Computer Science and Artificial Intelligence Laboratory
Technical Report

MIT-CSAIL-TR-2009-022 May 29, 2009

Gradient Clock Synchronization in Dynamic Networks

Fabian Kuhn, Thomas Locher, and Rotem Oshman

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Gradient Clock Synchronization in Dynamic Networks
Technical Report

Fabian Kuhn[*] Thomas Locher[†] Rotem Oshman[*]
fkuhn@csail.mit.edu lochert@tik.ee.ethz.ch rotem@csail.mit.edu

[*] Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA 02139, USA
[†] Computer Engineering and Networks Laboratory, ETH Zurich, 8092 Zurich, Switzerland

May 26, 2009

1 Introduction

Establishing coordination between participants is at the core of many algorithmic challenges in distributed computing. A fundamental coordination task, and a basic prerequisite for many applications, is achieving a common notion of time. Typically every node in the network has access to a local hardware clock, but the hardware clocks of different nodes run at slightly different rates, and the rates can change over time. In addition, although a bound on the message delays in the network may be known, specific message delays are unpredictable. As a consequence it is generally not possible for any node in the network to get an accurate estimate of the clock values of neighboring nodes.

Operating under these uncertainties, a distributed clock synchronization algorithm computes logical clocks at every node, with the goal of synchronizing these clocks as tightly as possible. Traditionally, distributed clock synchronization algorithms tried to minimize the maximal difference between any two logical clocks in the network. We call this quantity the *global skew* of a clock synchronization algorithm. It is well-known that no algorithm can guarantee a global skew better than $\Omega(D)$, where D is the diameter of the network [3].

In many cases it is more important to tightly synchronize the logical clocks of nearby nodes in the network than it is to minimize the global skew. For example, to run a time division multiple access (TDMA) protocol for coordinating access to the shared communication medium in a wireless network, one only needs to synchronize the clocks of nodes that interfere with each other when transmitting. The problem of achieving synchronization that depends on the distance between the two nodes is called *gradient clock synchronization*. It was introduced in a seminal paper by Fan and Lynch, where it is also shown that surprisingly, a clock skew of $\Omega(\log D / \log \log D)$ cannot be prevented even between immediate neighbors in the network [8]. The maximal difference between the two logical clocks of adjacent nodes in the network is called the *local skew* of a clock synchronization algorithm; for static networks, Lenzen et. al. have recently proven an asymptotically tight bound of $\Theta(\log D)$ for the best possible local skew an algorithm can achieve [11, 12]. For other related work on clock synchronization, see Section 2.

Most existing work on clock synchronization considers static networks. However, many modern networks are inherently dynamic. Typically formed by autonomous agents without central control, nodes can join and leave the network in an arbitrary pattern. In ad hoc networks where often the devices are even assumed to be mobile, the resulting network topology can be highly dynamic even if the set of participating nodes remains stable. Coordination in dynamic networks is challenging, and due to the increasing significance of such networks, it is also particularly important.

In this paper we study the gradient clock synchronization problem in dynamic networks. Because the distance between nodes in the network can change over time, the problem becomes significantly harder in a dynamic setting. Consequently, unlike the static case, the requirements we make on the skew between the logical clocks of different nodes can also change over time. Every new edge that is formed induces a new and stronger constraint on the skew between its endpoints; the algorithm must adapt by reducing the skew on the edge until the new constraint is satisfied.

Hence, we distinguish between two requirements: a *stable local skew* bound applies, conceptually, to edges that exist for a long time. This is analogous to the local skew guaranteed by gradient clock synchronization algorithms for static networks. In practice, we impose a weaker *dynamic local skew* bound on all the edges, including new ones. The dynamic local skew bound is a function of how long the edge has existed: the bound starts out weak and grows stronger with time, until in the limit it converges to the stable local skew bound.

The following intuitive example shows that in general, the clock skew on a new edge cannot be reduced too quickly without violating the stable local skew bound on edges that were formed a long time before. Let u and v be two nodes at distance k from each other. As no algorithm can prevent a skew of $\Omega(k)$ between nodes at distance k, a newly formed edge between nodes u and v can carry $\Omega(k)$ local skew. To reduce the skew on the new edge, whichever node is behind must increase its logical clock by a large amount. However, a sudden increase in u or v's clocks will create a large skew along the edges of the old path that connects them. Specifically, if the algorithm guarantees a stable local skew of $\mathcal{S}$, neither u nor v can instantaneously increase their logical clocks to more than $\mathcal{S}$ ahead of their next neighbor along the old path. In turn, when this neighbor realizes it must increase its clock, it cannot increase it to more than $\mathcal{S}$ ahead of *its* next neighbor, and so on. It takes $\Omega(k/\mathcal{S})$ time until the skew can be reduced, as information about the new edge can take time to propagate through the path.

