node v_j raised its clock in the interval $(\tilde{t}_{j-1}, t_j)$ and, according to Line 14 of the algorithm, the potential raise at time t_j does not increase the value of $\Gamma_{v_j}^{v_{j-1}}(t_j)$. Inequality (13) can be derived using Inequality (15). Finally, Inequality (14) follows directly from the assumption that $\gamma\kappa \geq 1 - \varepsilon + 2(1+\varepsilon)\alpha$.

Inductive use of Inequality (14) combined with Inequality (10) yields that for all $j \geq i+1$, the demand Δ_{j-1} that v_j receives from v_{j-1} at time t_j is bounded by

$$\begin{aligned}
\Delta_{j-1} &\geq L_{v_i}(\tilde{t}_i) - L_{v_j}(\tilde{t}_i) - (j-i)(s\kappa - (1-\varepsilon)) + 2(1+\varepsilon)\alpha \\
&\quad -\mathcal{I}_{v_j}(\tilde{t}_i, \tilde{t}_{j-1}) + \Gamma_{v_j}^{v_{j-1}}(\tilde{t}_{j-1}) \\
&\geq L_{v_i}(t_0) - L_{v_j}(t_0) - (j-i)(s\kappa - (1-\varepsilon)) + 2(1+\varepsilon)\alpha \\
&\quad -\mathcal{I}_{v_j}(t_0, \tilde{t}_{j-1}) - \mathcal{I}_{v_j}(\tilde{t}_{j-1}, t_j) + \mathcal{I}_{v_j}(\tilde{t}_{j-1}, t_j) + \Gamma_{v_j}^{v_{j-1}}(\tilde{t}_{j-1}) \\
&\geq L_{v_i}(t_0) - L_{v_j}(t_0) - (j-i)(s\kappa - (1-\varepsilon)) + 2(1+\varepsilon)\alpha \\
&\quad -\mathcal{I}_{v_j}(t_0, t_j) + \Gamma_{v_j}^{v_{j-1}}(t_j) + R_{v_j}(t_j).
\end{aligned}$$

In the last step, we again used Inequality (15). When the order(s, Δ_{j-1}) is received at time t_j, v_j will decrease the demand it stores by both $\Gamma_{v_j}^{v_{j-1}}(t_j)$ and $R_{v_j}(t_j)$. Thus, node v_j will store at least a demand of $L_{v_i}(t_0) - L_{v_j}(t_0) - (j-i)(s\kappa - 1 + \varepsilon) + 2(1+\varepsilon)\alpha - \mathcal{I}_{v_j}(t_0, t_j)$ after processing the message. This implies that v_j will store at least $\Delta_{v_j}^s(t) \geq L_{v_i}(t_0) - L_{v_j}(t_0) - (j-i)(s\kappa - 1 + \varepsilon) - \mathcal{I}_{v_j}(t_0, t)$ demand on level s at all times $t_j \leq t \leq t_j + 2$, as the stored demand can only be reduced in two ways: First, the demand is continuously reduced at the rate $\alpha h_{v_j}(\cdot)$ (compensated by the term $2(1+\varepsilon)\alpha$), and second, clock raises (upper bounded by $\mathcal{I}_{v_j}(t_j, t)$) may reduce demand as well.

Since i is the largest index such that $L_{v_i}(t_0) - L_{v_k}(t_0) \geq (k-i)(s\kappa - 1 + \varepsilon) + \psi$ by definition, we get that

$$\begin{aligned}
\Delta_{v_j}^s(t) &\geq L_{v_i}(t_0) - L_{v_k}(t_0) + L_{v_k}(t_0) - L_{v_j}(t_0) - (j-i)(s\kappa - 1 + \varepsilon) - \mathcal{I}_{v_j}(t_0, t) \\
&\geq L_{v_k}(t_0) - L_{v_j}(t_0) - (j-k)(s\kappa - 1 + \varepsilon) - \mathcal{I}_{v_j}(t_0, t) + \psi
\end{aligned}$$

Thus, Inequality (7) is established at times $t \in [t_j, t_j + 2]$. However, since nodes continue to send demands and exchange messages with their neighbors at least every 2 time units, the previous bounds also apply to subsequent messages. Hence, Inequality (7) holds for any node v_j at times $t \geq t_j$.

Let $d(v,w)$ denote the *distance* between $v, w \in V$, i.e., the length of a shortest path connecting v and w. Since $t_j \leq t_0 + 3(j-i) = t_0 + 3((k-i) + (j-k))$ and the extension of $v_0, \ldots, v_k$ is arbitrary, the claim is true for all nodes $w \in V$ for which $d(v_k, w) \leq C_{s-1} - (k-i)$. Considering a node $v_j \in V$ where $d(v_k, v_j) \geq C_{s-1} - (k-i)$, any path $v_i, \ldots, v_k, \ldots, v_j$ must have a length of at least C_{s-1}. Thus, the legal state conditions for level $s-1$, Inequality (8), and the assumption $\lambda + \frac{1-\varepsilon}{\kappa} \leq 1$ entail that

$$\begin{aligned}
0 &\geq L_{v_i}(t_0) - L_{v_j}(t_0) - (s - 1 + \lambda)\kappa(j-i) \\
&\geq L_{v_k}(t_0) - L_{v_j}(t_0) - (s - 1 + \lambda)\kappa(j-k) + \psi \\
&\geq L_{v_k}(t_0) - L_{v_j}(t_0) - (s\kappa - 1 + \varepsilon)(j-k) - \mathcal{I}_{v_j}(t_0, t) + \psi.
\end{aligned} \tag{16}$$

Hence, Inequality (7) is true for any $t \geq t_0$.

