Tight Bounds for Clock Synchronization

CHRISTOPH LENZEN, THOMAS LOCHER, AND ROGER WATTENHOFER

ETH Zurich, Zurich, Switzerland

Abstract. We present a novel clock synchronization algorithm and prove tight upper and lower bounds on the worst-case clock skew that may occur between any two participants in any given distributed system. More importantly, the worst-case clock skew between neighboring nodes is (asymptotically) at most a factor of two larger than the best possible bound. While previous results solely focused on the dependency of the skew bounds on the network diameter, we prove that our techniques are optimal also with respect to the maximum clock drift, the uncertainty in message delays, and the imposed bounds on the clock rates. The presented results all hold in a general model where both the clock drifts and the message delays may vary arbitrarily within pre-specified bounds.

Furthermore, our algorithm exhibits a number of other highly desirable properties. First, the algorithm ensures that the clock values remain in an affine linear envelope of real time. A better worst-case bound on the accuracy with respect to real time cannot be achieved in the absence of an external timer. Second, the algorithm minimizes the number and size of messages that need to be exchanged in a given time period. Moreover, only a small number of bits must be stored locally for each neighbor. Finally, our algorithm can easily be adapted for a variety of other prominent synchronization models.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Bounded rates, global skew, gradient property, local skew, variable clock drifts, worst-case analysis

ACM Reference Format:

Lenzen, C., Locher, T., and Wattenhofer, R. 2010. Tight bounds for clock synchronization. J. ACM 57, 2, Article 8, (January 2010), 42 pages.
DOI = 10.1145/1667053.1667057 http://doi.acm.org/10.1145/1667053.1667057

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

---

Preliminary results of this article were published in Lenzen et al. [2008, 2009a].

Authors' address: Computer Engineering and Networks Laboratory (TIK), ETH Zurich, 8092 Zurich, Switzerland, e-mail: {lenzen;lochert;wattenhofer}@tik.ee.ethz.ch

DOI 10.1145/1667053.1667057 http://doi.acm.org/10.1145/1667053.1667057

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 neighbors. Since it is reasonable to expect that events occurring 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 within certain bounds. Moreover, the nodes do not have access to a source of real time (e.g., by means of a GPS receiver) that could provide the nodes with real time information once in a while.

Naturally, one objective is to minimize the skew between any two nodes in the graph, regardless of the distance in G between them. We call the maximum worst-case skew between any two nodes in the graph the *global skew*. Apart from minimizing the global skew, it is essential for several distributed applications that the clock skew between neighboring nodes is as small as possible. One could even think of applications where the global skew is not of great concern, but any node only needs to be well synchronized with nodes in its vicinity. This is the case if occurrences of events are only of local importance and do not bear any (immediate) significance for nodes that are not close-by.[1] The so-called *gradient property*, which has been introduced in Fan and Lynch [2004], captures this optimization criterion. It requires that the clock skew between any two nodes v, w is bounded by a monotonically increasing function of their distance $d(v, w)$. Thus, neighboring nodes should always be well synchronized, 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 smallest possible global skew that any algorithm can achieve is $D/2$ [Biaz and Lundelius Welch 2001], where D denotes the diameter of the graph.[2] As far as the local skew is concerned, it has been proved in the surprising work by Fan and Lynch [2004] 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. At the time when Fan and Lynch [2004] presented their lower bound, no algorithm guaranteeing a local skew sublinear in the diameter was known. In the meantime, the upper bound on the local skew has been improved to $\mathcal{O}(\sqrt{D})$ [Locher and Wattenhofer 2006] and subsequently to $\mathcal{O}(\log D)$ [Lenzen et al. 2008]. However, both the upper and

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

[2] For ease of presentation, we normalize the uncertainty in message delays to one in this introduction and the related work section as all bounds depend linearly on it.

the lower bounds so far neglected the influence of other parameters such as the maximum clock drift rate, leaving room for improvements not visible when considering only the network diameter.

The objective of this work is to provide tight bounds on the degree of synchronization that can be achieved, taking many parameters such as the delay uncertainty and the maximum clock drift rate into account. In other words, we show how the bounds on the worst-case skews depend on all of these parameters. Another aspect of clock synchronization, which so far has not received much attention, is that a practical clock synchronization algorithm must ensure that the rates of progress of all logical clock values are always within specific bounds, that is, the clock values are not allowed to change substantially in a short time.[3] However, bounding the clock rates inhibits the ability of an algorithm to react to clock skews, which have to be kept as small as possible. Apart from bounding the logical clock rates in order to ensure that the clock values do not change abruptly, it may further be desirable to keep the progress of the logical clock values as close to the progress of real time as possible, that is, an algorithm should guarantee the best possible real-time approximation in the absence of an external timer.

We propose an algorithm that bounds the minimum and maximum progress of the logical clocks and ensures that the logical clock values always remain within an affine linear envelope of real time. We prove matching bounds on both the global and the local skew of this algorithm. What is more, we show that the message frequency can be kept quite low without increasing the worst-case clock differences significantly, which implies that techniques such as piggybacking can be employed. This is a viable option especially considering that we only require a few bits to be sent in each message, which can be included in (or appended to) any message sent by another application. To round off our analysis, we discuss how to adapt our algorithm to other synchronization models, for example, external synchronization where a source of real time is available to some of the participants in the system. Our results imply that the techniques in this work offer asymptotically optimal solutions to the synchronization problem with respect to several optimization criteria for a wide range of models.

The remainder of this article is organized as follows. After reviewing related work in Section 2, the formal model used throughout this work is presented in Section 3. The proposed algorithm is described and analyzed in Section 4 and Section 5, respectively. The message, bit, and space complexities of the algorithm are discussed in Section 6. In Section 7, we show that the algorithm is asymptotically optimal by proving matching lower bounds on both the global and the local skew. The applicability of our algorithm to other model assumptions is discussed in Section 8. Finally, Section 9 concludes the article.

2. *Related Work*

There is a large body of work on the fundamental problem of synchronizing clocks in distributed systems. Most work mainly focuses on bounding the skew that may occur between any two clocks and also the communication costs that are required in

[3] For instance, if velocities are to be determined with the help of local clocks, clock jumps would severely deteriorate the accuracy of the measurements.

order to guarantee a certain degree of synchronization (see, e.g., Lundelius Welch and Lynch [1984], Ostrovsky and Patt-Shamir [1999], Patt-Shamir and Rajsbaum [1994], and Srikanth and Toueg [1987]). It has been shown that a skew of $D/2$ cannot be avoided on any graph G of diameter D [Biaz and Lundelius Welch 2001]. We will prove a stronger lower bound of roughly D for clock synchronization algorithms that strive to keep all clock values within a linear envelope of real time. The clock synchronization algorithm by Srikanth and Toueg [1987] achieves a bound of $\mathcal{O}(D)$ on the skew of any two clocks at all times and is thus asymptotically optimal. The authors further show that the accuracy of their algorithm with respect to real time is also optimal as all clocks are always within a linear envelope of real time. However, their algorithm incurs a skew of $\Theta(D)$ between neighboring nodes in the worst case.

In their seminal work that introduced the problem of synchronizing clocks of close nodes as accurately as possible, Fan and Lynch [2004] showed that no algorithm can avoid a clock skew of $\Omega(\log_b D)$ between neighboring nodes, where $b \in \mathcal{O}(\log D)$. The only imposed constraint is that nodes are required to increase their clock values at a given minimum progress rate, which is quite natural as otherwise events that occur at different (real) times may happen at the same logical time because an algorithm could simply halt the clocks. Subsequently, it has been shown that this bound also holds if all messages arrive instantaneously, but an adversary can determine when synchronization messages may be sent [Meier and Thiele 2005]. However, Fan and Lynch treated the maximum hardware clock drift ε as a constant; taking it into account as a parameter reveals that $b \in \mathcal{O}((\log D)/\varepsilon)$. We improve their result to $b \in \Theta(1/\varepsilon)$, that is, any algorithm may experience a local skew of $\Omega(\log_{1/\varepsilon} D)$ [Lenzen et al. 2009a]. Furthermore, we show that if clock rates are upper bounded by a constant, b depends linearly on the difference of the maximum and the minimum rate. In particular, if logical clocks must guarantee an optimal drift of $\mathcal{O}(\varepsilon)$, the bound on the local skew becomes $\Theta(\log D)$ [Lenzen et al. 2009a].

There has also been a lot of (more practical) work on clock synchronization for specific computing environments. For example, the clock synchronization problem in wireless sensor networks has been extensively studied [Elson et al. 2002; Ganeriwal et al. 2003; Maróti et al. 2004; PalChaudhuri et al. 2004]. It can be argued that in such systems message delays are not only bounded, but also distributed (independently) at random. This assumption constitutes a far-reaching difference to the worst-case scenario as it impacts the achievable skew bounds. Recently, it has been shown that in the respective model the global skew can be upper bounded by $\tilde{\mathcal{O}}(\sqrt{D})$ with high probability [Lenzen et al. 2009b]. The same work proves that on most graphs at any point in time there is a constant probability that a clock skew of $\Omega(\sqrt{D})$ can be observed between some nodes. Synchronizing clocks is also an important issue for other forms of distributed systems such as the Internet [Mills 1991] or systems-on-a-chip [Függer et al. 2006]. However, apart from processor design, where one seeks to control signal delays by means of placement and wiring (see, e.g., Korte et al. [2007] and references therein), synchronizing close-by devices particularly well has scarcely been considered as an optimization criterion. In fact, to the best of our knowledge, the only attempt has been made in the context of sensor networks [Sommer and Wattenhofer 2009].

The first algorithm guaranteeing a sublinear bound on the clock skew between neighboring nodes achieved a bound of $\mathcal{O}(\sqrt{\varepsilon D})$ [Locher 2009; Locher and Wattenhofer 2006], a result that has recently been generalized to dynamic settings [Kuhn et al. 2009]. For static distributed systems, the upper bound has been improved to $\mathcal{O}(\log D)$ [Lenzen et al. 2008]. However, the algorithm that guarantees a logarithmic bound has several disadvantages: Apart from being quite complicated, the algorithm has the undesirable property that the clock values can "jump" by $\Theta(\log D)$ in a single time step, and thus the clock values may not change smoothly. Moreover, both the message frequency and the size of the messages are fairly large, which prohibits techniques such as piggybacking and which may imply that the algorithm is not useful in practice. What is more, the base of the logarithm can hardly be increased if ε becomes small. Since typically $\varepsilon \ll 1$, a notable gap to the lower bound remains. The algorithm presented in this article does not have these shortcomings [Lenzen et al. 2009a].

3. *Model*

We model a distributed system as a connected, undirected graph $G = (V, E)$ of diameter D, where nodes represent computational devices and edges represent bidirectional communication links. Each node v can communicate with all neighboring nodes by exchanging messages. The set of v's neighbors is denoted by $\mathcal{N}_v := \{w \in V \mid \{v, w\} \in E\}$. We assume that, for any two nodes $u, w \in \mathcal{N}_v$, node v can distinguish u from w, for example, by means of a port numbering or node identifiers, and also that all communication is reliable, that is, messages are never lost. However, communication takes some time, and this *delay* may vary. In general, a message delay may consist of two parts, a fixed known delay and an additional variable delay. Since any fixed fraction of the total delay can be added to a received clock value, we define it to be zero (we will discuss the impact of this simplification in Section 8.3). Thus, the time that passes from the moment a message is sent until the recipient can act upon it may be any value in the range $[0, \mathcal{T}]$, where $\mathcal{T}$ is the *delay uncertainty*. While the bound $\mathcal{T}$ is unknown to the algorithm, we assume that the nodes know an upper bound $\hat{\mathcal{T}}$ on $\mathcal{T}$, which can easily be obtained by measuring round-trip times (see Section 8.1 for a more detailed discussion).

Each node v is equipped with a *hardware clock* H_v which starts running at the time t_v when v is initialized. The first node starts its clock at real time $t = 0$. An *initialization message* is then flooded through the distributed system in order to start the clocks at the other nodes. We denote the value of the hardware clock at real time t by $H_v(t)$, that is, $H_v : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ is a monotonically increasing function. The value of the hardware clock of v is 0 until time t_v and $H_v(t) := \int_{t_v}^{t} h_v(\tau)\, d\tau$ afterwards, where $h_v(\tau)$ is the *hardware clock rate* of v at time τ. The clock rates may vary over time, but we assume that there is a constant $0 < \varepsilon < 1$ such that the following condition holds.

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

While the exact value of ε is unknown, we assume that the nodes know an upper bound $\hat{\varepsilon}$ that is strictly smaller than one, that is, hardware clocks guarantee a strictly positive minimum progress rate.

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

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

Moreover, we demand that the logical clocks behave normally in the sense that the logical clock values may not change dramatically in a short time. Formally, there are constants $0 < \alpha \leq 1-\varepsilon$ and $\beta \geq 1+\varepsilon$ such that

$$\forall v \in V \; \forall t' \geq t \geq t_v : \; \alpha(t'-t) \leq L_v(t') - L_v(t) \leq \beta(t'-t). \tag{2}$$

The increased (or lowered) clock rates of the logical clocks allow the nodes to correct differences between the logical clock values in the system. The difference between the values of logical clocks is called *clock skew*. Ideally, the logical clocks behave just like the hardware clocks even in the presence of clock skews, albeit with a slightly worse clock drift, that is, $\alpha \in 1-\mathcal{O}(\varepsilon)$ and $\beta \in 1+\mathcal{O}(\varepsilon)$. Note that Condition (2) implies that clocks must always make progress.

A clock synchronization algorithm $\mathcal{A}$ executed at node v specifies how the logical clock $L_v(t)$ of node v is increased based on its hardware clock and the information received from its neighbors up to time t in such a way that Conditions (1) and (2) are satisfied. Given a clock synchronization algorithm $\mathcal{A}$ and a (connected) graph G, an *execution* $\mathcal{E}$ specifies the delays of all messages and also the hardware clock rates of all nodes at each point in time when $\mathcal{A}$ is executed on G. Thus, the information contained in an execution completely determines the state of the system at any time for a run of $\mathcal{A}$ on G. The *global* and the *local skew* are formally defined as follows:

Definition 3.1 (*Global Skew*). Given a connected graph $G = (V, E)$ and a clock synchronization algorithm $\mathcal{A}$, the *global skew* is defined as the value

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

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

Definition 3.2 (*Local Skew*). Given a connected graph $G = (V, E)$ and a clock synchronization algorithm $\mathcal{A}$, the *local skew* is defined as the value

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

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

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

4. *Algorithm*

In this section, we introduce the synchronization algorithm $\mathcal{A}^{\text{opt}}$. After a brief overview of the structure and main concepts of the algorithm, we give a line-by-line description of the pseudocode.

4.1. OVERVIEW. In short, any clock synchronization algorithm needs to perform two tasks: First, the algorithm must gather up-to-date information about other nodes'

clock values, and second, based on these estimates it needs to determine the progress rate of the (local) logical clock.

The algorithm $\mathcal{A}^{\text{opt}}$ ensures that the estimates are up-to-date as follows. Every node periodically announces its current clock value and an estimate of the current maximum clock value in the system to its neighbors. This way, nodes obtain estimates of the neighbors' clock values that were in transit for at most $\mathcal{T}$ time. Note that since an estimate of the maximum clock value may be forwarded through the system, it is possible that up to $D\mathcal{T}$ time passes from the time it was originally sent until a particular node receives it. In an effort to keep the estimates as accurate as possible, all nodes increase the estimates at the rate of their hardware clock between updates. As we will see later, this allows us to increase the time between sending events notably without suffering from a considerable decay in the resulting synchronisation quality.

While it is fairly easy to perform the necessary bookkeeping, it is more challenging to determine the "right" logical clock rate given the current estimates of the neighbors' clock values and the largest clock value. The progress rate of the logical clock is determined by the subroutine *setClockRate()*. It is called whenever a new message from a neighbor has been processed since the message may have caused an update to the local estimates of clock values. The subroutine *setClockRate()* either determines that the logical clock rate must be the same as the hardware clock rate or that the logical clock must increase more quickly than the hardware clock, in which case, the logical clock rate is simply a (predefined) constant times larger than the hardware clock rate. In other words, the logical clock value increases at the same rate as the hardware clock value by default, but the subroutine *setClockRate()* can decide to impose a higher logical clock rate. The main challenge is to determine when the logical clock rate has to be switched to the faster mode (and back); this is discussed in the following section where we describe the algorithm in more detail.

4.2. DETAILED DESCRIPTION. Since naturally a clock synchronization algorithm runs in an asynchronous system, its actions are triggered by events. In our case, an action is triggered if either a message is received (Algorithm 2) or the local hardware clock reaches certain values (Algorithms 1 and 4). The subroutine *setClockRate()* (Algorithm 3) is invoked directly by Algorithm 2 after the newly received message has been processed.

We proceed by examining the algorithm in greater detail. In order to synchronize the logical clocks, any node v must perpetually send synchronization messages informing the neighboring nodes about its current clock value L_v. Node v itself adapts its clock value according to the information received from its neighbors. In order to ensure that nodes indeed inform each other on a regular basis, we impose that messages are sent at least once in time H_0 according to the local hardware clock. Certainly, we want to keep H_0 as large as possible for the purpose of minimizing the number of transmissions. However, information about neighboring clock values only is not sufficient to guarantee an optimal bound on the global skew because the neighboring nodes might have similar clock values while the skew to nodes at greater distances may be large. Naturally, nodes may not increase their clock values over the largest received clock value as such a behavior might violate Condition (1). Thus, we need to spread information about large clock values quickly without increasing the frequency of communication substantially.

This problem can be solved by including an estimate $L_v^{\max}$ of the maximum clock value in each message. This estimate is increased at the (local) hardware clock rate, that is, the nodes assume that the maximum clock value in the system increases at the same rate as the local hardware clock. Nodes send a message to their neighbors exactly when their local estimate $L_v^{\max}$ reaches an integer multiple of H_0. Thus, they send a message $\langle L_v, L_v^{\max}\rangle$ on their own after at most $H_0/(1-\varepsilon)$ real time (Algorithm 1).[4] However, when receiving an estimate larger than their own, they will immediately forward it to their neighbors so that distant nodes obtain this larger estimate as quickly as possible. Since any received estimate must already be an integer multiple of H_0, any node sends only one message for each multiple of H_0. Apart from $L_v^{\max}$, each node v keeps estimates L_v^w of the clock values of its neighbors $w \in \mathcal{N}_v$. In order to decide whether a newly received value from w is more recent and thus more accurate, v keeps track of the largest clock value ℓ_v^w that v has received from w so far. If the received clock value is larger than ℓ_v^w, L_v^w is adapted accordingly. Each node v also increases the estimates L_v^w at the progress rate h_v of its own hardware clock.