Somewhat surprisingly, the example above is not the worst one possible: adjusting the local skew on a newly formed edge can require even more than $\Omega(k/\mathcal{S})$ time, where k is the previous distance between the endpoints of the new edge. We show that (almost) independent of the initial skew on a new edge, the time required to reduce the initial skew to $\mathcal{S}$ is at least $\Omega(n/\mathcal{S})$ where n is the number of nodes in the system. This is shown in Section 4.

In Section 5 we show that this lower bound is asymptotically tight for moderately small values of $\mathcal{S}$ by extending a simple gradient clock synchronization algorithm described in [13] to the dynamic case. In a static setting, the algorithm of [13] guarantees a local skew of $O(\sqrt{\rho D})$ where ρ is the maximum hardware clock drift. When modeling a dynamic network, we assume that the set of nodes remains fixed, but edges can appear and disappear in a completely arbitrary pattern. If a weak connectivity requirement is satisfied, the algorithm guarantees a global skew of $O(n)$ at all times. Further, for a parameter $\mathcal{S} \geq \sqrt{\rho n}$ and a sufficiently large constant λ, the algorithm guarantees a local skew of at most $\mathcal{S}$ on all edges that are present for at least $\lambda \cdot n/\mathcal{S}$ time. It will be interesting to see whether techniques used in the recent strong static gradient clock synchronization algorithms in [11, 12] can be adapted to the dynamic setting, in order to obtain similar results for smaller values of $\mathcal{S}$. A first step in this direction was recently made in [9], where we extended the algorithm from [12] to handle links with different bounds on message delay [6].

2 Related Work

Being a fundamental problem, it is not surprising that there is a rich literature on clock synchronization algorithms and lower bounds Until recently, the work on clock synchronization focused on global synchronization, i.e., on minimizing the maximal clock difference between any two nodes in the system. Essentially all lower bounds on distributed clock synchronization use the *shifting* technique introduced in [14], which exploits uncertainty resulting from unknown message delays, the *scaling* technique from [5], which uses uncertainty that arises as a consequence of different clock rates, or a combination of the two techniques. Using the shifting technique, it is shown in [3] that even if clocks experience no drift, a clock skew of $D/2$ can not be avoided in a network of diameter D. In light of this result, the algorithm described in [18] which guarantees a global skew of $O(D)$ is asymptotically optimal.

A number of related algorithms and lower bounds for varying models and with different properties have been described (see e.g. [1, 2, 7, 16, 17]). The algorithms described in these papers do not guarantees a skew between neighboring nodes that is better than $O(D)$. The gradient clock synchronization problem was introduced in [8], where it is shown that on a path of length D, no clock synchronization algorithm can avoid having a skew of $\Omega(\log D/\log\log D)$ between adjacent nodes. This lower bound has recently been improved to $\Omega(\log D)$ in [12]. The first algorithm to guarantee a non-trivial local skew was described by Locher and Wattenhofer in [13]. The algorithm in [13] guarantees a local skew of $O(\sqrt{\rho D})$ between any two neighbors in a network of diameter D, where ρ denotes the maximal hardware clock drift. The algorithm of [13] forms the basis for the dynamic gradient clock synchronization algorithm described in this paper. For static networks, the upper bound was recently improved to an asymptotically optimal bound of $O(\log D)$ by Lenzen et. al. [11, 12].

Most closely related to the dynamic clock synchronization problem considered in this work are algorithms that cope with faulty nodes (e.g. [4, 5, 10, 15]). While this line of work goes far beyond studying crash failures and describes algorithms that even cope with Byzantine faults, a topic that is out of the scope of the present paper, none of these papers consider a truly dynamic setting. In particular, the results rely on the fact that a considerable part of the network remains non-faulty and stable. Moreover, all the described algorithms and lower bounds focus solely on global synchronization. To the best of our knowledge, the present paper is the first to look at gradient clock synchronization in dynamic networks.