In order to complete the proof of the claim, it remains to show that the same bounds hold if some node v_l on a path $v_i, \ldots, v_k, \ldots, v_j$, where $d(v_k, v_j) \leq C_{s-1} - (j-k)$, does not send an order on level s at time $\tilde{t}_l$. Recall that $t_l \leq \tilde{t}_l \leq t_l + 2$ denotes the time when v_l sends its next message to node v_{l+1} after receiving the order (s, Δ_{l-1}). The case $l = i$ can be treated formally by defining $t_i := t_0$. Since v_l does not send an order on level s, we have that $\Delta_l^s(\tilde{t}_l) = 0$. We can again use Inequality (14) inductively which, together with Inequality (10), yields that

$$\begin{aligned} 0 &\geq L_{v_i}(\tilde{t}_i) - L_{v_{l+1}}(\tilde{t}_i) - (l+1-i)(s\kappa - (1-\varepsilon)) + 2(1+\varepsilon)\alpha - \mathcal{I}_{v_{l+1}}(\tilde{t}_i, \tilde{t}_l) + \Gamma_{v_{l+1}}^{v_l}(\tilde{t}_l) \\ &\geq L_{v_i}(t_0) - L_{v_{l+1}}(t_0) - (l+1-i)(s\kappa - (1-\varepsilon)) - \mathcal{I}_{v_{l+1}}(t_0, \tilde{t}_l). \end{aligned} \tag{17}$$

If $l < k$, we get that the clock skew between v_{l+1} and v_k at time $\tilde{t}_l$ is at least

$$\begin{aligned} L_{v_{l+1}}(\tilde{t}_l) - L_{v_k}(\tilde{t}_l) &= L_{v_{l+1}}(t_0) + \mathcal{I}_{v_{l+1}}(t_0, \tilde{t}_l) - L_{v_k}(t_0) - \mathcal{I}_{v_k}(t_0, \tilde{t}_l) \\ &\geq L_{v_i}(t_0) - (l+1-i)(s\kappa - (1-\varepsilon)) - L_{v_k}(t_0) - \mathcal{I}_{v_k}(t_0, \tilde{t}_l) \\ &\geq (k-(l+1))(s\kappa - (1-\varepsilon)) + \psi - \mathcal{I}_{v_k}(t_0, \tilde{t}_l). \end{aligned}$$

Thus, we may consider a node $v_{i'}$, where $l < i' < k$, instead of v_i at time $\tilde{t}_l \leq t_0 + 3(l-i) + 2 < t_0 + 3(l+1-i)$ as the source of the first order message, and prove the lemma by showing that v_k increases its clock by at least $\psi' := \psi - \mathcal{I}_{v_k}(t_0, \tilde{t}_l)$ in the time interval $(\tilde{t}_l, \bar{t}_0]$. This argument can also be used inductively in case we again have $l < k$ on this shorter path.

In case of $l \geq k$, Inequality (17) and Inequality (8) together imply that

$$\begin{aligned} \mathcal{I}_{v_{l+1}}(t_0, \tilde{t}_l) &\geq L_{v_i}(t_0) - L_{v_{l+1}}(t_0) - (l+1-i)(s\kappa - (1-\varepsilon)) \\ &\geq L_{v_k}(t_0) - L_{v_{l+1}}(t_0) - (l+1-k)(s\kappa - (1-\varepsilon)) + \psi. \end{aligned} \tag{18}$$

Hence, node v_{l+1} already increased its clock sufficiently to fulfill Inequality (7) at any time $t \geq \tilde{t}_l$. Lastly, assume that some node $l' > l+1$ still requires a demand

$$\psi' := L_{v_k}(t_0) - L_{v_{l'}}(t_0) - (l'-k)(s\kappa - (1-\varepsilon)) - \mathcal{I}_{l'}(t_0, \tilde{t}_l) + \psi > 0$$

to fulfill the bound at time $\tilde{t}_l$. In this case, we apply Inequality (18) to see that

$$\begin{aligned} L_{v_{l+1}}(\tilde{t}_l) - L_{v_{l'}}(\tilde{t}_l) &= L_{v_{l+1}}(t_0) + \mathcal{I}_{v_{l+1}}(t_0, \tilde{t}_l) - L_{v_{l'}}(t_0) - \mathcal{I}_{v_{l'}}(t_0, \tilde{t}_l) \\ &\geq L_{v_k}(t_0) - (l+1-k)(s\kappa - (1-\varepsilon)) + \psi - L_{v_{l'}}(t_0) - \mathcal{I}_{v_{l'}}(t_0, \tilde{t}_l) \\ &= (l'-k)(s\kappa - (1-\varepsilon)) - (l+1-k)(s\kappa - (1-\varepsilon)) + \psi' \\ &= (l'-(l+1))(s\kappa - (1-\varepsilon)) + \psi'. \end{aligned}$$

However, recalling that $\tilde{t}_l \leq t_{l+1} \leq t_0 + 3(l+1-i)$ and $l' - i \leq C_{s-1}$, this case can be covered analogously: Again some node must start to send orders, and the same calculations show that $v_{l'}$ and its successors will receive sufficient demand in time. This completes the proof of the claim.

In order to finish the proof of the lemma, we have to show that since Inequality (7) holds for all nodes, v_k can raise its clock fast enough in additional $3C_{s-1}$ time units such that $\mathcal{I}_{v_k}(t_0, \bar{t}_0) \geq \psi$. We will prove this by induction on the distance $d(v_k, v_j)$, where each step will cost at most 3 time units. Define $d_0 := \lceil C_{s-1} \rceil - (k-i)$. The induction hypothesis states for $d \in \{0, \ldots, d_0+1\}$, for all times $t \geq t^d := t_0 + 3C_{s-1} + 3(d_0 - d)$, and all nodes $v_j \in V$ where $d(v_k, v_j) \leq d$ that

$$\mathcal{I}_{v_j}(t_0, t) \geq L_{v_k}(t_0) - L_{v_j}(t_0) - (s\kappa - 1 + \varepsilon)d + \psi. \tag{19}$$

Since $d_0 < C_{s-1}$, Inequality (19) for $j = k$ yields the desired statement $\mathcal{I}_{v_k}(t_0, \bar{t}_0) \geq \mathcal{I}_{v_k}(t_0, t^0) \geq \psi$.

We start the induction by proving that Inequality (19) holds for $d = d_0 + 1$. If $d(v_k, v_j)$ is exactly either d_0 or $d_0 + 1$, this follows immediately from Inequality (16) and the fact that $t^{d_0+1} = t_0 + 3C_{s-1} - 3 \geq t_0$, as $C_{s-1} \geq C_s \geq 1$ must hold in a legal state. If we consider some node v_j where $d(v_k, v_j) < d$, a path $v_i, \ldots, v_k, \ldots, v_j$ of length d_0 or $d_0 + 1$ must exist, since we can extend the concatenation of $v_i, \ldots, v_k$ and a shortest path $v_k, \ldots, v_j$ by any number of repetitions of, say, v_j, v_{j-1}, v_j. Hence, evaluating Inequality (16) for this path shows that Inequality (19) holds also in case $d(v_k, v_j) < d$.