Algorithm 1. $L_v^{\max}$ reaches an integer multiple of H_0. Note that $L_v^{\max}$ is increased at rate h_v.

```
1 send ⟨L_v, L_v^max⟩ to all u ∈ N_v
```

Based on this information, v has to decide how to adapt its logical clock. To this end, $\mathcal{A}^{\text{opt}}$ computes the estimates $\Lambda_v^{\uparrow} := \max_{w\in\mathcal{N}_v}\{L_v^w - L_v\}$ and $\Lambda_v^{\downarrow} := \max_{w\in\mathcal{N}_v}\{L_v - L_v^w\}$ of the skew to the clocks in its neighborhood that are ahead and behind the most, respectively. In order to satisfy Condition (2), the clock value L_v cannot be increased by a certain value instantaneously, i.e., $\mathcal{A}^{\text{opt}}$ can only manipulate the logical clock rate. For this purpose, the subroutine *setClockRate* is called, which adapts the logical clock rate whenever new information has been received. The actions that each node v takes upon receiving a message $\langle L_w, L_w^{\max}\rangle$ from a node w are summarized in Algorithm 2.

Algorithm 2. Message $\langle L_w, L_w^{\max}\rangle$ received from a node $w \in \mathcal{N}_v$. Note that L_v^u, $u \in \mathcal{N}_v$, and L_v are increased at the rates h_v and $\rho_v \cdot h_v$, respectively.

```
1 if L_w^max > L_v^max then
2     L_v^max := L_w^max
3     send ⟨L_v, L_v^max⟩ to all u ∈ N_v
4 end
5 if L_w > ℓ_v^w then
6     L_v^w := L_w, ℓ_v^w := L_w
7 end
8 Λ_v^↑ := max_{u∈N_v}{L_v^u − L_v}
9 Λ_v^↓ := max_{u∈N_v}{L_v − L_v^u}
10 setClockRate()
```

The steps of the subroutine *setClockRate*, the key ingredient of algorithm $\mathcal{A}^{\text{opt}}$, are summarized in Algorithm 3. By default, the logical clock runs at the hardware clock rate as well. The subroutine determines if and for how long the logical clock value has to increase more quickly than the hardware clock value (and the local

[4] Recall that all hardware clock rates always lie in the interval $[1-\varepsilon, 1+\varepsilon]$.

variables) as follows. First, the amount R_v by which v would increase L_v if it were allowed to raise its clock value instantaneously is computed. Roughly speaking, the goal of the subroutine is to ensure that the clock skew $\Lambda_v^{\downarrow}$ to the neighbor whose clock is assumed to be behind the most and the clock skew $\Lambda_v^{\uparrow}$ to the neighbor with the largest estimated clock value are the same integer multiple of a parameter $\kappa \in \Omega(\mathcal{T})$. The variable R_v is the largest value that satisfies this constraint. More precisely, if $\Lambda_v^{\uparrow} \leq s\kappa$ and $\Lambda_v^{\downarrow} \geq s\kappa$ for some $s \in \mathbb{N}_0$, then $R_v \leq 0$. Otherwise, $R_v > 0$ is exactly the increase of the clock value that causes this condition to hold for some $s \in \mathbb{N}_0$. Line 1 of Algorithm 3 is a concise formulation of this rule.

Note that the simpler approach that attempts to attain a clock value that lies in the middle between the largest and the smallest (estimated) clock value of the neighboring nodes fails to achieve even a sublinear bound on the local skew [Locher and Wattenhofer 2006]. Although $\mathcal{A}^{\text{opt}}$ also strives to balance the skew to the (estimated) fastest and slowest neighbor's clocks, the following simple example illustrates that the employed strategy is more aggressive than just trying to reach the average of the maximum and the minimum clock estimate: If $\Lambda_v^{\uparrow} = \Lambda_v^{\downarrow} = (s + 1/2)\kappa$ for any $s \in \mathbb{N}_0$, algorithm $\mathcal{A}^{\text{opt}}$ sets R_v to $\kappa/2$, whereas for $\Lambda_v^{\uparrow} \leq \Lambda_v^{\downarrow}$ the simpler strategy mandates that $R_v = 0$. Finally, the value R_v may be at least $\kappa - \Lambda_v^{\downarrow}$ because a skew of κ is always tolerated, but never more than $L_v^{\max} - L_v$, since v must not increase the clock to a value greater than $L_v^{\max}$ (see Line 2 of Algorithm 3).

Algorithm 3. setClockRate(): Adjust the logical clock rate of the clock L_v according to the current information.

1 $R_v := \sup\left\{R \in \mathbb{R} \,\middle|\, \left\lfloor \frac{\Lambda_v^{\uparrow} - R}{\kappa} \right\rfloor \geq \left\lfloor \frac{\Lambda_v^{\downarrow} + R}{\kappa} \right\rfloor\right\}$
2 $R_v := \min\left\{\max\left\{\kappa - \Lambda_v^{\downarrow}, R_v\right\}, L_v^{\max} - L_v\right\}$
3 **if** $R_v > 0$ **then**
4 $\quad \rho_v := 1 + \mu$
5 $\quad H_v^R := H_v + \frac{R_v}{\mu}$
6 **else**
7 $\quad \rho_v := 1$
8 **end**

In order to bound the increase of the logical clock, the algorithm dictates that any node's logical clock value may increase at most $1 + \mu$ times faster than its hardware clock value for a given $\mu > 0$. If the computed increase R_v is positive, v sets its *logical clock rate multiplier* ρ_v to $1 + \mu$, which ensures that $L_v(t_2) - L_v(t_1) = (1 + \mu)(H_v(t_2) - H_v(t_1))$ for any time interval $[t_1, t_2]$ where $\rho_v = 1 + \mu$. The clock rate multiplier ρ_v is reset to 1 as soon as the logical clock has made a progress that is R_v larger than the progress of the hardware clock, that is, ρ_v is set to 1 when the hardware clock value reaches $H_v^R = H_v + R_v/\mu$ (Algorithm 4). Naturally, new information can cause v to set H_v^R to a smaller or a larger value. Of course, if we do not insist on a strict upper bound on the logical clock rate, the computed increase R_v can simply be added to the logical clock value.

Algorithm 4. H_v reaches H_v^R.

1 $\rho_v := 1$

So far we did not cover the initialization process of $\mathcal{A}^{\text{opt}}$. This can be treated in a straightforward manner by interpreting the first received message as an initialization

message, starting both the hardware and the logical clock, setting $L_v^{\max}$ to the received value, and triggering a sending event. Any node waking up by itself simply sets $L_v^{\max} := 0$ and sends $\langle 0, 0 \rangle$ to its neighbors. The first message from a neighbor $w \in \mathcal{N}_v$ further initializes L_v^w and l_v^w. Until this message is received v is oblivious to w's existence and acts based only on the subset of neighbors from which it has already received a message. This scheme also allows for initially unknown topologies as nodes are integrated by means of their first message.

5. *Analysis of* $\mathcal{A}^{\text{opt}}$

Before we begin our analysis of the worst-case clock skew that may occur when algorithm $\mathcal{A}^{\text{opt}}$ is used, we need to discuss some preliminaries. En route, we will see that $\mathcal{A}^{\text{opt}}$ satisfies Constraints (1) and (2), that is, $\mathcal{A}^{\text{opt}}$ is indeed a clock synchronization algorithm that adheres to the rules of our model.

5.1. GENERAL STATEMENTS. We will frequently use two basic definitions. The first definition captures how much the increase of a logical clock exceeds the minimum increase, which is determined by the minimum hardware clock rate $1 - \varepsilon$, in a given time period $[t_1, t_2]$.

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

It follows directly from the definition that $\mathcal{I}_v$ is positive, monotonic in both arguments and interval additive.

The second definition captures how much the logical clock value increased more than the hardware clock value in a given time period $[t_1, t_2]$:

$$\mathcal{R}_v(t_1, t_2) := L_v(t_2) - L_v(t_1) - (H_v(t_2) - H_v(t_1)).$$

Again, $\mathcal{R}_v$ is positive, monotonic, and interval additive. Given that $h_v(t) \in [1 - \varepsilon, 1 + \varepsilon]$ at all times t, the two quantities $\mathcal{I}_v(t_1, t_2)$ and $\mathcal{R}_v(t_1, t_2)$ are related by the inequality

$$\mathcal{R}_v(t_1, t_2) \leq \mathcal{I}_v(t_1, t_2) \leq \mathcal{R}_v(t_1, t_2) + 2\varepsilon(t_2 - t_1). \tag{3}$$

Given these definitions, we can prove the following essential property of $\mathcal{A}^{\text{opt}}$.

LEMMA 5.1. *If the subroutine* setClockRate *(Algorithm 3) is called at a node v at a time when no message is received, both ρ_v and H_v^R remain unchanged.*

PROOF. Assume that the subroutine is called at time t, and let $t' < t$ be the time when the last message arrived at node v. By definition, the increase of the logical clock value in the time interval $[t', t]$ exceeds the increase of the local estimates $L_v^{\max}$ and L_v^w by $\mathcal{R}_v(t', t)$.

If $\mathcal{R}_v(t', t) = 0$, Algorithm 3 must have determined at time t' that the logical and the hardware clock value have to increase at the same rate. Therefore, if *setClockRate* was called at time t, we would have that $R_v(t) \leq 0$ because $\Lambda_v^{\uparrow}$ and $\Lambda_v^{\downarrow}$, and also $L_v^{\max} - L_v$, remain unaltered. Thus, the logical clock rate multiplier ρ_v would still be set to 1 after the procedure call at time t. Moreover, H_v^R is not changed at time t.

If $\mathcal{R}_v(t', t) > 0$, it holds that $\Lambda_v^{\uparrow}(t) = \Lambda_v^{\uparrow}(t') - \mathcal{R}_v(t', t)$ and $\Lambda_v^{\downarrow}(t) = \Lambda_v^{\downarrow}(t') + \mathcal{R}_v(t', t)$. The value R_v in Line 1 of Algorithm 3 thus reduces by exactly $\mathcal{R}_v(t', t)$. Moreover, $\kappa - \Lambda_v^{\downarrow}$ and $L_v^{\max} - L_v$ also reduce by exactly $\mathcal{R}_v(t', t)$ in Line 2, implying

that $R_v(t) = R_v(t') - \mathcal{R}_v(t', t)$. If $R_v(t)$ evaluates to zero, i.e., $\mathcal{R}_v(t', t) = R_v(t')$, the hardware clock must have reached H_v^R and the logical clock rate multiplier has been reset to 1 by time t.[5] The logical clock rate multiplier is not set to $1 + \mu$, because $R_v(t) \leq 0$, and H_v^R is again not changed at time t. If $R_v(t) > 0$, the logical clock rate multiplier is set to $1 + \mu$ until the hardware clock value reaches $H_v^R(t)$. Since $\mathcal{R}_v(t', t) = \mu(H_v(t) - H_v(t'))$, it holds that

$$\begin{aligned} H_v^R(t) = H_v(t) + \frac{R_v(t)}{\mu} &= H_v(t') + (H_v(t) - H_v(t')) + \frac{R_v(t') - \mathcal{R}_v(t', t)}{\mu} \\ &= H_v(t') + \frac{\mathcal{R}_v(t', t)}{\mu} + \frac{R_v(t') - \mathcal{R}_v(t', t)}{\mu} \\ &= H_v(t') + \frac{R_v(t')}{\mu} = H_v^R(t'). \end{aligned}$$

Hence, the logical clock rate multiplier remains $1 + \mu$ until the same hardware clock time as before. We conclude that ρ_v and H_v^R remain exactly the same in all cases. □

This property is useful in that it allows us to determine the logical clock rate also at times when no message is processed: We can simply assume that the variables are updated and the subroutine *setClockRate* is called, which does not cause the logical clock rate to change at any such time. As a first application, this permits a straightforward proof that Condition (1) holds.

COROLLARY 5.2. *Let* $L^{\max}(t) := \max_{v \in V}\{L_v^{\max}(t)\}$ *denote the maximum estimate of the maximum clock value in the system. It holds that*

(*i*) $\forall v \in V\ \forall t\ :\ L_v(t) \leq L^{\max}(t)$

(*ii*) $\forall t' > t\ :\ L^{\max}(t') - L^{\max}(t) \leq (1 + \varepsilon)(t' - t)$.

PROOF. At initialization time t_v we have that $L_v(t_v) \leq L_v^{\max}(t_v)$. At times when $L_v = L_v^{\max}$ Subroutine (3) computes $R_v \leq 0$ in Line 2 and thus ρ_v is set to 1. Hence, due to Lemma 5.1, at such times indeed it holds that $\rho_v = 1$ and L_v and $L_v^{\max}$ increase at the same rate. As L_v is continuous and $L_v^{\max}$ is monotonically increasing, we conclude that $L_v(t) \leq L_v^{\max}(t) \leq L^{\max}(t)$ at any time t. Since $L_v^{\max}$ is increased at the rate $h_v(t) \leq 1 + \varepsilon$ locally and never set to a value not yet reached by another node, Statement (*ii*) holds. □

COROLLARY 5.3. *Algorithm* $\mathcal{A}^{\text{opt}}$ *satisfies Condition* (1) *and Condition* (2) *with* $\alpha = 1 - \varepsilon$ *and* $\beta = (1 + \varepsilon)(1 + \mu)$.

PROOF. Since $\rho_v \geq 1$ at all times, we have that $L_v(t) \geq (1 - \varepsilon)(t - t_v)$ at all times $t \geq t_v$. Statement (*ii*) of Corollary 5.2 shows that $L^{\max}(t) \leq (1 + \varepsilon)t$, which together with Statement (*i*) implies $L_v(t) \leq (1 + \varepsilon)t$. Since ρ_v may only take the values 1 and $1 + \mu$, logical clocks increase at a rate of at least $h_v(t) \geq 1 - \varepsilon$ and at most $h_v(t)(1 + \mu) \leq (1 + \varepsilon)(1 + \mu)$. □

[5]Note that $R_v(t)$ cannot be negative because ρ_v is set to 1 as soon as $\mathcal{R}_v(t', t) = R_v(t)$ and no message is received in the interval $(t', t]$.

Hence, $\mathcal{A}^{\text{opt}}$ is correct regardless of the choices of H_0, κ and μ. Not surprisingly, to achieve a good synchronization quality, these parameters have to be chosen carefully. Since κ contributes linearly to the local skew as we will see, it should be kept as small as possible. However, we require that it is at least

$$\kappa \geq 2((1+\varepsilon)(1+\mu)\mathcal{T} + \bar{H}_0), \tag{4}$$

where

$$\bar{H}_0 := (2\varepsilon + \mu)H_0. \tag{5}$$

The intuition behind this lower bound is that κ must be large enough to compensate for the inaccuracy of the known clock values of the neighboring nodes due to the delay uncertainty $\mathcal{T}$. In order to give the algorithm a chance to react to clock skews, $\mathcal{T}$ is multiplied by $(1+\varepsilon)(1+\mu)$. Obviously, the accuracy of the information about neighboring clocks deteriorates if H_0 is set to a large value. Since clock skew can be built up at a rate of at most $\mathcal{O}(\mu)$, the additional skew is bounded by $\mathcal{O}(\mu H_0)$. Therefore, κ must further include the term $\bar{H}_0$ as defined above. The factor of two is due to the fact that any node v might possess outdated information about both the nodes whose clocks are ahead and the nodes whose clocks are behind.

It is sufficient to choose a $\mu \in \Omega(\varepsilon)$. However, as long as $\mu \ll 1$, larger choices will result in a smaller local skew. In order to incorporate this in our analysis, we define that $\sigma \geq 2$ is the largest integer such that

$$\mu \geq 7\sigma \frac{\varepsilon}{1-\varepsilon}. \tag{6}$$

We see that it suffices to set μ to roughly 14ε for any reasonable ε, that is, the precision of the clocks reduces by merely one order of magnitude while clock skews are corrected. In the following, we assume that Inequalities (4) and (6) are always satisfied.

Before we can proceed to bound the global and local skew of $\mathcal{A}^{\text{opt}}$, we need one more tool. The following lemma gives a bound on the accuracy of the estimates that the nodes have of their neighbors' clock values.

LEMMA 5.4. *For all nodes $v \in V$ and $w \in \mathcal{N}_v$ it holds for all times t when v has received at least one message from w that*

$$L_v^w(t) > L_w(t - \mathcal{T}) - \bar{H}_0. \tag{7}$$

PROOF. Set $t' := t - \mathcal{T}$ and let t_s denote the time when w sent the largest clock value that v received at the latest at time t, and let $t_r \leq t$ be the time when v received this clock value. Since v sets its estimate to the received value at time t_r, it holds that $L_v^w(t_r) = L_w(t_s)$.

If $t_s \geq t'$, we have that

$$L_v^w(t) \geq L_v^w(t_r) = L_w(t_s) \geq L_w(t') > L_w(t') - \bar{H}_0.$$

If $t_s < t'$, it must hold that $H_w(t') - H_w(t_s) < H_0$, as otherwise w sends a message that arrives at v at the latest at time t and that contains a larger clock value than the clock value sent at time t_s. Furthermore, $L_w(t') - L_w(t_s)$ is upper bounded

by $(1+\mu)(H_w(t') - H_w(t_s))$. Consequently, it holds that

$$\begin{aligned}\mathcal{I}_w(t_s, t') &= L_w(t') - L_w(t_s) - (1-\varepsilon)(t' - t_s)\\ &\leq \left(1 + \mu - \frac{1-\varepsilon}{1+\varepsilon}\right)(H_w(t') - H_w(t_s))\\ &< (\mu + 2\varepsilon)H_0 \overset{(5)}{=} \bar{H}_0. \end{aligned} \tag{8}$$

This observation and the fact that L_v^w is increased at the hardware clock rate in the interval $[t_r, t]$ allow us to bound

$$\begin{aligned}L_v^w(t) &\geq L_w(t_s) + (1-\varepsilon)(t - t_r)\\ &= L_w(t') - \mathcal{I}_w(t_s, t') - (1-\varepsilon)(t' - t_s) + (1-\varepsilon)(t - t_r)\\ &\overset{(8)}{>} L_w(t') - \bar{H}_0 - (1-\varepsilon)(t_r - t_s) + (1-\varepsilon)(t - t')\\ &\geq L_w(t') - \bar{H}_0.\end{aligned}$$

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

5.2. GLOBAL SKEW. First, we derive a bound on the global skew when executing $\mathcal{A}^{\text{opt}}$ on any graph G.

THEOREM 5.5. *The global skew of Algorithm $\mathcal{A}^{\text{opt}}$ is upper bounded by*

$$\mathcal{G} := (1+\varepsilon)D\mathcal{T} + \frac{2\varepsilon}{1+\varepsilon}H_0. \tag{9}$$

PROOF. Define $L^{\max}(t) := \max_{v\in V}\{L_v^{\max}(t)\}$ as in Corollary 5.2. Instead of bounding the clock skew between the nodes directly, we show that $L^{\max}(t) - L_v(t) \leq \mathcal{G}$ for all nodes $v \in V$ and times t. As $L^{\max}(t) \geq L_v(t)$ for all $v \in V$ and times t according to Corollary 5.2, this statement proves the theorem.