3 Preliminaries

3.1 Notation

Given an undirected static graph $G = (V, E)$, we denote by $\mathcal{P}$ the set of all (undirected) paths in G. For convenience in notation we regard each path $P \in \mathcal{P}$ as a set of edges $P \subseteq E$. We use $\mathcal{P}(u, v)$ to denote all paths between two nodes $u, v \in V$. The distance between two nodes u and v is defined by

$$\mathrm{dist}(u, v) := \min_{P \in \mathcal{P}(u,v)} |P|.$$

The definitions above are used only in the context of a static graph. (We use static graphs in the proof of the lower bound in Section 4). In this work we are often concerned with dynamic graphs, which do not have a static set of edges. We use $V^{(2)} := \{\{u, v\} \mid u, v \in V\}$ to denote the set of all *potential* edges over a static set V of nodes.

3.2 Network Model

We model a dynamic network over a static set V of nodes using Timed I/O Automata (TIOA). Each node in the network is modelled as a TIOA, and the environment is also modelled as a TIOA. The dynamic behavior of the network is modelled using events of the form $\mathsf{add}(\{u,v\})$ and $\mathsf{remove}(\{u,v\})$ for $u, v \in V$, which correspond to the formation and failure (respectively) of a link between u and v. It is assumed that no edge is both added and removed at the same time.

The history of link formations and failures in a particular execution α, together with an initial set of edges E_0^α, induces a *dynamic graph* $G = (V, E^\alpha)$, where $E^\alpha : \mathbb{R}^+ \to V^{(2)}$ is a function that maps a time $t \geq 0$ to the set of edges (links) that exist in α at time t. We define $E^\alpha(t)$ to be the set of edges that are added no later than time t, and not removed between the last time they are added and time t (inclusive). This includes edges that appear in E_0^α and are not removed by time t. We say that an edge e *exists throughout the interval* $[t_1, t_2]$ in α if $e \in E^\alpha(t_1)$ and e is not removed at any time during the interval $[t_1, t_2]$.

A *static execution* is one in which no edges are added or removed. Formally, α is a static execution if for all $t_1, t_2 \in \mathbb{R}^+$ we have $E^\alpha(t_1) = E^\alpha(t_2)$.

We consider a very general model, in which edges can be inserted or removed arbitrarily, subject only to the following connectivity constraint.

Definition 3.1 (T-interval connectivity)**.** *We say that a dynamic graph $G = (V, E^\alpha)$ is T-*interval connected *if for all $t \geq 0$, the static subgraph $G_{[t,t+T]} = (V, E^\alpha|_{[t,t+T]})$ is connected, where $E^\alpha|_{[t,t+T]}$ is the set of all edges that exist throughout the interval $[t, t+T]$.*

In the sequel we omit the superscript α when it is clear from the context.

We assume that nodes do not necessarily find out immediately about edge insertions and removals[1]. Instead, we assume that there is a parameter $\mathcal{D}$, such that if an edge appears or disappears at time t in an execution, and the change is not reversed by time $t + \mathcal{D}$, the endpoints of the edge find out no later than time $t + \mathcal{D}$. Transient link formations or failures, which do not persist for $\mathcal{D}$ time, may or may not be detected by the nodes affected. We model the discovery by node u of a link formation or failure $X \in \{\mathsf{add}(\{u,v\}), \mathsf{remove}(\{u,v\}) \mid v \in V\}$ by an event $\mathsf{discover}(X)$ that occurs at node u. (A $\mathsf{discover}(X)$ event is always preceded by event X itself.)

We also assume reliable FIFO message delivery, with message delays bounded by $\mathcal{T}$. This is modelled using events of the form $\mathsf{send}(u,v,m)$ and $\mathsf{receive}(u,v,m)$ that occur at node u. If node u sends a message to node v at time t, the environment guarantees the following. If edge $\{u,v\}$ exists throughout the interval $[t, t+\mathcal{T}]$, then node v is guaranteed to receive the message no later than time $t+\mathcal{T}$. If edge $\{u,v\}$ exists at time t but is removed at some point in the interval $[t, t+\mathcal{T}]$, there are two possible outcomes: either the message is delivered before the edge is removed, or the message is not delivered and node u discovers the edge removal no later than time $t + \mathcal{D}$. Finally, if edge $\{u,v\}$ does not exist at time t, the message is not delivered, and node u discovers that the edge does not exist no later than time $t + \mathcal{D}$. These definitions correspond to an abstract version of MAC layer acknowledgements.