Now assume that the induction hypothesis is true for $d + 1$ and fix some $v_j \in V$ where $d(v_k, v_j) \leq d$. The claim states that v_j has positive demand on level s at any time $t \geq t_0 + 3C_{s-1}$ unless it fulfills Inequality (19). The estimate for the maximum clock value $L_{v_j}^{\max}$ is also sufficiently large for v_j to raise its clock: The estimate of the largest clock value $L_{v_j}^{\max}(t)$ is at least the value $L_i(t_0)$ plus the amount by which the estimate is increased until t. On the way from v_i to v_j, the estimate can be in transit for a total duration of at most $j - i$ during which the estimate is not increased, i.e., we have that $L_{v_j}^{\max}(t) \geq L_{v_i}(t_0) + ((t - t_0) - (j - i))(1 - \varepsilon)$. Thus, since $j - i \leq k - i + d$ and $s\kappa - (1 - \varepsilon) \geq 1 - \varepsilon$ by the assumption that $\kappa \geq 2(1 - \varepsilon)$, we get

$$\begin{aligned} L_{v_j}^{\max}(t) - L_{v_j}(t) &\geq L_{v_i}(t_0) + ((t - t_0) - (j - i))(1 - \varepsilon) \\ &\quad -(L_{v_j}(t_0) + \mathcal{I}_{v_j}(t_0, t) + (1 - \varepsilon)(t - t_0)) \\ &\geq L_{v_i}(t_0) - L_{v_j}(t_0) - (s\kappa - 1 + \varepsilon)(k - i + d) - \mathcal{I}_{v_j}(t_0, t) \\ &\geq L_{v_k}(t_0) - L_{v_j}(t_0) - (s\kappa - 1 + \varepsilon)d - \mathcal{I}_{v_j}(t_0, t) + \psi, \end{aligned}$$

which is positive if Inequality (19) is violated.

Thus, it remains to show that no neighbor prevents v_j from raising its clock unless Inequality (19) holds. Assuming the contrary, define $t' \in [t^d - 2, t^d]$ as the time when v_j receives the last message until t^d. Denote by $t_r \in [t^d - 2, t']$ the time when v_j received the last message from some neighbor $w \in \mathcal{N}_{v_j}$ until time t' and let $t_s \geq t^d - 3 = t^{d+1}$ be the corresponding sending time. Since $d(v_k, w) \leq d + 1$, we estimate

$$\begin{aligned} L_{v_j}(t_s) - L_w(t_s) &= L_{v_j}(t_0) - L_w(t_0) + \mathcal{I}_{v_j}(t_0, t^d) - \mathcal{I}_{v_j}(t_s, t^d) - \mathcal{I}_w(t_0, t_s) \\ &< L_{v_k}(t_0) - L_w(t_0) - (s\kappa - 1 + \varepsilon)d + \psi - \mathcal{I}_{v_j}(t_s, t^d) - \mathcal{I}_w(t_0, t_s) \quad (20) \\ &\leq s\kappa - 1 + \varepsilon - \mathcal{I}_{v_j}(t_s, t^d) - \mathcal{I}_w(t^{d+1}, t_s) \quad (21) \\ &\leq s\kappa - 1 + \varepsilon - \mathcal{I}_{v_j}(t_s, t^d). \end{aligned}$$

Inequality (20) uses the assumption that Inequality (19) is violated by v_j at time t^d, whereas it is true for node w at time t_s, which leads to Inequality (21). We get the following bound on $\Lambda_{v_j}^w(t')$:

$$\begin{aligned} \Lambda_{v_j}^w(t') &\leq \Lambda_{v_j}^w(t_r) + \mathcal{I}_{v_j}(t_r, t') \\ &= L_{v_j}(t_r) - L_w(t_s) + \mathcal{I}_{v_j}(t_r, t') \\ &\leq L_{v_j}(t_s) - L_w(t_s) + 1 - \varepsilon + \mathcal{I}_{v_j}(t_s, t_r) + \mathcal{I}_{v_j}(t_r, t') \\ &< s\kappa - \mathcal{I}_{v_j}(t_s, t^d) + \mathcal{I}_{v_j}(t_s, t') \leq s\kappa. \end{aligned}$$

This contradicts the assumption that v_j observes a skew of at least $s\kappa$ to a neighbor, implying that the induction step succeeds. Inserting $j = k$, $d = 0$, and $t = \bar{t}_0$ into Inequality (19) yields $\mathcal{I}_{v_k}(\bar{t}_0) \geq \psi$ as desired. □

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*

$$\Delta \leq L_{v_k}(t') - L_{v_0}(t') - ((s-\gamma)\kappa - 1 - 3\varepsilon)k - \mathcal{I}_{v_0}(t', t_0). \tag{22}$$

Proof Outline. Again, we analyze a sequence of messages containing orders with demand on level s, but this time we construct this sequence "backwards": 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.

Proof. We consider a specific path $v_0, v_1, \ldots, v_k$ starting at node v_0. Node v_1 is the node that sent the order(s, Δ_1), at time $\tilde{t}_1$, leading to an increase of the stored demand at v_0. If v_1 itself had demand on level s at time $\tilde{t}_1$, let $t_1 \leq \tilde{t}_1$ denote the last time when v_1 received an order(s, Δ_2) from one of its neighbors that caused v_1 to increase its demand. This neighbor is node v_2 in the path. As long as we do not arrive at a node that has no demand stored on level s, we can iteratively extend the constructed path $v_0, v_1, \ldots, v_i$ by adding the particular node v_{i+1} that caused the last increase of the stored demand at node v_i at a time t_i due to an order v_{i+1} sent at time $\tilde{t}_{i+1}$. This path is finite in length, since the number of messages up to time t_0 is finite, hence the construction will eventually stop at some node v_k, i.e., v_k does not store demand on level s at time $\tilde{t}_k$.