For the sake of contradiction, assume that $\bar{t}$ is the infimum of all times when the clock skew between $L^{\max}$ and the clock value L_v of some node v exceeds $\mathcal{G}$. Since $L^{\max}$ and L_v are continuous, it holds that

$$L^{\max}(\bar{t}\,) - L_v(\bar{t}\,) = \mathcal{G}. \tag{10}$$

First, assume that $\Lambda_v^{\downarrow}(\bar{t}\,) < \kappa$. Let L be the largest estimate of the maximum clock value that v receives at the latest at time $\bar{t}$. Moreover, let t_s and $t_r \leq \bar{t}$ be the times when L is first sent and when it is received by v, respectively.[6] Since t_s is the first time when L was sent, we have that $L = L^{\max}(t_s)$. Assume that $t_s \geq \bar{t} - D\mathcal{T}$. As $L_v^{\max}$ increases at the rate $h_v \geq 1-\varepsilon$, we get that

$$\begin{aligned}L_v^{\max}(\bar{t}\,) &\geq L + (1-\varepsilon)(\bar{t} - t_r)\\ &= L^{\max}(t_s) + (1-\varepsilon)(\bar{t} - t_r)\\ &\geq L^{\max}(\bar{t}\,) - (1+\varepsilon)(\bar{t} - t_s) + (1-\varepsilon)(\bar{t} - t_r) \qquad (11)\\ &\overset{(10)}{\geq} L_v(\bar{t}\,) + \mathcal{G} - (1+\varepsilon)D\mathcal{T}\\ &> L_v(\bar{t}\,),\end{aligned}$$

where Inequality (11) follows from Statement (*ii*) of Corollary 5.2.

[6] Note that t_s and t_r are not (necessarily) the send and receive event of the same message. The estimate L of the maximum clock value in the system may be forwarded to v along a path of nodes.

The message may have been sent earlier at a time $t_s = \bar{t} - D\mathcal{T} - \gamma$ for a value $\gamma \in (0, H_0/(1-\varepsilon))$. Note that $t_s \leq \bar{t} - D\mathcal{T} - H_0/(1-\varepsilon)$ is not possible as the progress of any hardware clock until $\bar{t} - D\mathcal{T}$ is at least H_0, implying that another message containing a larger estimate would certainly have been sent until time $\bar{t} - D\mathcal{T}$. In this case, the estimate would have been forwarded towards v unless some node on the path had already sent a larger estimate of the maximum clock value earlier. Since the distance to v is upper bounded by D and the message delays are at most $\mathcal{T}$, node v would receive a message containing (at least) $L + H_0$ at the latest at time $\bar{t}$, contradicting the assumption that L is the largest estimate that arrives until $\bar{t}$. More generally, if $h^{\max} \in [1-\varepsilon, 1+\varepsilon]$ denotes the average rate at which $L^{\max}$ increases in the interval $[t_s, \bar{t} - D\mathcal{T}]$, it holds that $h^{\max}\gamma < H_0$, otherwise L is not the largest estimate of the maximum clock value that arrives at v at the latest at time $\bar{t}$. Since $t_r \leq t_s + D\mathcal{T}$ and thus $\bar{t} - t_r \geq \gamma$, it holds in this case that

$$\begin{aligned} L_v^{\max}(\bar{t}) &\geq L + (1-\varepsilon)(\bar{t} - t_r) \\ &\geq L^{\max}(t_s) + (1-\varepsilon)\gamma \\ &\geq L^{\max}(\bar{t}) - (1+\varepsilon)D\mathcal{T} - h^{\max}\gamma + (1-\varepsilon)\gamma \\ &\overset{(10)}{=} \mathcal{G} + L_v(\bar{t}) - (1+\varepsilon)D\mathcal{T} - h^{\max}\gamma\left(1 - \frac{1-\varepsilon}{h^{\max}}\right) \\ &> \mathcal{G} + L_v(\bar{t}) - (1+\varepsilon)D\mathcal{T} - \frac{2\varepsilon}{1+\varepsilon}H_0 \qquad (12) \\ &= L_v(\bar{t}). \end{aligned}$$

Thus, in both cases it holds that $L_v(\bar{t}) < L_v^{\max}(\bar{t})$. This observation, the assumption that $\Lambda_v^{\downarrow} < \kappa$ and Lemma 5.1 imply that $\rho_v(\bar{t}) = 1 + \mu$. Consequently, the rate of v's logical clock at time $\bar{t}$ is at least $(1-\varepsilon)(1+\mu)$, which is larger than the maximum clock rate $1+\varepsilon$ of $L^{\max}$ according to Inequality (6). Hence, $\bar{t}$ cannot be the infimum of all times when the clock skew between $L^{\max}$ and L_v exceeds $\mathcal{G}$, a contradiction.[7]

It remains to study the case that $\Lambda_v^{\downarrow}(\bar{t}) \geq \kappa$. Let $w \in \mathcal{N}_v$ be any node such that $L_v(\bar{t}) - L_v^w(\bar{t}) \geq \kappa$, and let t_s denote the time when w sent the message containing the largest clock value that v received at a time $t_r \leq \bar{t}$. It holds that $L_v(t_r) - L_v^w(t_r) + \mathcal{R}_v(t_r, \bar{t}) = L_v(\bar{t}) - L_v^w(\bar{t}) \geq \kappa$ because L_v increases by exactly $\mathcal{R}_v(t_r, \bar{t})$ more than the hardware clock and thus also $\mathcal{R}_v(t_r, \bar{t})$ more than the estimate L_v^w. Hence, since the message received at time t_r contains the clock value $L_w(t_s)$, which implies that $L_v^w(t_r) = L_w(t_s)$, we know that

$$L_w(t_s) \leq L_v(t_r) + \mathcal{R}_v(t_r, \bar{t}) - \kappa. \qquad (13)$$

Furthermore, it holds that

$$\begin{aligned} L_v(\bar{t}) &= L_v(t_r) + H_v(\bar{t}) - H_v(t_r) + \mathcal{R}_v(t_r, \bar{t}) \\ &\geq L_v(t_r) + (1-\varepsilon)(\bar{t} - t_r) + \mathcal{R}_v(t_r, \bar{t}) \qquad (14) \end{aligned}$$

[7] Strictly speaking, this does not follow from the observation on v's clock rate at $\bar{t}$ only. However, since the number of messages sent until any given time is finite, a time interval of positive length starting at time $\bar{t}$ must exist in which v receives no messages and sticks to the faster clock rate.

If we assume that $t_s \geq \bar{t} - \mathcal{T}$, the skew between $L^{\max}$ and L_w at time t_s is at least

$$\begin{aligned} L^{\max}(t_s) - L_w(t_s) &\overset{(13)}{\geq} L^{\max}(\bar{t}) - (1+\varepsilon)(\bar{t} - t_s) - L_v(t_r) - \mathcal{R}_v(t_r, \bar{t}) + \kappa \\ &\overset{(14)}{\geq} L^{\max}(\bar{t}) - L_v(\bar{t}) + \kappa - (1+\varepsilon)(\bar{t} - t_s) + (1-\varepsilon)(\bar{t} - t_r) \\ &= L^{\max}(\bar{t}) - L_v(\bar{t}) + \kappa - (1-\varepsilon)(t_r - t_s) - 2\varepsilon(\bar{t} - t_s) \\ &\overset{(10)}{\geq} \mathcal{G} + \kappa - (1+\varepsilon)\mathcal{T} \overset{(4)}{>} \mathcal{G}. \end{aligned}$$

If $t_s = \bar{t} - \mathcal{T} - \gamma$ for a value $\gamma \in (0, H_0/(1-\varepsilon))$ as before, we again have that $h_w \gamma < H_0$, where h_w now denotes the average clock rate of w in the time interval $[t_s, \bar{t} - \mathcal{T}]$. The same arguments as in the previous cases reveal that

$$\begin{aligned} L^{\max}(\bar{t} - \mathcal{T}) - L_w(\bar{t} - \mathcal{T}) &\geq L^{\max}(\bar{t}) - (1+\varepsilon)\mathcal{T} - (L_w(t_s) + h_w\gamma) \\ &\overset{(13)}{\geq} L^{\max}(\bar{t}) - L_v(t_r) + \kappa \\ &\quad -\mathcal{R}_v(t_r, \bar{t}) - (1+\varepsilon)\mathcal{T} - h_w\gamma \\ &\overset{(14)}{\geq} L^{\max}(\bar{t}) - L_v(\bar{t}) + \kappa \\ &\quad -(1+\varepsilon)\mathcal{T} - h_w\gamma + (1-\varepsilon)(\bar{t} - t_r) \\ &\overset{\bar{t} - t_r \geq \gamma}{\geq} \mathcal{G} + \kappa - (1+\varepsilon)\mathcal{T} - h_w\gamma\left(1 - \frac{1-\varepsilon}{h_w}\right) \\ &\overset{h_w\gamma < H_0}{>} \mathcal{G} + \kappa - (1+\varepsilon)\mathcal{T} - \frac{2\varepsilon}{1+\varepsilon}H_0 \overset{(4)}{\geq} \mathcal{G}. \end{aligned}$$

Thus, in either case the skew between $L^{\max}$ and L_w exceeded $\mathcal{G}$ at a time earlier than $\bar{t}$. This is a contradiction as $L^{\max}(t) - L_v(t) \leq \mathcal{G}$ for all $v \in V$ at all times $t < \bar{t}$ because $\bar{t}$ is the infimum of all times when this bound is violated. □

5.3. Local Skew. We now focus our attention on the worst-case clock skew between neighboring nodes. Consider the maximum length of a path with a given average clock skew ΔL between the nodes of this path. The proof of the bound on the local skew relies on the fact that linearly increasing the average clock skew ΔL results in an exponential decrease in the length of the longest possible path that exhibits such an average clock skew. This fact implies that the average skew on paths of length one, that is, between neighboring nodes, is logarithmically bounded in the diameter D. In particular, we show that the system is always in a *legal state*, which is defined as follows.

Definition 5.6 (*Legal State*). Given the integer $\sigma \geq 2$, we say that a distributed system is in a *legal state* at time t, if and only if for all $s \in \mathbb{N}_0$ and all nodes $v, w \in V$ at distance

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

we have that

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

Note that Theorem 5.5 shows that any two nodes v and w at a distance of at least C_0 cannot violate the legal state condition, because $L_v(t) - L_w(t) \leq \mathcal{G} \leq d(v, w)\kappa/2$ at all times t.

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

LEMMA 5.7. *Given $\xi \geq 0$ and $s \in \mathbb{N}$, assume that the clock skew between two nodes v and w at time t_0 is*

$$L_v(t_0) - L_w(t_0) = d(v,w)\left(s - \frac{1}{2}\right)\kappa + \xi. \tag{15}$$

Define $t_1 := t_0 + \frac{\kappa C_{s-1}}{(1-\varepsilon)\mu} + \mathcal{T}$. If the system is in a legal state at time t_0, it follows that

$$\mathcal{I}_w(t_0, t_1) \geq \xi. \tag{16}$$

PROOF. Define

$$\Xi(t) := \max_{u \in V}\left\{L_v(t_0) - L_u(t_0) - d(v,u)\left(s - \frac{1}{2}\right)\kappa - \mathcal{I}_u(t_0, t)\right\}.$$

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

$$\mathcal{I}_w(t_0, t) \geq L_v(t_0) - L_w(t_0) - d(v,w)\left(s - \frac{1}{2}\right)\kappa = \xi.$$

Furthermore, $\Xi(\cdot)$ is monotonically decreasing, hence showing that $\Xi(t) \leq 0$ for any $t \leq t_1$ proves the lemma. We proceed by deriving an upper bound on $\Xi(t_0)$.

Consider any node $u \in V$. If $d(v,u) \geq C_0$, it holds that $L_v(t_0) - L_u(t_0) \leq d(v,u)\kappa/2$ due to the legal state condition, which was satisfied at time t_0. We get that $L_v(t_0) - L_u(t_0) - d(v,u)(s - 1/2)\kappa \leq 0$, i.e., $\Xi(\cdot)$ cannot become positive because of a node u at distance $d(v,u) \geq C_0$. Hence, we may assume that $d(v,u) < C_0$, i.e., there is an integer $r \geq 1$ such that $d(v,u) \in [C_r, C_{r-1})$. The legal state condition states that

$$L_v(t_0) - L_u(t_0) - d(v,u)\left(s - \frac{1}{2}\right)\kappa \leq d(v,u)(r - s + 1)\kappa. \tag{17}$$

The right-hand side of this inequality is at most 0 for $r < s$. If $r \geq s$ it holds that

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

because $\sigma \geq 2$. Hence it follows that $\Xi(t_0) \leq \kappa C_{s-1}$.

The second step is to prove that $\Xi(\cdot)$ decreases at least at an average rate of $(1-\varepsilon)\mu$ until it reaches zero. In particular, we claim that

$$\Xi(t) \leq \max\{0, \Xi(t_0) - (1-\varepsilon)\mu(t - t_0) + (1-\varepsilon)\mu\mathcal{T}\} \tag{18}$$

for all times $t \geq t_0$.

This statement is trivially true for all times $t \leq t_0 + \mathcal{T}$ because in this case we only require that $\Xi(t) \leq \Xi(t_0)$, which follows from the fact that $\Xi(\cdot)$ is monotonically decreasing. Assume for the sake of contradiction that the claim that Inequality (18) holds at all times $t \geq t_0$ is false. Let $\bar{t}$ be the infimum of all times $t \geq t_0 + \mathcal{T}$ when $\Xi(t) > \Xi(t_0) - (1-\varepsilon)\mu(t - t_0) + (1-\varepsilon)\mu\mathcal{T} > 0$ and let u be a node that

maximizes $\Xi(\bar{t}\,)$. Note that $u \neq v$ because v cannot cause the value of the function $\Xi(\cdot)$ to become positive due to the fact that $\mathcal{I}_v(t_0, t) \geq 0$. Since $\Xi(\cdot)$ is a continuous function, it holds at time $\bar{t}$ that

$$\begin{aligned} &L_v(t_0) - L_u(t_0) - d(v,u)\left(s - \frac{1}{2}\right)\kappa - \mathcal{I}_u(t_0, \bar{t}\,) \\ &\quad = \Xi(t_0) - (1-\varepsilon)\mu(\bar{t} - t_0) + (1-\varepsilon)\mu\mathcal{T}. \end{aligned} \tag{19}$$

Consider any neighbor u' of u that is closer to v than u, that is, the distance between v and u' is $d(v,u') = d(v,u) - 1$. By definition of $\bar{t}$, Inequality (18) holds for node u' at time $t' := \bar{t} - \mathcal{T} \geq t_0$. Thus, we have that

$$\begin{aligned} &L_v(t_0) - L_{u'}(t_0) - d(v,u')\left(s - \frac{1}{2}\right)\kappa - \mathcal{I}_{u'}(t_0, t') \\ &\quad \leq \Xi(t_0) - (1-\varepsilon)\mu(t' - t_0) + (1-\varepsilon)\mu\mathcal{T} \\ &\quad = \Xi(t_0) - (1-\varepsilon)\mu(\bar{t} - t_0) + 2(1-\varepsilon)\mu\mathcal{T}. \end{aligned} \tag{20}$$

By subtracting Inequality (20) from Eq. (19), we get that

$$\begin{aligned} &L_{u'}(t_0) + \mathcal{I}_{u'}(t_0, t') - L_u(t_0) - \mathcal{I}_u(t_0, \bar{t}\,) \\ &\quad \geq (d(v,u) - d(v,u'))\left(s - \frac{1}{2}\right)\kappa - (1-\varepsilon)\mu\mathcal{T} \\ &\quad = \left(s - \frac{1}{2}\right)\kappa - (1-\varepsilon)\mu\mathcal{T}. \end{aligned} \tag{21}$$

If u had not received a message from u' until $\bar{t}$, then u' would not have been initialized at time $\bar{t} > t'$, that is, $L_{u'}(t_0) + \mathcal{I}_{u'}(t_0, t') = 0$. Since both $L_u(t_0)$ and $\mathcal{I}_u(t_0, \bar{t}\,)$ are positive, Inequality (21) then yields that

$$\frac{\kappa}{2} \leq \left(s - \frac{1}{2}\right)\kappa \leq (1-\varepsilon)\mu\mathcal{T}.$$

However, this is impossible as $\kappa/2 > \mu\mathcal{T}$ due to Inequality (4). We conclude that u must have already received a message from u' at time $\bar{t}$. Thus, as $t' = \bar{t} - \mathcal{T}$, Lemma 5.4 can be used in order to lower bound $\Lambda_u^{\uparrow}(\bar{t}\,)$:

$$\begin{aligned} \Lambda_u^{\uparrow}(\bar{t}\,) \geq L_u^{u'}(\bar{t}\,) - L_u(\bar{t}\,) &\overset{(7)}{>} L_{u'}(t') - \bar{H}_0 - L_u(\bar{t}\,) \\ &= L_{u'}(t_0) + \mathcal{I}_{u'}(t_0, t') - L_u(t_0) - \mathcal{I}_u(t_0, \bar{t}\,) \\ &\quad -(1-\varepsilon)(\bar{t} - t') - \bar{H}_0 \\ &\overset{(21)}{\geq} \left(s - \frac{1}{2}\right)\kappa - (1-\varepsilon)\mu\mathcal{T} - (1-\varepsilon)\mathcal{T} - \bar{H}_0 \\ &\overset{(4)}{>} (s-1)\kappa. \end{aligned}$$

This bound and the fact that the estimate of the maximum clock value is at least as large as the estimated clock value of any neighbor imply that

$$L_u^{\max}(\bar{t}\,) \geq \Lambda_u^{\uparrow}(\bar{t}\,) - L_u(\bar{t}\,) > (s-1)\kappa - L_u(\bar{t}\,) \geq L_u(\bar{t}\,).$$

Hence, in accordance with Lemma 5.1, the clock rate is $1 + \mu$ unless a clock value a neighboring node sent to u is too small.