[1]Otherwise reference-broadcast-style synchronization would be possible using these events [6].

In the sequel we assume that $\mathcal{D} > \mathcal{T}$, that is, nodes do not necessarily find out about changes to the network within $\mathcal{T}$ time units. This is a reasonable assumption because even if nodes transmit very frequently, as much as $\mathcal{T}$ time may pass without any message being received on a link, leaving the link formation or failure undiscovered.

3.3 The Clock Synchronization Problem

In the clock synchronization problem, each node $u \in V$ has access to a continuous *hardware clock* $H_u(t)$, which may progress at a different rate than real time. The hardware clocks suffer from *bounded drift* ρ: although they progress at a variable rate, their rate is always between $1-\rho$ and $1+\rho$ the rate of real time, so that for any node u and times $t_1 < t_2$ we have

$$(1-\rho)(t_2 - t_2) \le H_u(t_2) - H_u(t_2) \le (1+\rho)(t_2 - t_1).$$

For simplicity we assume that at the beginning of any execution the hardware clock values are all 0.

The goal of a dynamic clock synchronization algorithm (DCSA) is to output a *logical clock* $L_u(t)$ such that the logical clocks of different nodes are close to each other. In particular we consider two requirements. A *global skew constraint* bounds the difference between the logical clocks of any two nodes in the network at all times in the execution. A *dynamic local skew constraint* requires that if an edge exists for sufficiently long, the skew between the two endpoints of the edge should not be too large. These requirements are formally defined as follows.

Definition 3.2 (Global skew)**.** *A DCSA guarantees a* global skew of $\bar{\mathcal{G}}(n)$ *if in any execution of the algorithm in a network of n nodes, for any two nodes u, v and time $t \ge 0$ we have*

$$L_u(t) - L_v(t) \le \bar{\mathcal{G}}(n).$$

Definition 3.3 (Skew function)**.** *A function $s : \mathbb{N} \times \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$ (where $\mathbb{R}^+$ are the nonnegative reals) is said to be a* skew function *if the following conditions hold.*

1. *The function $s(n, I, t)$ is non-decreasing in I and non-increasing in t; and*
2. *For all $n \in \mathbb{N}$, $I \in \mathbb{R}^+$, the limit $\lim_{t\to\infty} s(n, I, t)$ is defined and finite; and*
3. *For all $I_1, I_2 \in \mathbb{R}^+$ we have*

$$\lim_{t\to\infty} s(n, I_1, t) = \lim_{t\to\infty} s(n, I_2, t).$$

Definition 3.4 (Dynamic local skew)**.** *A DCSA guarantees a* dynamic local skew *of $s : \mathbb{N} \times \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$, where s is a skew function, if in every execution of the algorithm in a network over n nodes, for any edge $e = \{u, v\}$ and times $t_1 \le t_2$ such that e exists throughout the interval $[t_1, t_2]$ in the execution, we have*

$$|L_u(t_2) - L_v(t_2)| \le s(n, |L_u(t_1) - L_v(t_1)|, t_2 - t_1).$$

Definition 3.5 (Stabilizing DCSA)**.** *A DCSA $\mathcal{A}$ is said to be* stabilizing *if there is a skew function s such that $\mathcal{A}$ guarantees a dynamic local skew of s. In this case we say that $\mathcal{A}$ guarantees a* stable local skew *of $\bar{s}(n) = \lim_{t\to\infty} s(n, I, t)$ for some $I \in \mathbb{R}^+$.*

Finally, logical clocks have to be strictly increasing and are thus not allowed to temporarily stop. In particular, we require the rate of each logical clock to be at least half the rate of real time; that is, for any node u and times $t_1 \le t_2$ we require

$$L_u(t_2) - L_u(t_1) \ge \frac{1}{2}(t_2 - t_1).$$

4 Lower Bound

We begin our analysis of dynamic clock synchronization algorithms with a lower bound on the time needed to adjust the local skew an a newly formed edge. The following theorem shows that for every sufficiently large initial skew I (a large enough constant times the stable local skew $\bar{s}(n)$ suffices), the time needed to reduce the skew by a constant factor is $\Omega(n/\bar{s}(n))$. Thus, there is an inherent tradeoff between the stable skew guaranteed by the algorithm and the time the algorithm requires to reduce the skew on new edges.