For $i \in \{0, \ldots, k-1\}$ we bound the demand Δ_i that v_i sent to v_{i-1} at time $\tilde{t}_i$:

$$\begin{aligned}
\Delta_i &\leq \Delta_{i+1} - \Gamma_{v_i}^{v_{i+1}}(t_i) - R_{v_i}(t_i) - \mathcal{I}_{v_i}(t_i, \tilde{t}_i) + \Lambda_{v_i}^{v_{i-1}}(\tilde{t}_i) - (s-\gamma)\kappa && (23)\\
&\leq \Delta_{i+1} - \Gamma_{v_i}^{v_{i+1}}(t_i) - R_{v_i}(t_i) - \mathcal{I}_{v_i}(t_i, \tilde{t}_i) + L_{v_i}(\tilde{t}_i) - L_{v_{i-1}}(\tilde{t}_i)\\
&\quad + \Gamma_{v_{i-1}}^{v_i}(\tilde{t}_i) - (s-\gamma)\kappa + 1 + \varepsilon && (24)\\
&= \Delta_{i+1} - \Gamma_{v_i}^{v_{i+1}}(t_i) - R_{v_i}(t_i) - \mathcal{I}_{v_i}(t_i, \tilde{t}_i) + L_{v_i}(\tilde{t}_k) - L_{v_{i-1}}(\tilde{t}_k)\\
&\quad + \mathcal{I}_{v_i}(\tilde{t}_k, \tilde{t}_i) - \mathcal{I}_{v_{i-1}}(\tilde{t}_k, \tilde{t}_i) + \Gamma_{v_{i-1}}^{v_i}(\tilde{t}_i) - (s-\gamma)\kappa + 1 + \varepsilon\\
&\leq \Delta_{i+1} - \Gamma_{v_i}^{v_{i+1}}(\tilde{t}_i) - \mathcal{I}_{v_i}(\tilde{t}_{i+1}, t_i) + 2\varepsilon - \mathcal{I}_{v_i}(t_i, \tilde{t}_i) + L_{v_i}(\tilde{t}_k) - L_{v_{i-1}}(\tilde{t}_k)\\
&\quad + \mathcal{I}_{v_i}(\tilde{t}_k, \tilde{t}_i) - \mathcal{I}_{v_{i-1}}(\tilde{t}_k, \tilde{t}_i) + \Gamma_{v_{i-1}}^{v_i}(\tilde{t}_i) - (s-\gamma)\kappa + 1 + \varepsilon && (25)\\
&= \Delta_{i+1} + \mathcal{I}_{v_i}(\tilde{t}_k, \tilde{t}_i) - \mathcal{I}_{v_i}(\tilde{t}_{i+1}, \tilde{t}_i) - \mathcal{I}_{v_{i-1}}(\tilde{t}_k, \tilde{t}_i) + L_{v_i}(\tilde{t}_k) - L_{v_{i-1}}(\tilde{t}_k)\\
&\quad - \Gamma_{v_i}^{v_{i+1}}(\tilde{t}_{i+1}) + \Gamma_{v_{i-1}}^{v_i}(\tilde{t}_i) - (s-\gamma)\kappa + 1 + 3\varepsilon\\
&= \Delta_{i+1} + \mathcal{I}_{v_i}(\tilde{t}_k, \tilde{t}_{i+1}) - \mathcal{I}_{v_{i-1}}(\tilde{t}_k, \tilde{t}_i) + L_{v_i}(\tilde{t}_k) - L_{v_{i-1}}(\tilde{t}_k)\\
&\quad - \Gamma_{v_i}^{v_{i+1}}(\tilde{t}_{i+1}) + \Gamma_{v_{i-1}}^{v_i}(\tilde{t}_i) - (s-\gamma)\kappa + 1 + 3\varepsilon.
\end{aligned}$$

According to Line 5 and Line 17 of the algorithm, the demand Δ_{i+1} received at time t_i from v_{i+1} is reduced by $\Gamma_{v_i}^{v_{i+1}}(t_i)$ and $R_{v_i}(t_i)$ immediately. All clock raises in the time interval $(t_i, \tilde{t}_i]$ further reduce the stored demand. Since the contribution of the hardware clock rate to $\mathcal{I}_{v_i}(t_i, \tilde{t}_i)$ is at most $2\varepsilon(\tilde{t}_i - t_i)$, the clock is raised in the interval $(t_i, \tilde{t}_i]$ by at least $\mathcal{I}_{v_i}(t_i, \tilde{t}_i) - 2\varepsilon(\tilde{t}_i - t_i)$.

Moreover, the demand is constantly reduced in this interval at the rate $\alpha h_{v_i}(\cdot)$, i.e., it is further reduced by $\alpha \int_{t_i}^{\tilde{t}_i} h_i(\tau)d\tau \geq \alpha(1-\varepsilon)(\tilde{t}_i - t_i) \geq 2\varepsilon(\tilde{t}_i - t_i)$. Thus, in total the demand is reduced by at least $\mathcal{I}_{v_i}(t_i, \tilde{t}_i) + \Gamma_{v_i}^{v_{i+1}}(t_i) + R_{v_i}(t_i)$, leading to Inequality (23). Since node v_i sends an order to v_{i-1} at time $\tilde{t}_i$, it must have also received a message from v_{i-1} at this time. Let t' denote the time when v_{i-1} sent this message. We have that $\Lambda_{v_i}^{v_{i-1}}(\tilde{t}_i) = L_{v_i}(\tilde{t}_i) - L_{v_{i-1}}(t')$. In the time interval $(t', \tilde{t}_i]$, v_{i-1} raises its clock by exactly $\Gamma_{v_{i-1}}^{v_i}(\tilde{t}_i)$ and the clock value further increases due to the hardware clock rate by at most $1+\varepsilon$ because $\tilde{t}_i - t' \leq 1$. We get that $L_{v_{i-1}}(t') + \Gamma_{v_{i-1}}^{v_i}(\tilde{t}_i) + 1 + \varepsilon \geq L_{v_{i-1}}(\tilde{t}_i)$, which immediately yields that $\Lambda_{v_i}^{v_{i-1}}(\tilde{t}_i) \leq L_{v_i}(\tilde{t}_i) - L_{v_{i-1}}(\tilde{t}_i) + (1+\varepsilon) + \Gamma_{v_{i-1}}^{v_i}(\tilde{t}_i)$, and thus Inequality (24) follows.