Consider an arbitrary node $u'' \in \mathcal{N}_u$ from which u has received a message until time $\bar{t}$. The distance between v and u'' is at most $d(v,u)+1$. As u'' did not violate the claimed bound at time $t' = \bar{t} - \mathcal{T}$, it holds that

$$\begin{aligned} &L_v(t_0) - L_{u''}(t_0) - d(v,u'')\left(s - \frac{1}{2}\right)\kappa - \mathcal{I}_{u''}(t_0,t') \\ &\quad \leq \Xi(t_0) - (1-\varepsilon)\mu(\bar{t}-t_0) + 2(1-\varepsilon)\mu\mathcal{T}. \end{aligned}$$

By subtracting Eq. (19), we get that

$$\begin{aligned} &L_u(t_0) + \mathcal{I}_u(t_0,\bar{t}) - L_{u''}(t_0) - \mathcal{I}_{u''}(t_0,t') \\ &\quad \leq (d(v,u'') - d(v,u))\left(s - \frac{1}{2}\right)\kappa + (1-\varepsilon)\mu\mathcal{T} \\ &\quad \leq \left(s - \frac{1}{2}\right)\kappa + (1-\varepsilon)\mu\mathcal{T}. \end{aligned} \tag{22}$$

This inequality is used to upper bound $L_u(\bar{t}) - L_u^{u''}(\bar{t})$:

$$\begin{aligned} L_u(\bar{t}) - L_u^{u''}(\bar{t}) &\overset{(7)}{<} L_u(\bar{t}) - L_{u''}(t') + \bar{H}_0 \\ &= L_u(t_0) + \mathcal{I}_u(t_0,\bar{t}) - L_{u''}(t_0) - \mathcal{I}_{u''}(t_0,t') \\ &\quad +(1-\varepsilon)(\bar{t}-t') + \bar{H}_0 \\ &\overset{(22)}{\leq} \left(s - \frac{1}{2}\right)\kappa + (1-\varepsilon)\mu\mathcal{T} + (1-\varepsilon)\mathcal{T} + \bar{H}_0 \\ &\overset{(4)}{<} s\kappa. \end{aligned}$$

Since this bound holds for any neighbor $u'' \in \mathcal{N}_u$ from which u receives a message until time $\bar{t}$, we can infer that $\Lambda_u^{\downarrow}(\bar{t}) < s\kappa$.

Altogether, we have that $L_u(\bar{t}) < L_u^{\max}(\bar{t})$, $\Lambda_u^{\uparrow}(\bar{t}) > (s-1)\kappa$, and $\Lambda_u^{\downarrow}(\bar{t}) \leq L_u(\bar{t}) - L_u^{u''}(\bar{t}) < s\kappa$. Hence, the subroutine *setClockRate* would set R_u to a positive value at time $\bar{t}$ and the logical clock rate multiplier to $1+\mu$. Thus, according to Lemma 5.1, the value of the logical clock rate multiplier ρ_u is $1+\mu$ at time $\bar{t}$. This result implies that $\mathcal{I}_u$ grows at a rate of at least $(1+\mu)(1-\varepsilon) - (1-\varepsilon) = (1-\varepsilon)\mu$ at time $\bar{t}$. Since the rate of $\mathcal{I}_u$ is at least $(1-\varepsilon)\mu$ for any u for which Inequality (19) holds, it follows that the rate of $\Xi(\cdot)$ at time $\bar{t}$ is at most $-(1-\varepsilon)\mu$, contradicting the definition that $\bar{t}$ is the infimum of all times when the claim does not hold.[8] Thus, at time $t_1 = t_0 + \frac{\kappa C_{s-1}}{(1-\varepsilon)\mu} + \mathcal{T}$, it holds that $\Xi(t_1) \leq \max\{0,\ \Xi(t_0) - (1-\varepsilon)\mu(t_1 - t_0) + (1-\varepsilon)\mu\mathcal{T}\} = 0$ as desired. □

The second lemma shows that the clock skew can only increase slowly once it reaches a certain level. More precisely, for any $s \in \mathbb{N}$, we consider the path on which the average clock skew exceeds $s\kappa$ the most. If v is the node with the largest and w is the node with the smallest clock value among all nodes on this path, then v's logical clock runs at the hardware clock rate, that is, the clock skew between v and w can only grow further at a rate of at most 2ε. Moreover, the clock skew

[8] Again, being strictly formal, there must be a time interval of positive length in which none of the critical nodes receive messages and their clock values increase quickly.

decreases if w increases its clock value quickly. In order to abbreviate the notation, the following definition is introduced.

Definition 5.8. Given a node $w \in V$ and $s \in \mathbb{N}$, define for any time t

$$\Psi_w^s(t) := \max_{v \in V} \{L_v(t) - L_w(t) - s\kappa d(v,w)\} \geq 0.$$

LEMMA 5.9. *If $\Psi_w^s(t) > 0$ for all $t \in (t_0, t_1]$, then it holds at any time $t \in [t_0, t_1]$ that*

$$\Psi_w^s(t) \leq 2\varepsilon(t - t_0) - \mathcal{I}_w(t_0, t) + \frac{\kappa}{7\sigma} + \Psi_w^s(t_0). \tag{23}$$

PROOF. Note that Inequality (23) holds trivially at time t_0. Assume for the sake of contradiction that $\bar{t} \in (t_0, t_1]$ is the infimum of times when the claim is false. As Ψ_w^s is continuous, this means that a node $v \in V$ exists such that

$$L_v(\bar{t}\,) - L_w(\bar{t}\,) - s\kappa d(v,w) = 2\varepsilon(\bar{t} - t_0) - \mathcal{I}_w(t_0, \bar{t}\,) + \frac{\kappa}{7\sigma} + \Psi_w^s(t_0). \tag{24}$$

Since w cannot cause $\Psi_w^s(\cdot)$ to become positive, we know that $v \neq w$ and thus $d(v,w) \geq 1$. Moreover, it holds that $L_v(\bar{t}\,) \geq s\kappa d(v,w) \geq \kappa$. This implies that v had been initialized earlier than at time $\bar{t} - 2\mathcal{T}$, as otherwise

$$L_v(\bar{t}\,) \leq 2(1+\varepsilon)(1+\mu)\mathcal{T} \overset{(4)}{<} \kappa.$$

Therefore, at time $\bar{t}$ node v must have already received the first message from each neighbor.

Since $d(v,w) \geq 1$, a neighbor $u \in \mathcal{N}_v$ at distance $d(u,w) = d(v,w) - 1$ from w exists. Let t_s denote the time when u sent the largest clock value that v received at a time $t_r \leq \bar{t}$. We need to distinguish between the following two cases.

If $t_s \geq t_0$, Inequality (23) was not violated at time t_s, which allows us to bound

$$\begin{aligned}
L_v(\bar{t}\,) - L_v^u(\bar{t}\,) &= L_v(t_r) - L_v^u(t_r) + \mathcal{R}_v(t_r, \bar{t}\,) \\
&\overset{(3)}{\geq} L_v(t_r) - L_u(t_s) + \mathcal{I}_v(t_r, \bar{t}\,) - 2\varepsilon(\bar{t} - t_r) \\
&= L_v(\bar{t}\,) - L_w(\bar{t}\,) - (L_u(t_s) - L_w(\bar{t}\,)) \\
&\quad - (1-\varepsilon)(\bar{t} - t_r) - 2\varepsilon(\bar{t} - t_r) \\
&= L_v(\bar{t}\,) - L_w(\bar{t}\,) - (L_u(t_s) - L_w(t_s)) + \mathcal{I}_w(t_s, \bar{t}\,) \\
&\quad + (1-\varepsilon)(\bar{t} - t_s) - (1-\varepsilon)(\bar{t} - t_r) - 2\varepsilon(\bar{t} - t_r) \qquad (25) \\
&\overset{(23,24)}{\geq} s\kappa(d(v,w) - d(u,w)) - \mathcal{I}_w(t_0, \bar{t}\,) + \mathcal{I}_w(t_0, t_s) + \mathcal{I}_w(t_s, \bar{t}\,) \\
&\quad + (1-\varepsilon)(t_r - t_s) - 2\varepsilon(\bar{t} - t_r) + 2\varepsilon(\bar{t} - t_0) - 2\varepsilon(t_s - t_0) \\
&= s\kappa + (1+\varepsilon)(t_r - t_s) \geq s\kappa.
\end{aligned}$$

If $t_s < t_0$, note that the time difference $t_s - t_0$ is bounded because each node sends a message to its neighbors at the latest after its hardware clock value increased by H_0, that is, after at most $\frac{H_0}{1-\varepsilon}$ time, since it last sent a message. If $t_s \leq t_0 - \frac{H_0}{1-\varepsilon} - \mathcal{T}$, another message, containing a larger clock value, is sent at a time $t_s' \leq t_0 - \mathcal{T}$, which arrives at a time $t_r' \leq t_0$, contradicting the assumption that the message sent at time t_s contains the largest clock value that v receives from u until $\bar{t} \geq t_0$. Hence,

it follows that $t_s > t_0 - \frac{H_0}{1-\varepsilon} - \mathcal{T}$ and thus

$$t_0 - t_r \leq t_0 - t_s < \frac{H_0}{1-\varepsilon} + \mathcal{T} \overset{(5)}{<} \frac{\bar{H}_0}{(1-\varepsilon)\mu} + \mathcal{T} \overset{(6)}{\leq} \frac{\bar{H}_0}{7\sigma\varepsilon} + \mathcal{T}. \tag{26}$$

The clock skew between L_u and L_w at time t_s is bounded by

$$\begin{aligned} L_u(t_s) - L_w(t_s) &= L_u(t_0) - L_w(t_0) - \mathcal{I}_u(t_s, t_0) + \mathcal{I}_w(t_s, t_0) \\ &\leq s\kappa d(u, w) - \mathcal{I}_u(t_s, t_0) + \mathcal{I}_w(t_s, t_0) + \Psi^s_w(t_0) \\ &\leq s\kappa d(u, w) + \mathcal{I}_w(t_s, t_0) + \Psi^s_w(t_0). \end{aligned} \tag{27}$$

In this case, the estimated clock skew between v and u at time $\bar{t}$ is

$$\begin{aligned} L_v(\bar{t}) - L^u_v(\bar{t}) &\overset{(25)}{\geq} L_v(\bar{t}) - L_w(\bar{t}) - (L_u(t_s) - L_w(t_s)) + \mathcal{I}_w(t_s, \bar{t}) \\ &\quad + (1-\varepsilon)(t_r - t_s) - 2\varepsilon(\bar{t} - t_r) \\ &\overset{(24,27)}{\geq} s\kappa(d(v, w) - d(u, w)) - \mathcal{I}_w(t_0, \bar{t}) - \mathcal{I}_w(t_s, t_0) \\ &\quad + \mathcal{I}_w(t_s, \bar{t}) + \frac{\kappa}{7\sigma} + (1-\varepsilon)(t_r - t_s) - 2\varepsilon(t_0 - t_r) \\ &\overset{(26)}{>} s\kappa + \frac{\kappa}{7\sigma} - 2\varepsilon\left(\mathcal{T} + \frac{\bar{H}_0}{7\sigma\varepsilon}\right) \\ &\overset{(4)}{>} s\kappa + \frac{2(\mu\mathcal{T} + \bar{H}_0)}{7\sigma} - 2\varepsilon\left(\mathcal{T} + \frac{\bar{H}_0}{7\sigma\varepsilon}\right) \\ &\overset{(6)}{>} s\kappa. \end{aligned}$$

Thus, $L_v(\bar{t}) - L^u_v(\bar{t}) \geq s\kappa$ holds in either case. Applying the same arguments to any node $u' \in \mathcal{N}_v$ yields that $L^{u'}_v(\bar{t}) - L_v(\bar{t}) \leq s\kappa$, as all terms in the previous estimates switch signs, except the term $s\kappa$ because $d(v, w) - d(v, u') \geq -1$. Since these estimate holds for any node u', we have that $\Lambda^{\uparrow}_v(\bar{t}) \leq s\kappa$. Moreover, it holds that $\Lambda^{\downarrow}_v(\bar{t}) \geq L_v(\bar{t}) - L^u_v(\bar{t}) \geq s\kappa$, which implies that R_v evaluates to zero in Line 1 of the subroutine *setClockRate* (Algorithm 3) if the subroutine is called at time $\bar{t}$. Given that $s \in \mathbb{N}$, we further know that $\kappa - \Lambda^{\downarrow}_v(\bar{t}) \leq \kappa(1-s) \leq 0$. Hence, $R_v(\bar{t}) = 0$ and the logical clock rate is the same as the hardware clock rate at time $\bar{t}$ according to Lemma 5.1. Since the minimum progress rate of any node is $1-\varepsilon$ and the progress rate of v at time $\bar{t}$ is at most $1+\varepsilon$, the clock skew between v and w can only increase at the rate 2ε.[9] As the previous reasoning applies to any node v which is going to violate the bound, this is a contradiction to the assumption that $\bar{t}$ is the infimum of all times when the claim is violated, which concludes the proof. □

We are now in the position to prove the main theorem, which states that the local skew grows logarithmically with the diameter D of the graph.

[9] Once again, in order to argue minutely we have to observe that clock rate of v will equal h_v for a time period of non-zero length.

THEOREM 5.10. *The local skew when executing algorithm $\mathcal{A}^{\text{opt}}$ on any graph G of diameter D is upper bounded by*

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

PROOF. Let $\mathcal{G}' \geq \mathcal{G}$ be the number for which $\log_\sigma(2\mathcal{G}'/\kappa) = \lceil \log_\sigma(2\mathcal{G}/\kappa)\rceil$. It is convenient to assume, without loss of generality, that $\mathcal{G}'$ is the bound on the global skew because in this case $C_s \in \mathbb{N}$ for all $s \in \{0, \ldots, s_{\max}\}$, where $s_{\max} := \log_\sigma(2\mathcal{G}'/\kappa)$.

By definition, a skew of more than $d(v,w)(s_{\max}+1/2)\kappa$ between the clocks of any two nodes at distance $d(v,w) \geq C_{s_{\max}}$ cannot occur as long as the system is in a legal state. Since $C_{s_{\max}} = 1$, this means that the claimed bound on the worst-case skew between neighboring nodes can only be violated if the system is not in a legal state. Thus, if the system is always in a legal state, the theorem follows immediately.

Assume for the sake of contradiction that $\bar{t}$ is the infimum of all times when the system is not in a legal state. As argued before, the legal state condition cannot be violated for $s = 0$ as a violation implies that the clock skew between two nodes exceeds $\mathcal{G}' \geq \mathcal{G}$, a contradiction to Theorem 5.5. Hence, two nodes v and w at distance $d(v,w) \geq C_s$ for some $s \in \{1, \ldots, s_{\max}\}$ exist such that

$$L_v(\bar{t}) - L_w(\bar{t}) = d(v,w)\left(s + \frac{1}{2}\right)\kappa. \tag{28}$$

Define $t_0 := \bar{t} - \frac{\kappa C_{s-1}}{(1-\varepsilon)\mu} - \mathcal{T}$. Since $\kappa C_s \geq \kappa > (1-\varepsilon)\mu\mathcal{T}$, it holds that

$$\bar{t} - t_0 < \frac{(\sigma+1)}{(1-\varepsilon)\mu}\kappa C_s. \tag{29}$$

If $\Psi_w^s \leq 0$ at some point in time in the interval $[t_0, \bar{t}]$, its continuity permits us to define $t \in [t_0, \bar{t}]$ as the largest time such that $\Psi_w^s(t) = 0$. Note that $t < \bar{t}$ because at time $\bar{t}$ we have that

$$\begin{aligned}
\Psi_w^s(\bar{t}) &\geq L_v(\bar{t}) - L_w(\bar{t}) - s\kappa d(v,w) \\
&\overset{(28)}{=} \frac{\kappa}{2} d(v,w) \\
&\overset{d(v,w)\geq C_s}{\geq} \frac{\kappa}{2} C_s > 0.
\end{aligned} \tag{30}$$

In this case, by applying Lemma 5.9, we get that

$$\begin{aligned}
\frac{\kappa}{2}C_s &\overset{(30)}{\leq} \Psi_w^s(\bar{t}) \\
&\overset{(23)}{\leq} 2\varepsilon(\bar{t}-t) - \mathcal{I}_w(t,\bar{t}) + \frac{\kappa}{7\sigma} \\
&\leq 2\varepsilon(\bar{t}-t_0) + \frac{\kappa}{7\sigma}C_s \\
&\overset{(29)}{<} 2\varepsilon\frac{\sigma+1}{(1-\varepsilon)\mu}\kappa C_s + \frac{\kappa}{7\sigma}C_s.
\end{aligned}$$

This estimate implies that

$$2\varepsilon\frac{\sigma+1}{(1-\varepsilon)\mu}\kappa C_s > \left(\frac{1}{2}-\frac{1}{7\sigma}\right)\kappa C_s$$

and thus

$$\mu < \frac{28\varepsilon\sigma(\sigma+1)}{(1-\varepsilon)(7\sigma-2)} \overset{\sigma\geq 2}{\leq} 7\sigma\frac{\varepsilon}{1-\varepsilon}, \tag{31}$$

a contradiction to Condition (6).