4.1 Overview

The formal statement of the lower bound is as follows.

Theorem 4.1. *Let $\mathcal{A}$ be a stabilizing DCSA that guarantees a global skew of $\bar{\mathcal{G}}(n) = O(n)$ and a dynamic local skew of s with a stable local skew of $\bar{s}(n) = o(n)$ in networks with diameter $\Theta(n)$. Then there exist constants $\lambda, \zeta \ge 0$ such that for all sufficiently large n and I we have $s(n, I, \lambda \cdot \frac{n}{\bar{s}(n)}) \ge \zeta \cdot I$.*

We begin by establishing a lemma that lets us set up a large skew between two nodes, while maintaining large message delays on some pre-specified links.

In the standard constructions, increasing the skew between nodes is done by adjusting message delays; in the resulting execution, some links in the network will have zero message delay, and we do not have control over which links these will be. For our purposes, certain predetermined links in the network *must* have large message delays. We accomplish this by "setting aside" these links, and using only the remaining links to build up skew. The following definitions capture this notion more formally.

Definition 4.1 (Delay mask)**.** *Given a network over a set V of nodes, a* delay mask *for N is a pair $M = (E^{\mathrm{C}}, P)$, where $E^{\mathrm{C}} \subseteq V^{(2)}$ is a set of* constrained links *and $P : E^{\mathrm{C}} \to [0, \mathcal{T}]$ is a* delay pattern *assigning a message delay to every constrained link.*

Definition 4.2 (Constrained executions)**.** *An execution is said to be M-constrained until time t, for a delay mask $M = (E^{\mathrm{C}}, P)$, if the delay of messages sent on a link $e \in E^{\mathrm{C}}$ and received by time t is in the range $[\frac{1}{1+\rho}P(e), P(e)]$. We say that an execution is M-*constrained *if for all times $t \ge 0$ the execution is M-constrained until time t.*

Definition 4.3 (Flexible distance)**.** *Given a delay mask $M = (E^{\mathrm{C}}, P)$, the M-*flexible distance *between two nodes $u, v \in V$, denoted $\mathrm{dist}_M(u, v)$, is defined as the minimum number of unconstrained edges on any path between u and v.*

Lemma 4.2 (Masking Lemma)**.** *Let $N = (V, E)$ be a static network, and let $M = (E^{\mathrm{C}}, P)$ be a delay mask for N. For any time $t > \mathcal{T} \cdot \mathrm{dist}_M(u, v)(1 + 1/\rho)$, there is an M-constrained static execution in which*

$$|L_u(t) - L_v(t)| \geq \frac{1}{4}\mathcal{T}\,\mathrm{dist}_M(u, v).$$

Proof. If $\mathrm{dist}_M(u, v) = 0$ the claim is immediate. Assume therefore that $\mathrm{dist}_M(u, v) = d > 0$.

▷ **Part I: definitions and setup**

We partition the graph into layers $L_0, \ldots, L_D$ where $L_i = \{w \in V \mid \mathrm{dist}_M(u, w) = i\}$ and $D = \max_{w \in V} \mathrm{dist}_M(u, w)$. In particular, $L_0 = \{u\}$ and $v \in L_d$. We define a total order $\preceq$ on nodes by $x \preceq y$ iff $\mathrm{dist}_M(u, x) \leq \mathrm{dist}_M(u, y)$. We write $x \equiv y$ if $\mathrm{dist}_M(u, x) = \mathrm{dist}_M(u, y)$, and $x \prec y$ if $x \preceq y$ and $x \not\equiv y$.

Note the following properties of the relations defined above: for any edge $\{x, y\} \in E$,

1. If $\{x, y\} \in E^{\mathrm{C}}$ then $x \equiv y$: if $\{x, y\} \in E^{\mathrm{C}}$, then any path from u to x can be extended to a path from u to y that has the same number of unconstrained edges, and vice-versa. It follows that $\mathrm{dist}_M(u, x) = \mathrm{dist}_M(u, y)$.