Finally, we get Inequality (25) by using the fact that

$$\Gamma_{v_i}^{v_{i+1}}(t_i) \geq \Gamma_{v_i}^{v_{i+1}}(\tilde{t}_{i+1}) + \mathcal{I}_{v_i}(\tilde{t}_{i+1}, t_i) - R_{v_i}(t_i) - 2\varepsilon. \tag{26}$$

Note that by definition v_i does not receive any message from node v_{i+1} in the time interval $(\tilde{t}_{i+1}, t_i)$, as the message from v_{i+1} sent to v_i at time $\tilde{t}_{i+1}$ is in transit until t_i, i.e., $\Gamma_{v_i}^{v_{i+1}}(t_i) - \Gamma_{v_i}^{v_{i+1}}(\tilde{t}_{i+1})$ is exactly the sum of all clock raises in this open interval. Since the maximum contribution of the hardware clock rate to $\mathcal{I}_{v_i}(\tilde{t}_{i+1}, t_i)$ is 2ε, the sum of all clock raises in the interval $(\tilde{t}_{i+1}, t_i)$ is at least $\mathcal{I}_{v_i}(\tilde{t}_{i+1}, t_i) - R_{v_i}(t_i) - 2\varepsilon$, proving Inequality (26).

Given that v_k does not store any demand on level s, we can bound the demand Δ_k that v_k sends at time $\tilde{t}_k$ by using the same bound on $\Lambda_{v_i}^{v_{i-1}}(\tilde{t}_i)$ as for Inequality (24), this time for $i = k$:

$$\Delta_k = \Lambda_{v_k}^{v_{k-1}}(\tilde{t}_k) - (s-\gamma)\kappa \leq L_{v_k}(\tilde{t}_k) - L_{v_{k-1}}(\tilde{t}_k) + \Gamma_{v_{k-1}}^{v_k}(\tilde{t}_k) - (s-\gamma)\kappa + 1 + \varepsilon.$$

Together with the recursive bound on each Δ_i, we get that

$$\begin{aligned} \Delta_1 &\leq L_{v_k}(\tilde{t}_k) - L_{v_0}(\tilde{t}_k) - ((s-\gamma)\kappa - 1 - 3\varepsilon)k - \mathcal{I}_{v_0}(\tilde{t}_k, \tilde{t}_1) + \Gamma_{v_0}^{v_1}(\tilde{t}_1) - 2\varepsilon \\ &\leq L_{v_k}(\tilde{t}_k) - L_{v_0}(\tilde{t}_k) - ((s-\gamma)\kappa - 1 - 3\varepsilon)k - \mathcal{I}_{v_0}(\tilde{t}_k, t_0) + \Gamma_{v_0}^{v_1}(t_0) + R_{v_0}(t_0), \end{aligned} \tag{27}$$

where Inequality (27) follows directly from Inequality (26). Node v_0 reduces the received demand Δ_1 by $\Gamma_{v_0}^{v_1}(t_0)$ and $R_{v_0}(t_0)$, which yields the desired bound on the stored demand Δ at time t_0:

$$\Delta \leq L_{v_k}(\tilde{t}_k) - L_{v_0}(\tilde{t}_k) - ((s-\gamma)\kappa - 1 - 3\varepsilon)k - \mathcal{I}_{v_0}(\tilde{t}_k, t_0).$$

□

Remark 5.6 *The obtained inequality is in fact slightly more general: Suppose at time t_0 the local variable Δ_v^s is set to $\Delta_1 - \Gamma_v^w(t_0)$ in Line 5 of the algorithm, i.e., after subtracting the total amount of clock raises $\Gamma_v^w(t_0)$ since $v = v_0$ last sent a message to $w = v_1$, the received value Δ_1 is still at least as large as the currently stored demand Δ_v^s. Then the r.h.s. of Inequality (22) is an upper bound on the value $\Delta_v^s - R$ in Line 17 of the algorithm. This follows directly from the proof of the lemma, as we need precisely the assumption that $\Delta_1 - \Gamma_v^w(t_0) \geq \Delta_v^s$ in order to be able to substract $\Gamma_{v_0}^{v_1}(t_0) = \Gamma_v^w$ and $R_{v_0}(t_0) = R$ from the upper bound on Δ_1 in Inequality (27), which immediately yields the bound on $\Delta_v^s - R$ in Line 17. The same reasoning applies to the next lemma, since its proof is based on applications of Lemma 5.5. We will need this fact to prove Corollary 5.8.*

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 5.7 *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* $\text{order}(s, \Delta_1)$ *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 Outline. 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.

Proof. We claim that for any path $v_0, \ldots, v_k$ and all times t and t', where $t \leq t'$, the inequality

$$L_{v_k}(t') - L_{v_0}(t') - k(s-1+\lambda)\kappa \leq \kappa C_{s-1} - \mathcal{I}_{v_0}(t, t') + 2\varepsilon(t' - t) \tag{28}$$

holds. If $k \in [C_r, C_{r-1})$, for some $r \in \mathbb{N}$, we have by definition of C_r that

$$L_{v_k}(t) - L_{v_0}(t) \leq k(r+\lambda)\kappa$$

independent of t. Thus, using that $k < C_{r-1} = \beta^{s-r} C_{s-1}$, we get the bound

$$L_{v_k}(t) - L_{v_0}(t) - k(s-1+\lambda)\kappa < \beta^{s-r} C_{s-1}(r-s+1)\kappa.$$

Due to the assumption that $\beta \geq 2$ this function attains its maximum at $r = s$, hence Inequality (28) is established for $t = t'$.[8]

Now assume that Inequality (28) is violated for some times $t < t'$ and a path $v_0, \ldots, v_k$. The clock skew between v_0 and v_k can only increase by at most $2\varepsilon(t'-t)$ by means of hardware clocks in the interval $(t, t']$. However, this term also appears on the right hand side, hence it follows that a clock *raise* must cause the inequality to be false, implying that there is a specific time when it is violated first. We may therefore choose t' to be exactly that time. If Inequality (28) does not hold on several paths at time t', we consider a path $v_0, \ldots, v_k$ that violates it the most, i.e., the path $v_0, \ldots, v_k$ maximizes ξ, where

$$\xi := L_{v_k}(t') - L_{v_0}(t') - k(s-1+\lambda)\kappa - \big(\kappa C_{s-1} - \mathcal{I}_{v_0}(t, t') + 2\varepsilon(t'-t)\big) > 0.$$