Hence, it must hold that $\Psi_w^s(t) > 0$ for all $t \in [t_0, \bar{t}\,]$. Another application of Lemma 5.9 yields that

$$\begin{aligned}\frac{\kappa}{2}C_s - \Psi_w^s(t_0) &\overset{(30)}{\leq} \Psi_w^s(\bar{t}\,) - \Psi_w^s(t_0)\\ &\overset{(23)}{\leq} 2\varepsilon(\bar{t}-t_0) - \mathcal{I}_w(t_0,\bar{t}\,) + \frac{\kappa}{7\sigma}\\ &\overset{(29)}{<} 2\varepsilon\frac{\sigma+1}{(1-\varepsilon)\mu}\kappa C_s - \mathcal{I}_w(t_0,\bar{t}\,) + \frac{\kappa}{7\sigma}.\end{aligned}$$

By rearranging the terms, we get that

$$\Psi_w^s(t_0) > \frac{\kappa}{2}C_s - 2\varepsilon\frac{\sigma+1}{(1-\varepsilon)\mu}\kappa C_s + \mathcal{I}_w(t_0,\bar{t}\,) - \frac{\kappa}{7\sigma}. \tag{32}$$

Since $t_0 = \bar{t} - \frac{\kappa C_{s-1}}{(1-\varepsilon)\mu} - \mathcal{T}$, Lemma 5.7 can be used to lower bound $\mathcal{I}_w(t_0,\bar{t}\,)$. As $\mathcal{I}_w(t_0,\bar{t}\,) \geq \xi$ if there is *any* node u such that $L_u(t_0) - L_w(t_0) = d(u,w)\left(s-\frac{1}{2}\right)\kappa + \xi$, we have that

$$\begin{aligned}\mathcal{I}_w(t_0,\bar{t}\,) &\overset{(15)}{\geq} \max_{u\in V}\left\{L_u(t_0) - L_w(t_0) - \left(s-\frac{1}{2}\right)\kappa d(u,w)\right\}\\ &\geq \Psi_w^s(t_0)\\ &\overset{(32)}{>} \frac{\kappa}{2}C_s - 2\varepsilon\frac{\sigma+1}{(1-\varepsilon)\mu}\kappa C_s + \mathcal{I}_w(t_0,\bar{t}\,) - \frac{\kappa}{7\sigma}\\ &\geq \left(\frac{1}{2}-\frac{1}{7\sigma}\right)\kappa C_s - 2\varepsilon\frac{\sigma+1}{(1-\varepsilon)\mu}\kappa C_s + \mathcal{I}_w(t_0,\bar{t}\,).\end{aligned}$$

This result again leads to the Inequality (31), which contradicts Inequality (6). Thus, the system can never leave the legal state, which proves the claimed bound on the local skew. □

Note that this theorem also holds if each node v increases its logical clock value by the value R_v computed in the subroutine *setClockRate* at once instead of raising the logical clock rate: Theorem 5.10 is proved using Lemma 5.7 and Lemma 5.9. Clearly, increasing the clock values instantly is a more aggressive strategy and it is easy to see that Lemma 5.7 still holds. In Lemma 5.9, we found that $R_v = 0$ if the clock skew between two nodes v and w is sufficiently large, which implies that v increases its logical clock value at the hardware clock rate in this situation even if the nodes are allowed to increase the clock values instantaneously.

Since the base of the logarithm σ is the largest integer such that Inequality (6) holds (for a given $\mu \geq 14\varepsilon/(1-\varepsilon)$), it follows that $\sigma \in \Theta(\mu/\varepsilon)$. Thus, choosing

$\kappa \in \Theta((1+\mu)\mathcal{T} + \mu H_0)$ results in a local skew of

$$\mathcal{O}(((1+\mu)\mathcal{T} + \mu H_0) \log_{\mu/\varepsilon} D).$$

Recall that we can choose $\kappa \in \Theta(\mathcal{T})$ because we know upper bounds $\hat{\mathcal{T}} \in \mathcal{O}(\mathcal{T})$ and $\hat{\varepsilon} < 1$ on $\mathcal{T}$ and ε, respectively. If $\mu \in \Theta(\varepsilon)$ and $H_0 \in \mathcal{O}(\mathcal{T}/\mu) = \mathcal{O}(\mathcal{T}/\varepsilon)$, Theorem 5.10 states that the local skew is upper bounded by $\mathcal{O}(\mathcal{T} \log D)$. Note that choosing $\mu \in \Theta(\varepsilon)$ entails that the maximum logical clock rate β is upper bounded by $1 + \mathcal{O}(\varepsilon)$. If the logical clock rate is allowed to be larger than the hardware clock rate by a constant factor, that is, $\mu \in \Theta(1)$, and we choose $H_0 \in \mathcal{O}(\mathcal{T})$, the bound on the local skew reduces to $\mathcal{O}(\mathcal{T} \log_{1/\varepsilon} D)$.

6. *Complexity*

In this section, we discuss the cost of Algorithm $\mathcal{A}^{\text{opt}}$ with regard to several measures. In particular, we analyze how many messages need to be exchanged and also the (maximum) size of these messages. Moreover, we upper bound the number of bits that each node needs to store locally.

6.1. Message Complexity.

An essential optimization criterion is the frequency of communication required to sustain a given quality of synchronization. If resource consumption due to communication is critical, the average of this value over time, that is, the *amortized message frequency*, is highly relevant. Statement (*ii*) of Corollary 5.2 immediately yields that $\mathcal{A}^{\text{opt}}$ exhibits an amortized message frequency of $\Theta(1/H_0)$ at each node. The bound from Theorem 5.10 suggests to choose $H_0 \in \Theta(\hat{\mathcal{T}}/\mu)$, which for a minimal $\mu \in \Theta(\hat{\varepsilon})$ entails that the amortized message frequency is only $\Theta(\hat{\varepsilon}/\hat{\mathcal{T}})$.

In a short time period, however, a node v might receive $\Theta(\mathcal{G}/H_0)$ messages containing values $L^{\max}$, each larger by H_0 than the previous one, which cause v to send as many messages. Thus, the algorithm in the presented form does not guarantee a non-trivial lower bound on the message frequency. The message frequency could be bounded by adding another term in the order of $\Theta(\bar{H}_0)$ to κ and forcing nodes to wait at least until the progress of their hardware clocks is H_0 since they last sent a message. The price of this modification is that the bound on the global skew increases by $\Theta(\varepsilon D H_0)$ as the time it takes to propagate information through the whole system increases by $\mathcal{O}(D H_0)$ while nodes increase the estimate on $L^{\max}$ locally at their hardware clock rate. This results in a tunable trade-off between minimum message frequency and global skew as increasing the former inversely affects the latter. This is, up to constant factors, the best possible trade-off, since in the $\Theta(D H_0)$ time a pair of nodes at distance D may have to act without updates about each other's state a skew of $\Theta(\varepsilon D H_0)$ can be built up by manipulating the hardware clock rates.

6.2. Bit Complexity.

Another important property is the *bit complexity*, that is, the maximum number of bits that must be sent in a message. Since the same update information is sent to all neighbors at the same send event, we define that the bit complexity in our model is simply the maximum size of this message. Certainly, we cannot encode arbitrary real numbers; however, inaccuracies in the communicated values can simply be accounted for by increasing κ. Therefore, nodes are merely required to transfer rounded values to their neighbors. Note that the parameter $\mu \in \mathcal{O}(1)$ can be encoded as a constant times $1/n$, for a number

$n \in \mathbb{N}$. The other parameters, κ and H_0, can then be chosen (and encoded) as multiples of μ. Thus, any value that is bounded by a constant can be encoded using $\mathcal{O}(\log(1/\mu)) \subseteq \mathcal{O}(\log(1/\hat{\varepsilon}))$ bits. We will use this simple observation in the following.

In order to bound the bit complexity, we cannot send the unbounded clock values. Instead, nodes simply communicate the $\mathcal{O}((1+\mu)H_0)$ progress their clocks made since they last sent a message, which requires $\mathcal{O}(\log H_0 + \log(1/\mu))$ bits. However, since we have that $\kappa \in \Omega(\mu H_0)$, we might as well add another μH_0 to κ and discretize the sent values in steps of μH_0. Thus, for the clock value L_v only $\mathcal{O}(\log(1/\mu))$ bits are necessary.

Advantageously, the estimate $L_v^{\max}$ is a multiple of H_0, but unfortunately it may increase by $\Theta(\mathcal{G})$ in a single message, which could be encoded using $\Omega(\log(\mathcal{T}D/H_0))$ bits. In order to reduce the number of bits that are required to encode $L_v^{\max}$, we may limit the maximum increase of $L_v^{\max}$ that a node *informs* its neighbors about in a single message to $\lceil (1+\hat{\varepsilon})(1+\mu)/(1-\hat{\varepsilon})\rceil H_0 \in \mathcal{O}(H_0)$, which can be encoded using $\mathcal{O}(1)$ bits. If the actual value is larger, v stores the difference and informs its neighbors about the remaining increase in its subsequent messages. The intuition behind this is that $L^{\max}$ does not increase faster than at rate $1+\varepsilon$, therefore sending an update of at least $(1+\varepsilon)H_0/(1-\varepsilon)$ every $H_0/(1-\varepsilon)$ time is sufficient not to fall behind. If larger estimates are received than can be propagated immediately, then these estimates simply arrived quickly because the message delays were small. In the scenario where all messages are as slow as possible, this would not happen. Thus, the estimates of $\hat{L}^{\max}$ that slow nodes receive are still sufficiently large for Theorem 5.5 to hold. Since a node v sends updates of more than $(1+\mu)H_0$, its clock value L_v at the time of sending is still upper bounded by the estimate of $L^{\max}$ that it sends along. Hence, Lemma 5.7 still holds as v is not prohibited from increasing its clock value because its estimate $L_v^{\max}$ is too small, and thus also Theorem 5.10 remains correct. We conclude that Algorithm $\mathcal{A}^{\text{opt}}$ can be implemented with a bit complexity of $\mathcal{O}(\log(1/\mu)) \subseteq \mathcal{O}(\log(1/\hat{\varepsilon}))$.

Moreover, if we enforce that nodes wait between sending events for H_0 local time, we can further reduce this number. Since now we implicitly know of the progress of the hardware clock since the last message, we can encode the differences between the logical clock values more efficiently. To this end, the progress of the logical clock is described relative to the progress H_0 of the hardware clock, meaning that a difference of at most μH_0 has to be conveyed. Since we may round the progress to multiples of μH_0, this requires merely $\mathcal{O}(1)$ bits. From this and the previous observation that only $\mathcal{O}(1)$ bits are needed to update the estimate of the maximum clock value, we deduce that this variant of $\mathcal{A}^{\text{opt}}$ offers a constant bit complexity.

Note that if all messages must contain globally (or locally) unique node identifiers, $\mathcal{O}(\log|V|)$ (or $\mathcal{O}(\log\Delta)$, where Δ denotes the maximum node degree) additional bits are required.

6.3. SPACE COMPLEXITY. The *space complexity* of an algorithm is the maximum amount of memory that it requires to run. Since the logical clock value L_v grows indefinitely, we disregard it in our analysis of the space complexity of $\mathcal{A}^{\text{opt}}$. Furthermore, we will consider the amount of memory that the implementations of $\mathcal{A}^{\text{opt}}$ discussed in the previous section require.[10]

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

Each node v must store the estimated clock skew $L_v - L_v^w$ to each node $w \in \mathcal{N}_v$ and the estimated difference $L_v^{\max} - L_v$ to the maximum clock value. The value of $L_v - L_v^w$ is bounded by $\mathcal{O}(\kappa \log_{\mu/\varepsilon} D)$ for all neighbors. Assuming that we choose $H_0 \in \Theta(\mathcal{T}/\mu)$ and $\kappa \in \Theta(\mathcal{T})$, the rounding of clock values to multiples of μH_0 implies that the maximum memory requirement for these values is bounded by $\mathcal{O}(\Delta \log\log_{\mu/\varepsilon} D)$, where Δ denotes the maximum node degree. As $L_v^{\max} - L_v$ is bounded by $\mathcal{G} \in \mathcal{O}(\mathcal{T}D)$ and it is a multiple of H_0, it can be encoded using $\mathcal{O}(\log(\mathcal{T}D/H_0)) = \mathcal{O}(\log(\mu D))$ bits.

Furthermore, in order to properly adapt L_v^w, each node must store the amount of local time elapsed since the last message from $w \in \mathcal{N}_v$ was received as in the absence of better information v assumes that the neighbors' clocks run at the same rate as its own hardware clock. If the frequency of the hardware clock is f, i.e., roughly every $1/f$ time an event is triggered at a node v, this requires $\mathcal{O}(\log(f H_0))$ memory for each $w \in \mathcal{N}_v$.[11] However, the high accuracy of $1/f$ is not needed. Instead, a local counter of size $\mathcal{O}(\log(f\mu H_0))$ is used to generate events every $\Theta(\mathcal{T})$ local time, reducing the resolution to $\Theta(\mu H_0)$ and the necessary memory to $\mathcal{O}(1/\mu)$ bits per neighbor, at the price of adding another term in the order of $\mathcal{O}(\mu H_0)$ to κ. Thus, these values require $\mathcal{O}(\Delta \log \mu + \log(f\mathcal{T}))$ bits in total.

If the proposed technique to update the estimates on $L^{\max}$ only by constant multiples of H_0 is employed (in order to keep the bit complexity low), we have to keep track of the updates that have already occurred. Since nodes send identical messages to all neighbors, a node must store only another multiple of H_0 bounded by $\mathcal{O}(\mathcal{G})$, which requires $\mathcal{O}(\log(\mu D))$ bits. However, nodes may receive different estimates from different neighbors. If messages are sent every $\Theta(H_0)$ time, these values may drift apart at an (amortized) rate of 2ε for $\mathcal{O}(\mathcal{G})$ time, until the (local) estimates reach $L^{\max}$. Therefore, if these values are encoded relative to each other, $\mathcal{O}(\Delta \log(\varepsilon\mu D))$ additional bits are needed.

The same is true if messages are also triggered upon receiving messages, but nodes control the rate at which they forward estimates by means of their hardware clocks. Setting the maximal amortized update rate to a constant factor times the hardware clock rate (i.e., at least $(1+\hat{\varepsilon})(1+\mu)/(1-\hat{\varepsilon}) \in \mathcal{O}(1)$) ensures that estimates are forwarded fast enough, but locally drift apart at a rate of at most $\mathcal{O}(\varepsilon)$. Since the nodes with small clock values again receive information about larger clock values quickly enough, the algorithm preserves its asymptotic bounds on both the global and the local skew when following this approach.

Hence, adding all terms up, we see that $\mathcal{A}^{\text{opt}}$ needs to allocate in total at most $\mathcal{O}(\log(f\mathcal{T})+\log(\mu D)+\Delta(\log(1/\mu)+\log(\varepsilon\mu D)+\log\log_{\mu/\varepsilon} D))$ bits of memory.[12]

7. *Lower Bounds*

The lower bounds on both the global and the local skew are proved using *indistinguishability type arguments*. Concretely, we construct *indistinguishable executions*

[11] Of course, also hardware clocks do not offer a continuous time, rather they generate clock ticks at a high frequency f. Naturally, the clock granularity $1/f$ yields a trivial lower bound for the best synchronization that can be guaranteed. In particular, we need that $1/f \in \mathcal{O}(\mathcal{T})$.

[12] This notation is somewhat sloppy. To be formally correct, each summand has to be replaced by the maximum of the term itself and 1.

for any given synchronization algorithm $\mathcal{A}$ and any graph G in such a way that at least one of the executions causes large clock skews. Given two executions $\mathcal{E}$ and $\bar{\mathcal{E}}$ of an algorithm $\mathcal{A}$ on a graph G, let $H_v^{\mathcal{E}}(t)$ and $H_v^{\bar{\mathcal{E}}}(t)$ denote the hardware clock values of v at time t in $\mathcal{E}$ and $\bar{\mathcal{E}}$, respectively. The respective logical clock values are denoted by $L_v^{\mathcal{E}}(t)$ and $L_v^{\bar{\mathcal{E}}}(t)$. The following definition formalizes the concept of indistinguishable executions.

Definition 7.1 (*Indistinguishability of Executions*). We call $\mathcal{E}$ and $\bar{\mathcal{E}}$ *indistinguishable at node v until hardware clock time H*, if v observes the same message pattern with respect to its local time H_v in both $\mathcal{E}$ and $\bar{\mathcal{E}}$ until its hardware clock reaches H. More precisely, if v receives a message at a time t_r when $H_v^{\mathcal{E}}(t_r) \leq H$ in $\mathcal{E}$, it receives an identical message in $\bar{\mathcal{E}}$ at the time $\bar{t}_r$ when $H_v^{\bar{\mathcal{E}}}(\bar{t}_r) = H_v^{\mathcal{E}}(t_r)$, and vice versa. Note that in this situation v behaves the same way in $\mathcal{E}$ and $\bar{\mathcal{E}}$ until local time H, i.e., if $H_v^{\mathcal{E}}(t) = H_v^{\bar{\mathcal{E}}}(\bar{t}\,) \leq H$, it follows that $L_v^{\mathcal{E}}(t) = L_v^{\bar{\mathcal{E}}}(\bar{t}\,)$ and v sends the same messages (if any) at times t and $\bar{t}$ in $\mathcal{E}$ and $\bar{\mathcal{E}}$, respectively.

We base the construction of indistinguishable executions $\mathcal{E}$ and $\bar{\mathcal{E}}$ on a simple principle called *shifting* where both the clock rates and the message delays are "shifted" in such a way that all events occur at the same local times in $\mathcal{E}$ and $\bar{\mathcal{E}}$ [Lundelius Welch and Lynch 1984]. If it is clear from the context, we may omit the specification of the execution in the notation and write, for example, $H_v(t)$ instead of $H_v^{\mathcal{E}}(t)$. Throughout this section, we assume, without loss of generality, that all nodes are initialized at time 0, that is, all messages of the initial flooding activating the system have zero delay and are not considered in the following.

7.1. GLOBAL SKEW. In our model, we assumed that an algorithm only possesses estimates of ε and $\mathcal{T}$. If the estimates are inaccurate, κ is set to a value that is larger than necessary, which has a negative impact on the bound on the local skew. The following theorem gives a lower bound on the global skew that depends on the accuracy of both $\hat{\varepsilon}$ and $\hat{\mathcal{T}}$.

THEOREM 7.2. *Assume that a clock synchronization algorithm $\mathcal{A}$ is equipped with initial parameters $\hat{\varepsilon} \in (0, 1)$, and $\hat{\mathcal{T}} \in \mathbb{R}^+$ such that $c_1\hat{\mathcal{T}} \leq \mathcal{T} \leq \hat{\mathcal{T}}$ and $c_2\hat{\varepsilon} \leq \varepsilon \leq \hat{\varepsilon}$ for certain values $c_1, c_2 \in (0, 1]$. Define $\varrho := \min\{\varepsilon, (1-c_2\hat{\varepsilon})/c_1 - 1\} \in [-\varepsilon, \varepsilon]$. If algorithm $\mathcal{A}$ is bound to satisfy Condition* (1)*, it cannot avoid a global skew of at least*

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

on any graph G of diameter D.

PROOF. For the sake of simplicity, we formally allow relative clock drifts of $\varepsilon + \tilde{\varepsilon}$, where $\tilde{\varepsilon}$ is infinitesimally small.[13]

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

[13] The same result could be obtained, for example, by replacing ε by $\varepsilon - \tilde{\varepsilon}$, proving a bound of $(1 + \varrho - \mathcal{O}(\tilde{\varepsilon}))D\mathcal{T}$, and taking the limit $\tilde{\varepsilon} \to 0$.

and $c_1(1+\rho)/(1-\varepsilon') \leq 1$, implying that

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

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

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

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

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

No node v can increase its logical clock at a rate lower than its hardware clock rate, as otherwise it violates Condition (1) in execution $\mathcal{E}_1$. Likewise, v cannot increase its logical clock faster than its hardware clock because the envelope condition would be violated in execution $\mathcal{E}_2$. Thus, in both executions it must hold that $L_v(t) = H_v(t)$ at all times t.