2. If $x \prec y$ then $\mathrm{dist}_M(u, x) = \mathrm{dist}_M(u, y) - 1$.

We define two executions α and β. In α, all hardware clocks progress at the rate of real time, and message delays on each edge e are defined as follows:

- If $e \in E^{\mathrm{C}}$ then messages on e are delayed by $P(e)$;
- If $e = \{x, y\} \in E \setminus E^{\mathrm{C}}$ where $x \prec y$, then messages from x to y are delayed by $\mathcal{T}$ and messages from y to x are delayed by 0.

In β, we slowly increase the skew of the hardware clocks of nodes at different layers, while keeping the difference small enough that it can be disguised by altering message delays. We begin by keeping u's hardware clock rate at 1 and letting nodes in layers $L_1, \ldots, L_D$ run at a rate of $1 + \rho$, until a skew of $\mathcal{T}$ is built up between the hardware clock of u and any node in L_1. Then we keep u and all L_1-nodes at a rate of 1 while nodes in layers $L_2, \ldots, L_D$ run at a rate of $1 + \rho$ until a skew of $\mathcal{T}$ is built up between nodes in L_1 and nodes in L_2. At this point the hardware clock skew between u and any node in L_2 is $2\mathcal{T}$. We continue in this manner until we have built up a skew of $d \cdot \mathcal{T}$ between u and any node in layer L_d, including v.

More formally, β is constructed as a sequence of segments $\beta_0\beta_1 \ldots \beta_{d-1}\beta_*$, where β_* is an infinite suffix and for all $0 \leq i \leq d - 1$, $\beta_i = \left[\frac{i}{\rho}\mathcal{T}, \frac{i+1}{\rho}\mathcal{T}\right)$ is a finite segment of duration $\mathcal{T}/\rho$.

(This is the time required to to build a skew of $\mathcal{T}$ between the hardware clocks of nodes in adjacent layers when one node runs at a rate of 1 and the other at $1+\rho$.)

At any time $t \in \beta_i$, the hardware clock rate of a node $x \in L_j$ is given by

$$\frac{d}{dx} H_x^\beta = \begin{cases} 1 & \text{if } i \le j, \\ 1+\rho & \text{otherwise.} \end{cases}$$

In β_* all hardware clocks run at a rate of 1.

Message delays throughout β are adjusted so that β is indistinguishable from α to all nodes. In particular, if $t_s^\alpha, t_s^\beta, t_r^\alpha$ and t_r^β are times such that

1. At time t_s^α in α node x sends a message that node y receives at time t_r^α, and
2. $H_x^\alpha(t_s^\alpha) = H_x^\beta(t_s^\beta)$ and $H_y^\alpha(t_r^\alpha) = H_x^\beta(t_r^\beta)$,

then in β, node x will send the same message at time t_s^β and node y will receive it at time t_r^β.

From the definition of β, at any time t we have

$$H_x^\beta(t) = \begin{cases} (1+\rho)t & \text{if } t \in \beta_i \text{ where } \operatorname{dist}_M(u,x) > i, \\ t + \mathcal{T} \cdot \operatorname{dist}_M(u,x) & \text{otherwise} \end{cases}$$

That is,

$$H_x^\beta(t) = t + \min\left\{\rho t, \mathcal{T} \cdot \operatorname{dist}_M(u,x)\right\}. \tag{1}$$

In α, where all hardware clocks run at a rate of 1, $H_x^\alpha(t) = t$ for all $x \in V$.

$\triangleright$ **Part II: β is an M-constrained execution**

Next we claim that β is a legal M-constrained execution, that is, all message delays are in the range $[0, \mathcal{T}]$, and for all $e \in E^{\mathrm{C}}$, message delays on e are in the range $\left[\frac{1}{1+\rho}P(e), P(e)\right]$. Consider a message sent by node x at time t_s^β and received by node y at time t_r^β. Let t_s^α, t_r^α be the send and receive times (respectively) of the same message in execution α; that is,

$$H_x^\beta(t_s^\beta) = H_x^\alpha(t_s^\alpha) = t_s^\alpha, \qquad H_y^\beta(t_r^\beta) = H_y^\alpha(t_r^\alpha) = t_r^\alpha.$$

Using (1) we obtain

$$\begin{aligned} t_r^\alpha - t_s^\alpha &= H_y^\beta(t_r^\beta) - H_x^\beta(t_s^\beta) = \\ &= t_r^\beta + \min\left\{\rho t_r^\beta, \mathcal{T} \cdot \operatorname{dist}_M(u,y)\right\} - t_s^\beta - \min\left\{\rho t_s^\beta, \mathcal{T} \cdot \operatorname{dist}_M(u,x)\right\} \end{aligned} \tag{2}$$

We divide into cases.