In case several paths attain the same maximum ξ, we choose a path of maximum length k.[9] In the absence of demand on level s or higher, node v_k will not raise its clock above $L_{v_{k-1}}(t') + (s-1)\kappa$. However, this implies for the path $v_0, \ldots, v_{k-1}$ that

$$\begin{aligned} L_{v_{k-1}}(t') - L_{v_0}(t') - (k-1)(s-1+\lambda)\kappa &\geq L_{v_k}(t') - L_{v_0}(t') - k(s-1+\lambda)\kappa + \lambda\kappa \\ &= \xi + \lambda\kappa > \xi, \end{aligned}$$

contradicting the assumption that the path $v_0, \ldots, v_k$ violates Inequality (28) the most. Hence, node v_k must raise its clock due to demand on level s or higher.

[8] Note that $\mathcal{I}_{v_0}(t, t') = 0$ for $t = t'$.

[9] The bound on the global skew proven in Theorem 5.2 gives a trivial upper bound for k.

Thus, v_k must have received a message order(s', Δ_k) at some time t_k, where $t_k \leq t'$ and $s' \geq s$, increasing its demand on level s'. We apply Lemma 5.5, yielding a path $v_k, \ldots, v_l$ and a time $\tilde{t}_l \leq t_k \leq t'$, such that the demand Δ that v_k stores due to the order is bounded by

$$\begin{aligned}
\Delta &\leq L_{v_l}(\tilde{t}_l) - L_{v_k}(\tilde{t}_l) - (l-k)((s'-\gamma)\kappa - 1 - 3\varepsilon) - \mathcal{I}_{v_k}(\tilde{t}_l, t_k) \\
&\leq L_{v_l}(\tilde{t}_l) - L_{v_k}(\tilde{t}_l) - (l-k)(s'-1+\lambda)\kappa - \mathcal{I}_{v_k}(\tilde{t}_l, t_k) \qquad (29) \\
&\leq L_{v_l}(t_k) - L_{v_k}(t_k) - (l-k)(s-1+\lambda)\kappa + \mathcal{I}_{v_l}(t_k, t'). \qquad (30)
\end{aligned}$$

In Inequality (29) we used the assumption that $\lambda + \gamma + \frac{1+3\varepsilon}{\kappa} \leq 1$.

Recall that the stored demand is reduced by at least $\mathcal{I}_{v_k}(t_k, t')$ in the interval $(t_k, t']$, as the permanent decrease of Δ at the rate $\alpha h_{v_k}(\cdot)$ at least counterbalances the entire contribution of the hardware clock to $\mathcal{I}_{v_k}(t_k, t')$. Since v_k raises its clock at time t' due to demand on level s or higher, we therefore must have that $\Delta \geq \mathcal{I}_{v_k}(t_k, t')$. This observation together with Inequality (30) implies that

$$\mathcal{I}_{v_k}(t_k, t') \leq L_{v_l}(t_k) - L_{v_k}(t_k) - (s-1+\lambda)\kappa(l-k) + \mathcal{I}_{v_l}(t_k, t').$$

Hence, we have that $L_{v_l}(t') - L_{v_k}(t') \geq (s-1+\lambda)\kappa(l-k)$. From this inequality it follows that

$$\begin{aligned}
L_{v_l}(t') - L_{v_0}(t') - l(s-1+\lambda)\kappa &= L_{v_l}(t') - L_{v_k}(t') + L_{v_k}(t') - L_{v_0}(t') - l(s-1+\lambda)\kappa \\
&\geq L_{v_k}(t') - L_{v_0}(t') - k(s-1+\lambda)\kappa = \xi.
\end{aligned}$$

Thus, we get a concatenated path $v_0, \ldots, v_k, \ldots, v_l$ of length $l > k$ which also violates Inequality (28) by ξ at time t', contradicting the assumption that $v_0, \ldots, v_k$ is a path violating Inequality (28) by ξ of maximum length. Hence, Inequality (28) holds for all paths and all times t and t'.

Applying Lemma 5.5 once more, this time for the message received by node v_0 at time t_0, we get a path $v_0, \ldots, v_k$ and a time $\tilde{t}_k \leq t_0$ such that the stored demand Δ is upper bounded by

$$\Delta \leq L_{v_k}(\tilde{t}_k) - L_{v_0}(\tilde{t}_k) - ((s-\gamma)\kappa - 1 - 3\varepsilon)k - \mathcal{I}_{v_0}(\tilde{t}_k, t_0).$$

Inequality (28) combined with the assumption that $\lambda + \gamma + \frac{1+3\varepsilon}{\kappa} \leq 1$ yields the desired estimate

$$\begin{aligned}
\Delta &\leq L_{v_k}(\tilde{t}_k) - L_{v_0}(\tilde{t}_k) - k((s-\gamma)\kappa - 1 - 3\varepsilon) - \mathcal{I}_{v_0}(\tilde{t}_k, t_0) + \mathcal{I}_{v_k}(\tilde{t}_k, t_0) \\
&= L_{v_k}(t_0) - L_{v_0}(t_0) - k((s-\gamma)\kappa - 1 - 3\varepsilon) \\
&\leq \kappa C_{s-1} + k\left(\lambda + \gamma + \frac{1+3\varepsilon}{\kappa} - 1\right)\kappa - \mathcal{I}_{v_0}(t, t_0) + 2\varepsilon(t_0 - t) \\
&\leq \kappa C_{s-1} - \mathcal{I}_{v_0}(t, t_0) + 2\varepsilon(t_0 - t).
\end{aligned}$$

□

As a direct consequence of Lemma 5.5 and Lemma 5.7, we get that nodes will have no demand if they increased their clocks by a specific amount in a preceding time period.