Assume now that $\mathcal{T}$ is the correct delay uncertainty and ε is the correct maximum clock drift. Consider the following execution:

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

Note that execution $\mathcal{E}_1$ and $\mathcal{E}_3$, and hence also $\mathcal{E}_2$ and $\mathcal{E}_3$, are indistinguishable at each node $v \in V$ by construction. It remains to verify that $\mathcal{E}_3$ is a legal execution. Since $\varrho \in [-\varepsilon, \varepsilon]$ and a clock drift of $\varepsilon + \tilde{\varepsilon}$ is allowed, the clock drifts of all clocks are in the legal range. As far as the message delays are concerned, we have at all times $t \leq t_0$ that $H_w(t) - H_v(t) = \tilde{\varepsilon}\, t/D \in [0, (1+\varrho)\mathcal{T}] = [0, (1-\varepsilon')\mathcal{T}']$ if $d(v_0, w) = d(v_0, v) - 1$. First, consider a message sent from v to w. If $H_w(t) - H_v(t) = (1-\varepsilon')\mathcal{T}'$, then the message delay is set to zero, which ensures that w "sees" exactly a difference of $(1-\varepsilon')\mathcal{T}'$. If $H_w(t) - H_v(t) = 0$, the message must be delayed. However, since the hardware clock rate of each node is at least $1+\varrho$, it takes at most $(1-\varepsilon')\mathcal{T}'/(1+\varrho) = \mathcal{T}$ time for w to reach the hardware clock value $H_v + (1-\varepsilon')\mathcal{T}'$. Thus, in case of $d(v_0, w) = d(v_0, v) - 1$, the message delays are

always in the range $[0, \mathcal{T}]$. If w sends a message to v, the same arguments apply, but, in this case, we need that the message delay is set to zero if $H_w(t) - H_v(t) = 0$ and at most $(1 - \varepsilon')\mathcal{T}'/(1 + \varrho) = \mathcal{T}$ if $H_w(t) - H_v(t) = (1 - \varepsilon')\mathcal{T}'$. Note that if $d(v_0, w) = d(v_0, v)$, then $H_v(t) = H_w(t)$ as in the other two executions, and the message delay remains zero. Finally, the message delays remain in the range $[0, \mathcal{T}]$ at any time $t > t_0$, because all clocks run at the same rate, that is, the differences between the hardware clock values do not change.

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

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

which proves the stated lower bound on the global skew of any algorithm $\mathcal{A}$ that satisfies Condition (1). □

We can conclude from this theorem that the estimates of $\mathcal{T}$ and ε must be extremely accurate in order guarantee a better bound than $(1+\varepsilon)D\mathcal{T}$. However, even if the exact values are known, a global skew of $(1-\varepsilon)D\mathcal{T}$ cannot be prevented subject to the condition that the logical clock values must be within a linear envelope of real time.

COROLLARY 7.3. *No clock synchronization algorithm without knowledge of a lower bound on ε can avoid a global skew of $D\mathcal{T}$. Furthermore, no algorithm without knowledge of bounds on $\mathcal{T}$ stronger than $\mathcal{T} \in [(1 - \varepsilon)\hat{\mathcal{T}}/(1 + \varepsilon), \hat{\mathcal{T}}]$ can achieve a better bound on the global skew than $(1 + \varepsilon)D\mathcal{T}$.*

This corollary and Theorem 5.5 imply that $\mathcal{A}^{\text{opt}}$ is essentially optimal as far as the global skew is concerned. What is more, it can be shown that a global skew of $D\mathcal{T}/2$ cannot be prevented even if the restriction that the algorithm must satisfy Condition (1) is dropped [Biaz and Lundelius Welch 2001]. Thus, the bound on the global skew of $\mathcal{A}^{\text{opt}}$ is roughly a factor of two worse than the bound on the global skew of any algorithm whose behavior is not constrained by any additional restrictions.

7.2. LOCAL SKEW. The concept of an *extended execution* will be useful when proving the bounds on the local skew.

Definition 7.4 (*Extended Executions*). Given an execution $\mathcal{E}$ running from time t_1 to t_2, $\mathcal{E}$ can be extended by specifying hardware clock rates and message delays in a time interval $[t_2, t_3]$. We will refer to this extension as an execution $\mathcal{E}'$ running from time t_2 until time t_3. Note that $\mathcal{E}'$ inherits the state of the system at time t_2, that is, the state of all nodes and any pending messages sent in $\mathcal{E}$ that did not reach their destination until time t_2.

The remaining lower bounds also exploit that specific executions cannot be distinguished. The following lemma, which is a variant of a lemma presented in the original work that introduced the problem of bounding the clock skews between neighboring nodes [Fan and Lynch 2004], states that for a specific execution $\mathcal{E}$ there is an indistinguishable execution $\bar{\mathcal{E}}$ such that the logical clock of v advances more quickly in $\bar{\mathcal{E}}$ than in $\mathcal{E}$, but the clock of w does not. In the lemma, we consider special executions for which the hardware clock rates and the message delays are always

within specific bounds. These executions, which we refer to as *framed executions*, are defined as follows.

Definition 7.5 (*Framed Executions*). Given $\varphi \in [0, 1/2]$, a *φ-framed* execution is any execution such that the hardware clock rates always lie in the interval $[1, 1+\varepsilon]$ and all message delays are in the range $[\varphi\mathcal{T}, (1-\varphi)\mathcal{T}]$.

LEMMA 7.6. *Fix any clock synchronization algorithm, any graph, an arbitrary pair of nodes $v, w \in V$ and some $\varphi \in [0, 1/(2(1+\varepsilon))]$. Given a φ-framed execution $\mathcal{E}_0$ that ends at a time $t_{\mathcal{E}_0}$, this execution can be extended by a φ-framed execution $\mathcal{E} = \mathcal{E}(\mathcal{E}_0, v, w, \varphi)$ with the following property. For any pair of nodes $v', w' \in V$ on a shortest path from v to w such that $d(v, v') < d(v, w')$ and for any time $t_{\mathcal{E}} \geq t_{\mathcal{E}_0} + (1+\varepsilon)(1-2(1+\varepsilon)\varphi)d(v', w')\mathcal{T}/\varepsilon$, $\mathcal{E}$ can be modified into the φ-framed execution $\bar{\mathcal{E}} = \bar{\mathcal{E}}(\mathcal{E}, v', w')$ such that at time $t_{\bar{\mathcal{E}}} := t_{\mathcal{E}} - (1-2(1+\varepsilon)\varphi)d(v', w')\mathcal{T}$ we have that $L^{\bar{\mathcal{E}}}_{v'}(t_{\bar{\mathcal{E}}}) = L^{\mathcal{E}}_{v'}(t_{\mathcal{E}})$ and $L^{\bar{\mathcal{E}}}_{w'}(t_{\bar{\mathcal{E}}}) = L^{\mathcal{E}}_{w'}(t_{\bar{\mathcal{E}}})$.*

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

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

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

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

In order to finish the proof it remains to show that $\bar{\mathcal{E}}$ is indeed a φ-framed execution, that is, all hardware clock rates are in the range $[1, 1+\varepsilon]$ and message delays are in the range $[\varphi\mathcal{T}, (1-\varphi)\mathcal{T}]$. The hardware clock rate of each node at any point in time is between 1 and $1+\varepsilon$ and thus always in the legal range. Hence, we have to show that all messages received by some node $u \in V$ arrive after at least $\varphi\mathcal{T}$ and at most $(1-\varphi)\mathcal{T}$ time. Any message arriving immediately at time $t_{\mathcal{E}_0}$ cannot violate these bounds because $\mathcal{E}_0$ is a φ-framed execution. Given a message sent from a node $u_s \in V$ to a node $u_r \in \mathcal{N}_{u_s}$ that arrives later than $t_{\mathcal{E}_0}$ in $\mathcal{E}$, let t_s and t_r denote the times when the message is sent and received, respectively, in execution $\mathcal{E}$ (or $\mathcal{E}_0$), and let $\bar{t}_s$ and $\bar{t}_r$ be the corresponding times in execution $\bar{\mathcal{E}}$ (or, again, $\mathcal{E}_0$).

Starting at time t', the differences between the hardware clock values of neighbors gradually shift in $\bar{\mathcal{E}}$ compared to $\mathcal{E}$, until these shifts attain their maximum at time

$t_{\bar{\mathcal{E}}}$. Inserting the definitions, we see that at this time we have that

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

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

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

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

$$t_r - t_s - \left(H_{u_r}^{\bar{\mathcal{E}}}(t) - H_{u_r}^{\mathcal{E}}(t) - \left(H_{u_s}^{\bar{\mathcal{E}}}(t) - H_{u_s}^{\mathcal{E}}(t)\right)\right) \in [(1+\varepsilon)\varphi\mathcal{T}, (1-(1+\varepsilon)\varphi)\mathcal{T}]. \tag{33}$$

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

$$H_{u_r}^{\bar{\mathcal{E}}}(\bar{t}_r) - H_{u_r}^{\mathcal{E}}(\bar{t}_r) - \left(H_{u_r}^{\bar{\mathcal{E}}}(\bar{t}_s) - H_{u_r}^{\mathcal{E}}(\bar{t}_s)\right) \in [0, \varepsilon(\bar{t}_r - \bar{t}_s)]. \tag{34}$$

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

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

Inserting Bound (33) and Bound (34) into this equation, we obtain

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

which completes the proof. □

This technique allows us to enforce (linearly) growing average clock skews on certain paths at the expense of exponentially shrinking the path lengths. The subsequent theorem gives a lower bound on the local skew of $\Omega(\mathcal{T}\log_b D)$, where the base b depends on α, β, and ε. Recall that α and β denote the minimum and the maximum logical clock rate, respectively.

THEOREM 7.7. *Define* $b := \lceil 2(\beta-\alpha)/(\alpha\varepsilon)\rceil$. *No clock synchronization algorithm* $\mathcal{A}$ *can prevent a local skew of*

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

on any graph G *of diameter* D.

PROOF. Define $D' := b^{\lfloor\log_b D\rfloor} \leq D$. We claim that for any k, where $0 \leq k \leq \log_b D'$, there are two nodes v_k and w_k at distance $d(v_k, w_k) = D'/b^k$ such that the

clock skew between these nodes at the time $t_{\bar{\mathcal{E}}_k}$ in an execution $\bar{\mathcal{E}}_k$ is

$$L_{v_k}^{\bar{\mathcal{E}}_k}(t_{\bar{\mathcal{E}}_k}) - L_{w_k}^{\bar{\mathcal{E}}_k}(t_{\bar{\mathcal{E}}_k}) \geq \frac{k+1}{2}\alpha d(v_k, w_k)\mathcal{T}. \tag{35}$$

We prove our claim by induction, starting at $k = 0$. Consider any two nodes v_0 and w_0 at distance $d(v_0, w_0) = D'$ and let $\emptyset$ denote the "empty" execution that immediately ends at time 0. Define $\mathcal{E}_0$ to be the execution $\mathcal{E}(\emptyset, v_0, w_0, 0)$ as in Lemma 7.6 and set $t_\mathcal{E} := (1+\varepsilon)d(v_0, w_0)\mathcal{T}/\varepsilon$. In accordance with the lemma, there is an execution $\bar{\mathcal{E}}_0$ such that at the time $t_{\bar{\mathcal{E}}_0} = d(v_0, w_0)\mathcal{T}/\varepsilon$ it holds that $L_{v_0}^{\mathcal{E}_0}(t_{\mathcal{E}_0}) = L_{v_0}^{\bar{\mathcal{E}}_0}(t_{\bar{\mathcal{E}}_0})$ and $L_{w_0}^{\mathcal{E}}(t_{\bar{\mathcal{E}}_0}) = L_{w_0}^{\bar{\mathcal{E}}_0}(t_{\bar{\mathcal{E}}_0})$. Recall that all nodes are initialized immediately at time 0. Since the minimum clock rate is α, we have that

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

Thus, in one of the executions, we must have a skew of at least $\alpha d(v_0, w_0)\mathcal{T}/2$ between v_0 and w_0 at time $t_{\bar{\mathcal{E}}_0}$. Renaming the respective execution to $\bar{\mathcal{E}}_0$ and switching the roles of v_0 and w_0, if necessary, proves Inequality (35) for $k = 0$.

Assume that the claim is true for k, where $k < \log_b D'$. Given such an execution $\bar{\mathcal{E}}_k$, we end it at time $t_{\bar{\mathcal{E}}_k}$ and extend it by the execution $\mathcal{E}_{k+1} := \mathcal{E}(\bar{\mathcal{E}}_k, v_k, w_k, 0)$ from Lemma 7.6. Set $t_{\mathcal{E}_{k+1}} := t_{\bar{\mathcal{E}}_k} + (1+\varepsilon)D'\mathcal{T}/(\varepsilon b^{k+1})$ and $t_{\bar{\mathcal{E}}_{k+1}} := t_{\bar{\mathcal{E}}_k} + D'\mathcal{T}/(\varepsilon b^{k+1})$. The clock skew between the nodes v_k and w_k at time $t_{\bar{\mathcal{E}}_{k+1}}$ is at least

$$\begin{aligned} &L_{v_k}^{\mathcal{E}_{k+1}}(t_{\bar{\mathcal{E}}_{k+1}}) - L_{w_k}^{\mathcal{E}_{k+1}}(t_{\bar{\mathcal{E}}_{k+1}}) \\ &\quad\geq \left(L_{v_k}^{\bar{\mathcal{E}}_k}(t_{\bar{\mathcal{E}}_k}) + \alpha(t_{\bar{\mathcal{E}}_{k+1}} - t_{\bar{\mathcal{E}}_k})\right) - \left(L_{w_k}^{\bar{\mathcal{E}}_k}(t_{\bar{\mathcal{E}}_k}) + \beta(t_{\bar{\mathcal{E}}_{k+1}} - t_{\bar{\mathcal{E}}_k})\right) \\ &\quad\overset{(35)}{\geq} \frac{k+1}{2}\alpha d(v_k, w_k)\mathcal{T} - \frac{\beta-\alpha}{\varepsilon}\frac{D'}{b^{k+1}}\mathcal{T} \\ &\quad\geq \frac{k+1}{2}\alpha d(v_k, w_k)\mathcal{T} - \frac{\alpha b}{2}\frac{d(v_k, w_k)}{b}\mathcal{T} \\ &\quad= \frac{k}{2}\alpha d(v_k, w_k)\mathcal{T}. \end{aligned}$$

Consequently, there must be two nodes v_{k+1} and w_{k+1} on a shortest path from v_k to w_k for which $d(v_k, v_{k+1}) < d(v_k, w_{k+1})$, $d(v_{k+1}, w_{k+1}) = d(v_k, w_k)/b = D'/b^{k+1}$, and

$$L_{v_{k+1}}^{\mathcal{E}_{k+1}}(t_{\bar{\mathcal{E}}_{k+1}}) - L_{w_{k+1}}^{\mathcal{E}_{k+1}}(t_{\bar{\mathcal{E}}_{k+1}}) \geq \frac{k}{2}\alpha d(v_{k+1}, w_{k+1})\mathcal{T}. \tag{36}$$

We define that $\mathcal{E}_{k+1}$ ends at the time $t_{\mathcal{E}_{k+1}}$ and apply Lemma 7.6 to obtain the execution $\bar{\mathcal{E}}_{k+1} := \bar{\mathcal{E}}(\mathcal{E}_{k+1}, v_{k+1}, w_{k+1}, 0)$ for which it holds that $L_{v_{k+1}}^{\mathcal{E}_{k+1}}(t_{\mathcal{E}_{k+1}}) = L_{v_{k+1}}^{\bar{\mathcal{E}}_{k+1}}(t_{\bar{\mathcal{E}}_{k+1}})$ and $L_{w_{k+1}}^{\mathcal{E}_{k+1}}(t_{\bar{\mathcal{E}}_{k+1}}) = L_{w_{k+1}}^{\bar{\mathcal{E}}_{k+1}}(t_{\bar{\mathcal{E}}_{k+1}})$. Since it holds that

$$L_{w_{k+1}}^{\mathcal{E}_{k+1}}(t_{\mathcal{E}_{k+1}}) - L_{w_{k+1}}^{\mathcal{E}_{k+1}}(t_{\bar{\mathcal{E}}_{k+1}}) \overset{(2)}{\geq} \alpha d(v_{k+1}, w_{k+1})\mathcal{T}, \tag{37}$$

the clock skew at time $t_{\bar{\mathcal{E}}_{k+1}}$ between v_{k+1} and w_{k+1} in execution $\bar{\mathcal{E}}_{k+1}$ is

$$\begin{aligned} L_{v_{k+1}}^{\bar{\mathcal{E}}_{k+1}}(t_{\bar{\mathcal{E}}_{k+1}}) - L_{w_{k+1}}^{\bar{\mathcal{E}}_{k+1}}(t_{\bar{\mathcal{E}}_{k+1}}) &= L_{v_{k+1}}^{\mathcal{E}_{k+1}}(t_{\mathcal{E}_{k+1}}) - L_{w_{k+1}}^{\mathcal{E}_{k+1}}(t_{\bar{\mathcal{E}}_{k+1}}) \\ &\overset{(37)}{\geq} \alpha d(v_{k+1}, w_{k+1})\mathcal{T} + L_{v_{k+1}}^{\mathcal{E}_{k+1}}(t_{\bar{\mathcal{E}}_{k+1}}) - L_{w_{k+1}}^{\mathcal{E}_{k+1}}(t_{\bar{\mathcal{E}}_{k+1}}) \\ &\overset{(36)}{\geq} \frac{k+2}{2}\alpha d(v_{k+1}, w_{k+1})\mathcal{T}, \end{aligned}$$

which proves the claim. If $k = \lfloor \log_b D \rfloor$, the distance between the considered nodes is 1, i.e., v_k and w_k are neighboring nodes. The local skew is thus at least $(\lfloor \log_b D \rfloor + 1)\alpha\mathcal{T}/2$. □

If we demand that the logical clocks run roughly at the same rates as the hardware clocks, for example, $\alpha \in 1 - \mathcal{O}(\varepsilon)$ and $\beta \in 1 + \mathcal{O}(\varepsilon)$, we get that $b \in \mathcal{O}(1)$ and thus a lower bound of $\Omega(\mathcal{T} \log D)$, which matches the upper bound of algorithm $\mathcal{A}^{\text{opt}}$ when $\mu \in \Theta(\varepsilon)$ and $H_0 \in \mathcal{O}(\mathcal{T}/\varepsilon)$. Similarly, if we allow a logical clock rate that is a constant times larger than real time, that is, $\beta \in \Theta(1)$, the lower bound reduces to $\Omega(\mathcal{T} \log_{1/\varepsilon} D)$. Algorithm $\mathcal{A}^{\text{opt}}$ guarantees an upper bound on the local skew of $\mathcal{O}(\mathcal{T} \log_{1/\varepsilon} D)$ when choosing $\mu \in \Theta(1)$ and $H_0 \in \mathcal{O}(\mathcal{T})$. More generally, we get the following result.