- $\rho t_s^\beta \le \mathcal{T} \cdot \mathrm{dist}_M(u,x)$ and $\rho t_r^\beta \le \mathcal{T} \cdot \mathrm{dist}_M(u,y)$. In this case (2) implies

$$t_r^\alpha - t_s^\alpha = (1+\rho)(t_r^\beta - t_s^\beta)$$

By the definition of α we have $t_r^\alpha - t_s^\alpha \in [0,\mathcal{T}]$, and hence $t_r^\beta - t_s^\beta \in [0,\mathcal{T}]$ as well. In addition, if $\{x,y\} \in E^{\mathrm{C}}$ then $t_r^\alpha - t_s^\alpha = P(e)$ (again by definition of α); in this case we have $t_r^\beta - t_s^\beta = P(e)/(1+\rho) \in [P(e)/(1+\rho), P(e)]$, as required.

- $\rho t_s^\beta > \mathcal{T} \cdot \mathrm{dist}_M(u,x)$ and $\rho t_r^\beta > \mathcal{T} \cdot \mathrm{dist}_M(u,y)$. In this case (2) implies

$$t_r^\alpha - t_s^\alpha = t_r^\beta - t_s^\beta + \mathcal{T}(\mathrm{dist}_M(u,y) - \mathrm{dist}_M(u,x))$$

If $\{x,y\} \in E^{\mathrm{C}}$ or $x \equiv y$, then $\mathrm{dist}_M(u,x) = \mathrm{dist}_M(u,y)$, and $t_r^\beta - t_s^\beta = t_r^\alpha - t_s^\alpha = P(e) \in [P(e)/(1+\rho), P(e)] \subseteq [0,\mathcal{T}]$, as needed.

Otherwise, either $x \prec y$ and $\mathrm{dist}_M(u,y) - \mathrm{dist}_M(u,x) = 1$, or $y \prec x$ and $\mathrm{dist}_M(u,y) - \mathrm{dist}_M(u,x) = -1$. In the first case we have $t_r^\beta - t_s^\beta = t_r^\alpha - t_s^\alpha - \mathcal{T} = \mathcal{T} - \mathcal{T} = 0$ (by definition of α), and in the second case, $t_r^\beta - t_s^\beta = t_r^\alpha - t_s^\alpha + \mathcal{T} = 0 + \mathcal{T} = \mathcal{T}$. In both cases the delays are legal.

- $\rho t_s^\beta > \mathcal{T} \cdot \mathrm{dist}_M(u,x)$ and $\rho t_r^\beta \le \mathcal{T} \cdot \mathrm{dist}_M(u,y)$. In this case (2) implies

$$t_r^\alpha - t_s^\alpha = t_r^\beta - t_s^\beta + \rho t_r^\beta - \mathcal{T} \cdot \mathrm{dist}_M(u,x)$$

Since $\rho t_r^\beta \le \mathcal{T} \cdot \mathrm{dist}_M(u,y)$ and $\mathcal{T} \cdot \mathrm{dist}_M(u,x) < \rho t_s^\beta$, we can write

$$(1+\rho)(t_r^\beta - t_s^\beta) < t_r^\alpha - t_s^\alpha \le t_r^\beta - t_s^\beta + \mathcal{T}(\mathrm{dist}_M(u,y) - \mathrm{dist}_M(u,x))$$

If $\{x,y\} \in E^{\mathrm{C}}$ or $x \equiv y$, then $\mathrm{dist}_M(u,x) = \mathrm{dist}_M(u,y)$, and we obtain

$$(1+\rho)(t_r^\beta - t_s^\beta) < t_r^\alpha - t_s^\alpha \le t_r^\beta - t_s^\beta$$

which is impossible, because $t_r^\alpha - t_s^\alpha \ge 0$ and $\rho \ge 0$.