Corollary 5.8 *Suppose for some node v we have $\mathcal{I}_v(t, t') > \kappa C_{s-1} + 2\varepsilon(t' - t)$, where $t' \geq t$, and the prerequisites of Lemma 5.7 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.*

Proof. First, we prove Statement (1). Assume that node v stores a positive demand $\Delta > 0$ on level $s' \geq s$ at time t'. Denote by $t_0 \leq t'$ the time when v received the last message (s', Δ_1) increasing its demand on level s'. By Lemma 5.7, the demand stored at this time was bounded by $\kappa C_{s'-1} - \mathcal{I}_v(t, t_0) + 2\varepsilon(t_0 - t)$. As t_0 was the last time when the demand of v was increased,

it has been positive in the time interval $(t_0, t']$. Hence, it has been reduced by all clock raises that occurred in this interval. As the contribution of the hardware clock to $\mathcal{I}_v(t_0, t')$ is bounded by $2\varepsilon(t' - t_0)$, it holds that

$$\begin{aligned} \Delta &\leq \kappa C_{s'-1} - \mathcal{I}_v(t, t_0) + 2\varepsilon(t_0 - t) - (\mathcal{I}_v(t_0, t') - 2\varepsilon(t' - t_0)) \\ &\leq \kappa C_{s-1} - \mathcal{I}_v(t, t') + 2\varepsilon(t' - t) < 0, \end{aligned} \tag{31}$$

a contradiction.

In order to prove Statement (2), assume that v raises its clock by $R_v(t') > 0$ at time t' and that v has a neighbor w such that $L_v(t') > L_w(t') + (s-1)\kappa$. The clock value of w is as least as large as when w sent the last message to v, thus we can bound

$$\Lambda_v^w(t') - (s-1)\kappa \geq L_v(t') - L_w(t') - (s-1)\kappa > 0. \tag{32}$$

Since $s > 1$, R is set to a negative value in Line 7 of the algorithm. Line 10 does not increase R above zero, hence the clock raise of v at time t' must be due to Line 9. The computed minimum in Line 9 is negative for all $s' < s$ because of Inequality (32). Consequently, an $s' \geq s$ must exist such that $\Delta_v^{s'} \geq R_v(t') > 0$, allowing R to be set to a positive value. Hence we have $\Delta_v^{s'} - R \geq 0$ in Line 17 of the algorithm. However, as pointed out in Remark 5.6, Lemma 5.7 also applies to $\Delta_v^{s'} - R$. Thus, the r.h.s. of Inequality (31) is an upper bound on $\Delta_v^{s'} - R$ as well. Hence $\Delta_v^{s'} - R$ must also be strictly negative, a contradiction. □

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 (36)-(40) are required in the main theorem itself.

Condition 5.9 *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*

$$\alpha \geq \frac{2\varepsilon}{1-\varepsilon} \tag{33}$$

$$\gamma\kappa \geq 1 - \varepsilon + 2(1+\varepsilon)\alpha \tag{34}$$

$$\lambda + \gamma + \frac{1+3\varepsilon}{\kappa} \leq 1 \tag{35}$$

$$\left(\beta^{-1} + \beta^{2-m}\right)(1+\lambda) \leq \lambda - \frac{1}{2c} \tag{36}$$

$$\left(1 - \lambda + \frac{1}{c}\right)\beta \leq \ell \tag{37}$$

$$\beta^m \leq \frac{\kappa}{24\varepsilon} \tag{38}$$

$$\beta^{\ell+1} \leq \frac{\kappa}{24\varepsilon c} \tag{39}$$

$$\log_\beta\left(\frac{1+5\varepsilon}{(\ell+\lambda)\kappa}D\right) > m - l - 1 \tag{40}$$

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

Remark 5.10 *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.*[10] *A corresponding admissable choice of constants is given by $\lambda := \frac{4}{7}$, $\beta := 4$, $\ell := 2$ $m := 5$, and $c := 14$.*

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

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

Theorem 5.11 *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 Outline. In a first step, we show that the bound on the local skew can never be violated in a legal state. If we assume the contrary, no demand above the reachable levels can exist until the first time of violation. Hence, at least the last fraction of $\frac{\kappa}{2}$ of the clock skew must be built up by hardware clocks, which takes at least $\frac{\kappa}{4\varepsilon}$ time units. However, due to Lemma 5.4, within this time the slower node increases its clock by at least $\frac{\kappa}{2}$, showing that the bound can never be violated.

In a second step, we prove that the network remains in a legal state at all times. Assuming the contrary again, until the first time of violation all lemmas are applicable. From the first part we get that the first violation must occur on a path $v_0,\dots,v_k$ and a level s such that $k>C_s\geq\beta^{m-2}$, and the established bound on the global skew implies $s>\ell$. Since initially no skew is present in the system, there must be a last point in time when the average skew on the path $v_0,\dots,v_k$ did not (significantly) exceed $(s-\ell)\kappa\geq\kappa$. After some computations relying on $k>\beta^{m-2}$, we conclude that by the time an average skew of $s\kappa$ is exceeded by a specific amount, all except the last $\lceil C_{s+1}\rceil$ nodes of the path meet the prerequisites of Corollary 5.8, and thus they will not raise their clocks more than $s\kappa$ above their neighbors'. The skew on the remaining part of the path can be controlled by the legal state conditions for $s+1$, implying that still a notable remaining portion of the skew necessary to violate the legal state conditions must be built up by hardware clocks. However, due to Lemma 5.4, in the time it takes to build up this skew, v_k will increase its clock quickly enough to reduce the skew by at least the same amount. This argument can be repeated to show that any skew built up by hardware clock rates will be reduced again before the claimed bound can be reached. Thus, we can conclude that there is no first time when the the skew needed to violate the legal state is reached, implying that the network always remains in a legal state.

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 2 due to Condition (40).

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.[11] 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

[11] 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.