COROLLARY 7.8. *If $\kappa \in \mathcal{O}(\mathcal{T})$, Algorithm $\mathcal{A}^{\text{opt}}$ achieves an asymptotically optimal local skew of $\Theta(\mathcal{T} \log_{\mu/\varepsilon} D)$.*

A slightly more careful analysis yields a bound which is stronger by an asymptotic factor of 2 [Lenzen et al. 2009a], revealing that the approximation ratio of $\mathcal{A}^{\text{opt}}$ with respect to the local skew tends to (at most) $2\hat{\mathcal{T}}/\mathcal{T}$ if $\hat{\varepsilon} \to 0$ and $D \to \infty$.

What is more, Theorem 5.10 is shown by proving that the legal state is never violated, whereas in the proof of Theorem 7.7 we iteratively constructed increasing average skews between nodes of exponentially decreasing distance. Hence, we see that $\mathcal{A}^{\text{opt}}$ features an asymptotically optimal gradient property as defined by Fan and Lynch [2004]. We note that this result holds irrespective of α since $\mathcal{A}^{\text{opt}}$ can exploit smaller values of α by simply reducing the speed of the logical clocks by an appropriate factor.

COROLLARY 7.9. *If $\beta - \alpha \in \mathcal{O}(1)$, the best worst-case clock skew between nodes at distance d any algorithm can achieve is $\Theta(\alpha\mathcal{T}d(1 + \log_{(\beta-\alpha)/(\alpha\varepsilon)}(D/d)))$.*

It is further important to see that the maximum clock skew among all neighboring nodes can be $\Omega(\alpha\mathcal{T} \log_b D)$ for more than a constant period of time. The proof of Theorem 7.7 reveals that, for example, for $k = (\log_b D')/2$, the average clock skew on a path of length $\Theta(\sqrt{D})$ is half the local skew that is built up between two neighbors until the end of the constructed execution. Since it takes $\Theta(\mathcal{T}\sqrt{D})$ time to increase the clock skew to $(\lfloor \log_b D \rfloor + 1)\alpha\mathcal{T}/2$, for $\Theta(\mathcal{T}\sqrt{D})$ time there are always some neighbors with a clock skew of $\Omega(\alpha\mathcal{T} \log_b D)$. More generally, for any constant $c < 1$, the average clock skew on some path, and thus also the maximum clock skew among all neighboring nodes, is $\Omega((\alpha\mathcal{T} \log_b D)/c)$ for $\Theta(D^{1-c}\,\mathcal{T})$ time.

Moreover, it is evident from the proof of Theorem 7.7 that the same asymptotic bound holds if the nodes are allowed to reduce their clock rates arbitrarily, even to or below zero, as long as the *average* clock rate in an interval of length $\Theta(D^{1-c}\,\mathcal{T})$, for some $c < 1$, is at least α. This observation implies that it is not a severe limitation

that the progress rate of all clocks is always at least α. Note that if the clocks are allowed to stand still for $\Theta(D\mathcal{T})$ time, a simple synchronizer [Awerbuch 1985], which trivially guarantees a bound of $\Theta(\alpha\mathcal{T})$ on the local skew, can be used instead of a clock synchronization protocol.

7.3. LOCAL SKEW IF CLOCK RATES ARE UNBOUNDED. Theorem 7.7 leaves the question open whether a smaller local skew can be maintained if clock rates are not bounded by constants ($\beta \in \omega(1)$) or clocks may even jump instantaneously ($\beta = \infty$). The lower bound in Fan and Lynch [2004] was proved by showing that, in specific executions, if any node has an average clock rate of ρ over a time period of $\Omega(\mathcal{T})$, a local skew of $\Omega(\rho\mathcal{T})$ can be enforced. Therefore, a clock skew of $\Omega(\alpha\mathcal{T}\log_{1/\varepsilon} D/(\log_{1/\varepsilon}\log_{1/\varepsilon} D))$ is inevitable. We will now improve this lower bound to $\Theta(\alpha\mathcal{T}\log_{1/\varepsilon} D)$, which implies that nothing can be gained by relaxing Inequality (2) further than $\beta - \alpha \in \mathcal{O}(1)$. Thus, no algorithm that allows the clock values to increase instantaneously by any amount can achieve a better asymptotic bound on the global or the local skew than $\mathcal{A}^{\text{opt}}$, even if $\mathcal{A}^{\text{opt}}$ only increases its clock values at bounded rates.

In order to prove the theorem, we require a few more tools. Similarly to Fan and Lynch [2004], we need to upper bound the clock rates an algorithm may utilize without giving an adversary the means to introduce a large local skew directly. To this end, we show that any φ-framed execution permits to introduce $\varphi\mathcal{T}$ hardware clock skew unnoticeably at arbitrary nodes.

LEMMA 7.10. *Fix any clock synchronization algorithm and any graph. Let a φ-framed execution $\mathcal{E}$, a time $t \geq \varphi\mathcal{T}/\varepsilon$, and a node $v \in V$ be given, and define that $t' := t - \varphi\mathcal{T}/(1+\varepsilon)$. We can indistinguishably modify $\mathcal{E}$ into an execution $\bar{\mathcal{E}}$ for which it holds that $L_v^{\bar{\mathcal{E}}}(t) = L_v^{\mathcal{E}}(t')$ and $L_w^{\bar{\mathcal{E}}}(t) = L_w^{\mathcal{E}}(t)$ for any node $w \in V \setminus \{v\}$.*

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

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

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

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

This statement implies that if an algorithm has a clock rate of $\Omega((\alpha\log_{1/\varepsilon} D)/\varphi)$ (on average) in a φ-framed execution for $\varphi\mathcal{T}/(1+\varepsilon)$ time, it can be tricked into building up a local skew of $\Omega(\alpha\mathcal{T}\log_{1/\varepsilon} D)$.

Combining Lemmas 7.6 and 7.10, we obtain the key lemma to the second lower bound on the local skew. If an algorithm uses large logical clock rates to reduce skews quickly, it trades a faster decrease of path lengths for a larger increase of the average skew on such paths.

LEMMA 7.11. *Fix any clock synchronization algorithm and any graph of diameter $D \geq 1/\varepsilon$. Assume that $1/\varepsilon$ is an integer (in particular, $\varepsilon \leq 1/2$), and let $X := \lceil 12 \log(8/\varepsilon) \rceil$. Set $\varphi_\varepsilon := \varepsilon/(2(1+\varepsilon))$ and $\zeta := 1 - \varepsilon - 12/X \geq 1/4$. Let a φ_ε-framed execution $\mathcal{E}_0$ ending at time $t_{\mathcal{E}_0} \geq \mathcal{T}/2$ be given such that a shortest path $p := v_0, \ldots, v_k$ with $L_{v_0}(t_{\mathcal{E}_0}) - L_{v_k}(t_{\mathcal{E}_0}) \geq \lambda \alpha \mathcal{T} k$ for some $\lambda \in \mathbb{R}$ exists, where X/ε^n divides $d(v_0, v_k) = k$ for some integer $n \geq 2 + \log_{1/\varepsilon} \log_{1/\varepsilon} D$.*

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

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

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

PROOF. We extend $\mathcal{E}_0$ by the execution $\mathcal{E} = \mathcal{E}(\mathcal{E}_0, v_0, v_k, \varphi_\varepsilon)$ from Lemma 7.6. Define for $m \in \{1, \ldots, n\}$ that $t_m := t_{\mathcal{E}_0} + \varepsilon^{m-1}(1-\varepsilon)\mathcal{T}k/X$. We make a case differentiation. First, assume that we have $L^{\mathcal{E}}_{v_0}(t_1) - L^{\mathcal{E}}_{v_i}(t_1) \geq (\lambda - 12/X)\alpha \mathcal{T} i$ for some $i \geq \varepsilon k/X$. Hence, there must be two nodes $v, w \in p$ at distance $d(v, w) = \varepsilon k/X$ such that

$$L^{\mathcal{E}}_v(t_1) - L^{\mathcal{E}}_w(t_1) \geq \left(\lambda - \frac{12}{X}\right) \alpha \mathcal{T} d(v, w) \tag{38}$$

and $d(v_0, v) < d(v_0, w)$. Define $t_{\mathcal{E}} := t_1 + (1-\varepsilon)d(v, w)\mathcal{T} = t_{\mathcal{E}_0} + (1+\varepsilon)(1-\varepsilon)d(v, w)\mathcal{T}/\varepsilon$ as the time when $\mathcal{E}$ ends. Observe that $1 - \varepsilon = 1 - 2(1+\varepsilon)\varphi_\varepsilon$. Thus, due to Lemma 7.6, we can modify $\mathcal{E}$ into the φ_ε-framed execution $\bar{\mathcal{E}} = \bar{\mathcal{E}}(\mathcal{E}, v, w, \varphi_\varepsilon)$ such that $L^{\bar{\mathcal{E}}}_v(t_1) = L^{\mathcal{E}}_v(t_{\mathcal{E}})$ and $L^{\bar{\mathcal{E}}}_w(t_1) = L^{\mathcal{E}}_w(t_1)$. It follows that

$$\begin{aligned} L^{\bar{\mathcal{E}}}_v(t_1) - L^{\bar{\mathcal{E}}}_w(t_1) &= L^{\mathcal{E}}_v(t_{\mathcal{E}}) - L^{\mathcal{E}}_v(t_1) + (L^{\mathcal{E}}_v(t_1) - L^{\mathcal{E}}_w(t_1)) \\ &\overset{(2,38)}{\geq} \alpha(t_{\mathcal{E}} - t_1) + \left(\lambda - \frac{12}{X}\right) \alpha \mathcal{T} d(v, w) \\ &= (\lambda + \zeta)\, \alpha \mathcal{T} d(v, w). \end{aligned}$$

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

$$L^{\mathcal{E}}_v(t_{\mathcal{E}}) - L^{\mathcal{E}}_w(t_{\bar{\mathcal{E}}}) \geq (\lambda + m\zeta)\, \alpha \mathcal{T} d(v, w) \tag{39}$$

holds, where $t_{\bar{\mathcal{E}}} := t_{\mathcal{E}} - \varepsilon^m(1-\varepsilon)\mathcal{T}k/X = t_{\mathcal{E}} - (1-\varepsilon)d(v, w)\mathcal{T}$. Since $t_{\mathcal{E}}$ is sufficiently large, Lemma 7.6 states that if $\mathcal{E}$ ends at time $t_{\mathcal{E}}$, it can be changed into the φ_ε-framed execution $\bar{\mathcal{E}} = \bar{\mathcal{E}}(\mathcal{E}, v, w, \varphi_\varepsilon)$ where

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

Third and last, assume that none of the former is true. Consider the following set of pairs of times and nodes $\{(t^j, v^j) \mid j \in \{0, \ldots, j_{\max} := (1-\varepsilon)(n-1)/\varepsilon^2\}\}$.Define

$v^0 := v_{i(0)}$, where $i(0) := k - (1-\varepsilon^{n-1})k/X$. Let $m_j := 2 + \lfloor \varepsilon^2 j/(1-\varepsilon) \rfloor$ and $i_{m_j} := i(0) + (1-\varepsilon)k/(\varepsilon^2 X) \sum_{m=2}^{m_j-1} \varepsilon^m$. Set

$$\begin{aligned} t^j &:= t_{m_j-1} - \left(j - \frac{(1-\varepsilon)(m_j-2)}{\varepsilon^2} \right) \frac{\varepsilon^{m_j}(1-\varepsilon)\mathcal{T}k}{X} \\ &\geq t_{m_{j_{\max}}-1} = t_n > t_{\varepsilon_0} \end{aligned}$$

and $v^j := v_{i(j)}$, where

$$i(j) := i_{m_j-1} + \left(j - (1-\varepsilon)(m_j-2)/\left(\varepsilon^2 X\right) \right) \varepsilon^{m_j} k \leq i(j_{\max}) = i_{n+1} = k.$$

These cumbersome choices of t^j and v^j ensure that, as the second case does not apply, we have

$$L^{\mathcal{E}}_{v^j}(t^j) - L^{\mathcal{E}}_{v^{j+1}}(t^{j+1}) \overset{(39)}{<} (\lambda + m_j\zeta)\alpha\mathcal{T}d(v^j, v^{j+1})$$

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

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

$$\begin{aligned} L^{\mathcal{E}}_{v^0}(t_1) - L^{\mathcal{E}}_{k}(t_n) &= \sum_{j=0}^{j_{\max}-1} \left(L^{\mathcal{E}}_{v^j}(t^j) - L^{\mathcal{E}}_{v^{j+1}}(t^{j+1}) \right) \\ &\overset{(39)}{<} \sum_{m=2}^{n} \sum_{\{j \mid m_j = m\}} (\lambda + m_j\zeta)\alpha\mathcal{T}d(v^j, v^{j+1}) \\ &= \lambda\alpha\mathcal{T}d(v^0, v_k) + \frac{\zeta\alpha\mathcal{T}k}{X} \sum_{m=2}^{n} \sum_{l=1}^{(1-\varepsilon)/\varepsilon^2} m\varepsilon^m \\ &< \lambda\alpha\mathcal{T}d(v^0, v_k) + (1-\varepsilon)\frac{\zeta\alpha\mathcal{T}k}{X} \sum_{m=0}^{\infty} (m+2)\varepsilon^m \\ &< \lambda\alpha\mathcal{T}d(v^0, v_k) + (1-\varepsilon)^2\frac{\alpha\mathcal{T}k}{X}\frac{2}{(1-\varepsilon)^2} \\ &= \lambda\alpha\mathcal{T}d(v^0, v_k) + \frac{2\alpha\mathcal{T}k}{X}. \end{aligned} \tag{40}$$

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

$$\begin{aligned}
L^{\mathcal{E}}_{v_k}(t_n) - L^{\mathcal{E}}_{v_k}(t_{\mathcal{E}_0}) &= L^{\mathcal{E}}_{v_k}(t_n) - L^{\mathcal{E}}_{v^0}(t_1) + \left(L^{\mathcal{E}}_{v^0}(t_1) - L^{\mathcal{E}}_{v^0}(t_{\mathcal{E}_0})\right) \\
&\quad + \left(L^{\mathcal{E}}_{v^0}(t_{\mathcal{E}_0}) - L^{\mathcal{E}}_{v_0}(t_{\mathcal{E}_0})\right) + \left(L^{\mathcal{E}}_{v_0}(t_{\mathcal{E}_0}) - L^{\mathcal{E}}_{v_k}(t_{\mathcal{E}_0})\right) \\
&\overset{(40)}{>} \left(-\lambda d(v^0, v_k) - \frac{2k}{X} - \left(\lambda - \frac{12}{X}\right) d\left(v_0, v^0\right)\right. \\
&\quad \left. + \lambda d\left(v_0, v^{j_{\max}}\right)\right)\alpha\mathcal{T} \\
&> \left(12\left(1 - \frac{1}{X}\right) - 2\right)\frac{\alpha\mathcal{T}k}{X} \\
&\overset{X>12}{>} \frac{9\alpha\mathcal{T}k}{X} \\
&\geq \frac{9\alpha(t_n - t_{\mathcal{E}_0})}{\varepsilon}\log_{1/\varepsilon} D,
\end{aligned}$$

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

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

As $\mathcal{E}_0$ extended by $\mathcal{E}$ meets the prerequisites of Lemma 7.10 for $t > t_{\mathcal{E}_0} \geq \mathcal{T}/2 = (1 + \varepsilon)\varphi_\varepsilon\mathcal{T}/\varepsilon$, an execution $\bar{\mathcal{E}}$ exists such that for any $u \in \mathcal{N}_{v^{j_{\max}}}$ the relation

$$L^{\bar{\mathcal{E}}}_{v_k}(t) - L^{\bar{\mathcal{E}}}_{u}(t) = L^{\mathcal{E}}_{v_k}(t') - L^{\mathcal{E}}_{u}(t) = L^{\mathcal{E}}_{v_k}(t') - L^{\mathcal{E}}_{v_k}(t) + L^{\mathcal{E}}_{v_k}(t) - L^{\mathcal{E}}_{u}(t)$$

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

Our last theorem is proved similarly to Theorem 7.7. Using Lemma 7.11 repeatedly, we accumulate skew on paths of exponentially decreasing length.

THEOREM 7.12. *No clock synchronization algorithm can achieve a better bound on the local skew than*

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

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

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

PROOF. If $D \leq (1/\varepsilon)^c$ for any constant c, the claimed bound reduces to $\Omega(\alpha\mathcal{T})$. Such a clock skew can easily be enforced between two neighbors as shown in the proof of Theorem 7.7.

Define $\varepsilon' := 1/\lceil 1/\varepsilon\rceil > \varepsilon/2$, that is, $1/\varepsilon'$ is an integer. Throughout this proof, we will use the notation of Lemma 7.11, however, with ε replaced by ε'. Set

$b := \lceil 12 \log(8/\varepsilon') \rceil / \varepsilon' = X/\varepsilon'$. As noted above, we may assume that D is sufficiently large such that $\log_{1/\varepsilon'} \log_{1/\varepsilon'} D$ is defined and we have that

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

Therefore, when setting $D_0 := (1/\varepsilon')^{1+\lceil \log_{1/\varepsilon'} \log_{1/\varepsilon'} D \rceil}$ we have that

$$D_0 \leq \left(\frac{1}{\varepsilon'}\right)^{\lfloor \log_b D \rfloor / 2} \leq \sqrt{D},$$

implying that

$$\ell := \left\lfloor \log_b \left(\frac{D}{D_0}\right) \right\rfloor \geq \frac{\lfloor \log_b D \rfloor}{2}.$$

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

$$L_{v_i}^{\mathcal{E}_i}(t_i) - L_{w_i}^{\mathcal{E}_i}(t_i) \geq i\zeta\alpha\mathcal{T} d(v_i, w_i). \tag{41}$$

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

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

Now assume that Inequality (41) holds for some $i \in \{0, \ldots, \ell - 1\}$. Because $b^{\ell - i} D_0$ is an integer multiple of X/ε^n for $n = 2 + \lceil \log_{1/\varepsilon'} \log_{1/\varepsilon'} D \rceil$, the nodes v_i, w_i and the execution $\mathcal{E}_i$ ending at time $t_i \geq \mathcal{T}/2$ meet the requirements of Lemma 7.11. Hence, either we immediately get some execution and two neighbors exhibiting a skew of $\alpha\mathcal{T} \log_{1/\varepsilon'} D$ at some time, or for some $m \in \mathbb{N}$ a $\varphi_{\varepsilon'}$-framed execution $\mathcal{E}_{i+m}$, two nodes $v, w \in V$ at distance $d(v, w) = (\varepsilon')^m d(v_i, w_i)/X$ and a time $t_{i+m} \geq t_i \geq \mathcal{T}/2$ exist when

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

Thus, there must also be two nodes $v_{i+m}, w_{i+m} \in V$ at distance $d(v, w)/X^{m-1} = d(v_i, w_i)/b^m = b^{l-i-m} D_0$ satisfying Inequality (41) for $i + m$, that is, the induction step succeeds.

We conclude that either an execution exists in which a skew of $\alpha\mathcal{T} \log_{1/\varepsilon} D$ occurs, or Inequality (41) holds for an $i \geq \ell \in \Omega(\log_{1/\varepsilon} D)$ in some execution. In the latter case, however, the same inequality is also true for a pair of neighboring nodes, implying that there is a clock skew of at least $\ell\zeta\alpha\mathcal{T} \in \Omega(\alpha\mathcal{T} \log_{1/\varepsilon} D)$ between two neighbors. Finally, for $\varepsilon \to 0$ we have $1/\varepsilon' - 1/\varepsilon \to 0$ and $\zeta \to 1$, and for $D \to \infty$ we get $((\log_{1/\varepsilon} D) - \ell)/\ell \to 1$. Therefore, for any $\delta > 0$, appropriate choices of ε and D imply a local skew of at least $(1 - \delta)\alpha\mathcal{T} \log_{1/\varepsilon} D$. □

In other words, even if an algorithm makes use of arbitrary large logical clock rates, it cannot beat a local skew of $\alpha\mathcal{T} \log_{1/\varepsilon} D$ by a constant factor (independent of

ε and D). Moreover, we infer that $\mathcal{A}^{\text{opt}}$ achieves an asymptotically optimal gradient property even if clock rates are unbounded.

COROLLARY 7.13. *If $\beta - \alpha \in \Omega(1)$, the best possible worst-case bound on the clock skew between nodes at distance d is $\Theta(\alpha \mathcal{T} d(1 + \log_{1/\varepsilon}(D/d)))$.*

8. *Different Models*

In this section, we analyze the details of the model assumptions and point out to what extend the results carry over to other prominent models of clock synchronization.

8.1. ESTIMATES OF $\mathcal{T}$ AND ε. Our model assumes that upper bounds $\hat{\mathcal{T}} \in \mathcal{O}(\mathcal{T})$ and $\hat{\varepsilon} < 1$ on $\mathcal{T}$ and ε are known. These assumptions can be justified as follows.

Assuming that $\mathcal{T}$ is completely unknown to the algorithm is no restriction. In this case, nodes acknowledge every message, and perpetually measure the corresponding round trip times by means of their hardware clocks. Multiplying the determined values by $1/(1 - \hat{\varepsilon})$ then yields an estimate of the round trip times that is in $\mathcal{O}(\mathcal{T})$ and which upper bounds the delays of the messages. Nodes keep track of the largest estimate they either measured themselves or received in a message. If a larger (estimated) round trip time is detected, it is flooded through the system and κ (and possibly H_0) is adjusted accordingly. Note that it is not a problem if the nodes underestimate $\mathcal{T}$ because, until the time when larger delays actually occur, the skew bounds hold with respect to the smaller delays and thus the smaller κ. In order to keep the number of messages low, one could initially use an estimate of $\Theta(1/f)$ and double it in every step, reducing the number of updates to at most $\mathcal{O}(\log(\mathcal{T}/f))$.[14]

As far as the assumption that ε is bounded by $\hat{\varepsilon} < 1$ is concerned, we point out that an ε arbitrarily close to one means that we do not have clocks in the truest sense of the word.[15] In particular, $\varepsilon = 1$ would allow the hardware clocks to stand still, a case in which nodes are not able to react to clock skews at all. In such a setting it would be reasonable to drop the constraint that the progress rates must be bounded at all times in favor of sudden clock jumps (i.e., $\mu = \infty$ and therefore still "large enough" to guarantee Inequality (6)). However, we do not cover this rather extreme scenario in our discussion.

8.2. MINIMUM CLOCK RATE α. As pointed out in Section 7, the lower bound on the local skew does not depend on clocks running always at least at a rate of α. Requiring an average rate of α merely for intervals of a certain minimum length does not change the asymptotic bounds, unless clocks are allowed to stand still (or even run backwards) for almost $D\mathcal{T}$ time. Moreover, the lower bound on the global skew of roughly $D\mathcal{T}$ trivially implies a bound of $\mathcal{T}$ on the local skew, that is, even if we do not insist on a certain minimum progress rate, any algorithm guaranteeing Condition (2) will suffer a clock skew of $\mathcal{T}$ between neighboring nodes in the worst case. Considering that typically $\mathcal{T} \log_{1/\varepsilon} D \in \mathcal{O}(\mathcal{T})$ and $\mathcal{A}^{\text{opt}}$ attains this bound on the local skew, there is little point in choosing $\alpha < 1 - \mathcal{O}(\varepsilon)$.

[14] Recall that f denotes the hardware clock frequency.

[15] A cheap quartz oscillator exhibits a relative drift of less than 10^{-4} and even the clock drift of a ring oscillator under varying temperatures and support voltages is not considerably larger than 0.2.

8.3. LOWER BOUNDED DELAYS. Throughout this article, we assumed that delays are always in the range $[0, \mathcal{T}]$. In many distributed systems, it is more adequate to assume that all delays lie in a range $[\mathcal{T}_1, \mathcal{T}_2]$, where $\mathcal{T}_2 - \mathcal{T}_1 \ll \mathcal{T}_1$. It is evident from the proofs of the lower bounds that they still hold with $\mathcal{T}$ replaced by $\mathcal{T}_2 - \mathcal{T}_1$ in this situation. Similarly, the algorithm can be applied efficiently if we add $\mathcal{T}_1$ to all values received. However, triggering messages when $L_v^{\max}$ reaches a multiple of H_0 in order to bound the (amortized) message frequency does not work any more. This can either be solved by simply sending messages every H_0 local time (as discussed in Section 6), or by enforcing one logical clock to be the fastest by slowing down all other clocks slightly and performing "external" synchronization where the distinguished node serves as the reference (cf. Section 8.5).

Another effect, however, might be of more concern: The skew bounds will degrade because the algorithm needs more time to react to clock skews. The global skew will increase by $\mathcal{O}(\varepsilon D\mathcal{T}_1)$, which is asymptotically optimal due to the fact that distant nodes may not receive messages from each other for $\Omega(D\mathcal{T}_2)$ time because of the slow information transport. Regarding the local skew, Lemma 5.7 has to be adapted in that now $\Xi(\cdot)$ reduces merely at an amortized rate of $\Omega(\min\{\mu, \kappa/\mathcal{T}_2\})$, implying that $\mu \geq (\mathcal{T}_2 - \mathcal{T}_1)/\mathcal{T}_2$ only improves the base σ of the logarithm at the expense of increasing κ linearly with σ. However, if $\mathcal{T}_2 \leq \sqrt{\varepsilon}(\mathcal{T}_2 - \mathcal{T}_1)$, for instance, choosing $\mu \in \Theta(\sqrt{\varepsilon})$ and κ appropriately will still result in an asymptotically optimal local skew of $\mathcal{O}(\kappa \log_{1/\varepsilon} D)$.

8.4. DISCRETE CLOCK SYNCHRONIZATION. Apparently, real-world systems are not able to resolve time arbitrarily precisely. Instead, hardware clocks generate "clock pulses" or "ticks" at a slightly varying frequency f. Thus, (local) time becomes discrete in the sense that computations can distinguish only between different ticks, and receiving and sending messages is only possible at specific times. The impact of the limited granularity of the hardware clock on $\mathcal{A}^{\text{opt}}$ has been studied in Lenzen et al. [2009a]. Not surprisingly, it turns out that $\mathcal{T}$ is basically replaced by $\max\{1/f, \mathcal{T}\}$, that is, typically the effects are negligible because $1/f < \mathcal{T}$.

8.5. EXTERNAL CLOCK SYNCHRONIZATION. There is a lot of work on *external* clock synchronization algorithms [Moses and Bloom 1994; Ostrovsky and Patt-Shamir 1999; Patt-Shamir and Rajsbaum 1994], where a source of real time is available and the objective is to synchronize all clocks to this source. Thus, there is a single node v_0 for which logical clock time, hardware clock time, and real time are identical. In order to allow for the fact that a distant node v may not be informed about the real time more accurately than $\mathcal{T}d(v, v_0)$, Condition (1) has to be changed to $t - d(v, v_0)\mathcal{T} - \tau \leq L_v(t) \leq t$, where $\tau \in \mathbb{R}^+$ addresses the issue that nodes should only send a finite number of messages in constant time.

$\mathcal{A}^{\text{opt}}$ can easily be adjusted to handle this modified constraint. The node v_0 has to propagate its clock value through the system periodically, at least every $\Theta(\tau/\hat{\varepsilon})$ time. The nodes behave the same as in $\mathcal{A}^{\text{opt}}$, except that they increase $L_v^{\max}$ at the rate $h_v/(1 + \hat{\varepsilon})$ and do the same with L_v whenever $L_v = L_v^{\max}$. This technique ensures that the logical clock rates are upper bounded by 1 whenever the largest clock value in the system is attained, implying that $L_v(t) \leq t$ at all times. On the other hand, nodes still raise their clocks quickly when large estimates are received. Apparently, the global skew is bounded by $\mathcal{T}D + \mathcal{O}(\tau)$, and the worst-case clock skew between some node v and v_0 is linearly bounded in the distance between the two nodes. The main difference is that the minimum progress rate is now only bounded by

$(1 - \mathcal{O}(\hat{\varepsilon}))$, which can easily be accounted for when determining μ and κ. Thus, the algorithm still guarantees roughly the same skew bounds without significantly increasing μ or κ. The amortized message frequency is $\Theta(\tau/\hat{\varepsilon} + 1/H_0)$.

8.6. HARDWARE CLOCK ENVELOPE CONDITION. A similar technique is applicable if Condition (1) is replaced by

$$\forall v \in V \ \forall t \ : \ \min_{w \in V}\{H_w(t)\} \leq L_v(t) \leq \max_{w \in V}\{H_w(t)\},$$

that is, the time envelope condition is sharpened to the requirement that all logical clock values must always be at least the smallest and at most the largest hardware clock value in the system. In this case, a node $v \in V$ must reduce its clock rate when $L_v(t) > H_v(t)$, while still responding to clock skews. This is accomplished by increasing $L_v^{\max}$ at the reduced rate $(1 - \hat{\varepsilon})h_v/(1 + \hat{\varepsilon})$ whenever it exceeds H_v and again refusing to increase L_v beyond $L_v^{\max}$. Thus, nodes will never have a larger logical clock rate than $1 - \varepsilon$ if $L_v(t) = L_v^{\max}(t) = \max_{w \in V}\{H_w(t)\} > H_v(t)$. As they also increase their logical clocks at the normal rate when $L_v(t) = H_v(t)$, the requested constraint is satisfied. Clock rates change merely by a factor of $1 - \mathcal{O}(\hat{\varepsilon})$, therefore the bounds on κ and μ, and the impact of H_0 remain basically the same.

9. *Conclusion*

We studied the well-known clock synchronization problem for arbitrary underlying topologies. As our main result, we presented the synchronization algorithm $\mathcal{A}^{\text{opt}}$ and proved matching upper and lower bounds on the worst-case clock skew between neighboring nodes and between arbitrary nodes in the distributed system. Remarkably, these results hold in a very general model where clock drifts and delays may vary arbitrarily within unknown bounds. Surprisingly, strong bounds can also be achieved if the logical clock rates must always be in the range $[1 - \mathcal{O}(\varepsilon), 1 + \mathcal{O}(\varepsilon)]$ and we require that $\Omega(\mathcal{T}/\varepsilon)$ time elapses between message transmissions at each node, where ε and $\mathcal{T}$ denote the maximum clock drift rate and the delay uncertainty, respectively. Moreover, the bound on the global skew is essentially optimal and, if upper bounds on both the clock drift and the delay uncertainty are known fairly accurately, the asymptotic approximation ratio of the proposed algorithm with regard to the local skew is 2. The algorithm $\mathcal{A}^{\text{opt}}$ can further be adjusted in order to be applicable to various other models and constraints, which demonstrates the generality and flexibility of our techniques.

Our results may be relevant for practical applications for the following reasons. We showed that although the local skew must grow logarithmically in the diameter D of the system network, the base of the logarithm can be bounded by $\Theta(1/\varepsilon)$. Thus, if we have that $D \in \mathcal{O}(1/\varepsilon^c)$ for a constant c, a worst-case clock skew of $\mathcal{O}(\mathcal{T})$ between neighboring nodes can be guaranteed. Since typical clock drifts are in the order of 10^{-5} and the diameters of most current networks are not larger than roughly 20 to 30, the clock skew between neighboring nodes can be bounded by $\mathcal{O}(\mathcal{T})$ in most real-world systems. Furthermore, it is clearly desirable that the clocks run smoothly and progress at reasonable rates at all times. It is possible to achieve such a strong bound even if we impose tight restrictions on the rates of the clocks, that is, the clocks are never allowed to change abruptly in order to correct the observed clock skews. In particular, rates of $1 \pm \mathcal{O}(\sqrt{\varepsilon})$ are sufficient to guarantee an optimal global skew and a roughly 4-competitive local skew. Given that $\mathcal{A}^{\text{opt}}$

is further computationally efficient and that it requires a low message frequency, we believe that practical protocols can be designed that guarantee not only small global skews at low communication costs, but also small local skews and smoothly progressing logical clocks. We hope that some of the ideas introduced in this work might lead to further advances in clock synchronization protocols.

ACKNOWLEDGMENT. We would like to thank Fabian Kuhn and the anonymous reviewers for many valuable comments.

REFERENCES

AWERBUCH, B. 1985. Complexity of network synchronization. *J. ACM 32,* 4, 804–823.

BIAZ, S., AND LUNDELIUS WELCH, J. 2001. Closed form bounds for clock synchronization under simple uncertainty assumptions. *Inf. Proc. Lett. 80,* 3, 151–157.

ELSON, J., GIROD, L., AND ESTRIN, D. 2002. Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Ope. Syst. Rev. 36*, 147–163.

FAN, R., AND LYNCH, N. 2004. Gradient clock synchronization. In *Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC)*. ACM, New York, 320–327.

FÜGGER, M., SCHMID, U., FUCHS, G., AND KEMPF, G. 2006. Fault-tolerant distributed clock generation in VLSI systems-on-Chip. In *Proceedings of the 6th European Dependable Computing Conference (EDCC-6)*. 87–96.

GANERIWAL, S., KUMAR, R., AND SRIVASTAVA, M. B. 2003. Timing-sync protocol for sensor networks. In *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys)*. ACM, New York, 138–149.

KORTE, B., RAUTENBACH, D., AND VYGEN, J. 2007. BonnTools: Mathematical innovation for layout and timing closure of systems on a chip. *Proc. IEEE 95,* 3, 555–572.

KUHN, F., LOCHER, T., AND OSHMAN, R. 2009. Gradient clock synchronization in dynamic networks. In *Proceedings of the 21st ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. ACM, New York, 270–279.

LENZEN, C., LOCHER, T., AND WATTENHOFER, R. 2008. Clock synchronization with bounded global and local skew. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society Press, Los Alamitos, CA, 500–510.

LENZEN, C., LOCHER, T., AND WATTENHOFER, R. 2009a. Tight bounds for clock synchronization. In *Proceedings of the 28th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. ACM, New York, 46–55.

LENZEN, C., SOMMER, P., AND WATTENHOFER, R. 2009b. Optimal clock synchronization in networks. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*. ACM, New York.

LOCHER, T. 2009. Foundations of aggregation and synchronization in distributed systems. Ph.D. dissertation, ETH Zurich.

LOCHER, T., AND WATTENHOFER, R. 2006. Oblivious gradient clock synchronization. In *Proceedings of the 20th International Symposium on Distributed Computing (DISC)*. 520–533.

LUNDELIUS WELCH, J., AND LYNCH, N. 1984. An upper and lower bound for clock synchronization. *Inf. Cont. 62,* 2/3, 190–204.

MARÓTI, M., KUSY, B., SIMON, G., AND LÉDECZI, Á. 2004. The flooding time synchronization protocol. In *Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys)*. ACM, New York, 39–49.

MEIER, L., AND THIELE, L. 2005. Brief announcement: Gradient clock synchronization in sensor networks. In *Proceedings of the 24th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. ACM, New York, 238.

MILLS, D. 1991. Internet time synchronization: The network time protocol. *IEEE Trans. Commu. 39*, 1482–1493.

MOSES, Y., AND BLOOM, B. 1994. Knowledge, timed precedence and clocks. In *Proceedings of the 13th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. ACM, New York, 294–303.

OSTROVSKY, R., AND PATT-SHAMIR, B. 1999. Optimal and efficient clock synchronization under drifting clocks. In *Proceedings of the 18th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. ACM, New York, 400–414.

PALCHAUDHURI, S., SAHA, A. K., AND JOHNSON, D. B. 2004. Adaptive clock synchronization in sensor networks. In *Proceedings of the 3rd ACM/IEEE International Symposium on Information Processing in Sensor Networks (IPSN)*. ACM, New York, 340–348.

PATT-SHAMIR, B., AND RAJSBAUM, S. 1994. A theory of clock synchronization. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, New York, 810–819.

SOMMER, P., AND WATTENHOFER, R. 2009. Gradient clock synchronization in wireless sensor networks. In *Proceedings of the 8th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. ACM, New York, 37–48.

SRIKANTH, T. K., AND TOUEG, S. 1987. Optimal clock synchronization. *J. ACM 34,* 3, 626–645.

RECEIVED JULY 2008; REVISED SEPTEMBER 2009; ACCEPTED SEPTEMBER 2009