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 (38), 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 $v_0, \dots, v_k$ implies that $k < \frac{1+5\varepsilon}{(s+\lambda)\kappa}D$. By Condition (36), we can bound $\beta^{\ell-s} \geq \left(\frac{1+\lambda}{\lambda}\right)^{\ell-s} \geq \frac{\ell+\lambda}{s+\lambda}$ for $s \in \{0,\dots,\ell\}$. 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 $s \in \{0,\dots,\ell\}$. Hence it follows that a path $v_0,\dots,v_k$ 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

$$L_{v_i}(t) - L_{v_k}(t) \leq (k-i)(s-\ell)\kappa + \frac{\kappa}{2c}k \tag{41}$$

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

$$L_{v_i}(t) - L_{v_k}(t) \leq (k-i)s\kappa + \left(\lambda - \frac{1}{2c}\right)\kappa k + 2\varepsilon(t-t_0), \tag{42}$$

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 $L_{v_i}(t) - L_{v_k}(t) \leq (k-i)(s+1+\lambda)\kappa$. Condition (36) and the observation that $C_s \geq \beta^{m-2}$ allow us to bound

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

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

Now assume that Inequality (42) is true for some node v_i, where $i \leq k - \lceil C_{s+1}\rceil$. 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 $t' \geq t_0$ when

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

Note that Inequality (42) 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 (41) might be violated at time t_0, we consider a time t shortly before t_0. If we choose $t_0 - t > 0$ sufficiently small, by definition of t_0 and Condition (37) we get that

$$\begin{aligned} \mathcal{I}_{v_{i-1}}(t,t') &> (k-(i-1))\ell\kappa + \left(\lambda - \frac{1}{c}\right)\kappa C_s + 2\varepsilon(t'-t) + \mathcal{I}_{v_k}(t,t') \\ &\geq \left(\beta^{-1}\ell + \lambda - \frac{1}{c}\right)\kappa C_s + 2\varepsilon(t'-t) \\ &\geq \kappa C_s + 2\varepsilon(t'-t). \end{aligned} \tag{43}$$

Since v_{i-1} raises its clock at time t' and given the bound on $\mathcal{I}_{v_{i-1}}(t,t')$, applying Corollary 5.8 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

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

contradicting the assumption that Inequality (42) holds for node v_i at all times $t \geq t_0$. 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 (42), we conclude that

$$(s+\lambda)\kappa k \leq L_{v_0}(t_{\max}) - L_{v_k}(t_{\max}) \leq \left(s + \lambda - \frac{1}{2c}\right)\kappa k + 2\varepsilon(t_{\max} - t_0), \tag{44}$$

implying that $t_{\max} - t_0 \geq \frac{\kappa}{4\varepsilon c}k \geq \frac{\kappa}{4\varepsilon c}C_s$. As Inequality (41) is violated at time t_0, Lemma 5.4 and Condition (39) yield that for time

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

it holds that $\mathcal{I}_{v_k}(t_0,t_1) \geq \frac{\kappa}{2c}k \geq \frac{\kappa}{2c}C_s \geq 2\varepsilon(t_1 - t_0)$, showing that $t_{\max} > t_1$. Furthermore, we can now improve Inequality (42) for times $t \geq t_1$ to

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

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 (43), which is compensated for by the lower bound on $\mathcal{I}_{v_k}(t_0,t_1)$. Thus Inequality (44) sharpens to

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

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 $t_{\max} \geq t_2 := t_1 + 6C_{s-\ell-1}$. Since, by definition of t_0, Inequality (41) is also violated at time $t_1 > t_0$, Lemma 5.4 yields that again $\mathcal{I}_{v_k}(t_1,t_2) \geq \frac{\kappa}{2c}C_s \geq 2\varepsilon(t_2 - t_1)$. 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 5.12 *We will now briefly discuss the special case of a network with a bounded diameter D, where $D < \frac{\kappa^2}{24\varepsilon}$. Set $\ell = 1$, $\lambda = 5\varepsilon$, $m := \left\lceil \log_\beta \frac{D}{\kappa} \right\rceil + 1$ and choose $\beta > 1$ small enough such that $D \leq \frac{\kappa^2}{24\beta^2\varepsilon}$. Hence, Condition* (38) *is fulfilled. Note that Lemma 5.4 holds for $s = 1$ even without Conditions* (34) *and* (35)*, 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* $\kappa \geq 2$ *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 5.13 *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. *Information Processing Letters*, 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] 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)*, 2008.

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

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

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

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

A Notation

Table 1: Quantities defined by the problem.

Variable	Interpretation	Known to the algorithm
h_v	Hardware clock rate of node v	No, but $h_v(t) \in [1-\varepsilon, 1+\varepsilon]$
H_v	Hardware clock value of node v ($H_v(t) = \int_0^t h_v(\tau)\ d\tau$)	Yes
ε	Bound on the relative clock drift	No, but bounded by a constant smaller than 1
D	Network diameter	No
$\mathcal{T}$	Bound on the message delay (normalized to 1)	No

Table 2: Parameters of the algorithm.

Variable	Interpretation	Comment
κ	Base unit defining the "skew per level"	Preferably small, but at least $2\mathcal{T}$
γ	Fraction of κ added to demands when sending orders	Roughly $\mathcal{T}/\kappa$
α	Demands are reduced at rate $\alpha h_v(t) \geq 2\varepsilon$	Set $\alpha := \frac{2\hat{\varepsilon}}{1-\hat{\varepsilon}}$ for any estimate $\varepsilon \leq \hat{\varepsilon} < 1$

Table 3: Local variables of the algorithm.

Variable	Interpretation	Comment
L_v	Logical clock value of v	Increased at rate h_v (via H_v), raised by R_v when a message is processed
Λ_v^w	Estimated clock skew to node w	Assumed to remain constant while no messages are processed
$L_v^{\max}$	Node v's estimate of the maximum clock value	Increased at rate h_v
Δ_v^s	Current demand on level s	Decreased at rate αh_v and by any clock raise
Γ_v^w	Sum of clock raises since the last message to w was sent	Used to correct the received demands from w, subsequently reset to zero

Table 4: Variables used solely in the analysis.

Variable	Interpretation	Comment
C_s	Bound on the length of a path with an average skew of more than $(s+\lambda)\kappa$ skew	Falls exponentially: $C_{s+1} = \beta^{-1}C_s$ for $\beta \geq 2$
β	See above	Base of the logarithm in the bound on the local skew
λ	Non-integer variable, see above	Allows to amortize effects of demands from higher levels
ℓ	The proof of the main theorem argues over ℓ levels	Has a considerable impact on the constants
m	The top $m+1$ levels s with $C_s \geq 1$ are never reached	These levels need to be handled differently in the proof, since C_s is small
c	Parameter used in the proof of the main theorem	Ensures that a fraction of the skew necessary to leave the legal state must be built up per hardware clocks, leaving time to react