Otherwise, if $x \prec y$, then $\mathrm{dist}_M(u,y) = \mathrm{dist}_M(u,x) + 1$, and we have $t_r^\alpha - t_s^\alpha = \mathcal{T}$ and

$$(1+\rho)(t_r^\beta - t_s^\beta) < \mathcal{T} \le t_r^\beta - t_s^\beta + \mathcal{T}$$

It follows that $t_r^\beta - t_s^\beta \in [0, \mathcal{T}/(1+\rho)) \subseteq [0,\mathcal{T}]$.

Finally, if $y \prec x$, then $\mathrm{dist}_M(u,x) = \mathrm{dist}_M(u,y) + 1$, and we have $t_r^\alpha - t_s^\alpha = 0$ and

$$(1+\rho)(t_r^\beta - t_s^\beta) < 0 \le t_r^\beta - t_s^\beta - \mathcal{T}$$

But this is impossible, because it implies both $t_r^\beta - t_s^\beta < 0$ and $t_r^\beta - t_s^\beta \ge \mathcal{T}$.

- $\rho t_s^\beta \leq \mathcal{T} \cdot \mathrm{dist}_M(u, x)$ and $\rho t_r^\beta > \mathcal{T} \cdot \mathrm{dist}_M(u, y)$. In this case (2) implies

  $$t_r^\alpha - t_s^\alpha = t_r^\beta - t_s^\beta + \mathcal{T} \cdot \mathrm{dist}_M(u, y) - \rho t_s^\beta$$

  This time, we can re-write this to obtain

  $$t_r^\beta - t_s^\beta + \mathcal{T}(\mathrm{dist}_M(u, y) - \mathrm{dist}_M(u, x)) \leq t_r^\alpha - t_s^\alpha < (1+\rho)(t_r^\beta - t_s^\beta)$$

  If $\{x, y\} \in E^{\mathrm{C}}$ or $x \equiv y$, then again $\mathrm{dist}_M(u, x) = \mathrm{dist}_M(u, y)$, and we have

  $$t_r^\beta - t_s^\beta \leq t_r^\alpha - t_s^\alpha < (1+\rho)(t_r^\beta - t_s^\beta)$$

  which implies that $t_r^\beta - t_s^\beta \in [P(e)/(1+\rho), P(e)] \subseteq [0, \mathcal{T}]$, as required.

  If $x \prec y$ and $\mathrm{dist}_M(u, y) = \mathrm{dist}_M(u, x) + 1$, then we have

  $$t_r^\beta - t_s^\beta + \mathcal{T} \leq \mathcal{T} < (1+\rho)(t_r^\beta - t_s^\beta)$$

  which is a contradiction.

  And finally, if $y \prec x$ and $\mathrm{dist}_M(u, x) = \mathrm{dist}_M(u, y) + 1$, then

  $$t_r^\beta - t_s^\beta - \mathcal{T} \leq 0 < (1+\rho)(t_r^\beta - t_s^\beta)$$

  and it follows that $t_r^\beta - t_s^\beta \in (0, \mathcal{T}]$.

$\triangleright$ **Part III: the skew between u and v**

It remains only to show that in either α or β, the skew between u and v at time t is large. Since $\mathrm{dist}_M(u, v) = d$ and $t > (\mathcal{T} \cdot \mathrm{dist}_M(u, v))(1 + 1/\rho)$, at time $t' := t - \mathcal{T} \cdot \mathrm{dist}_M(u, v)$ we have

$$\begin{aligned} H_v^\beta(t') &= t' + \min\left\{\rho t', \mathcal{T} \cdot \mathrm{dist}_M(u, v)\right\} = t' + \mathcal{T} \cdot \mathrm{dist}_M(u, v) = \\ &= H_v^\alpha(t' + \mathcal{T} \cdot \mathrm{dist}_M(u, v)) = H_v^\alpha(t), \end{aligned}$$

while

$$H_u^\beta(t') = t' + \min\left\{\rho t', \mathcal{T} \cdot \mathrm{dist}_M(u, u)\right\} = t' = H_u^\alpha(t').$$

No node in the network can distinguish between α and β, and consequently, $L_w^\alpha(t_1) = L_w^\beta(t_2)$ iff $H_w^\alpha(t_1) = H_w^\beta(t_2)$ for all $w \in V$ and times t_1, t_2. In particular,

$$L_u^\alpha(t') = L_u^\beta(t') \tag{3}$$

and

$$L_v^\alpha(t) = L_v^\beta(t'). \tag{4}$$

Since u increases its logical clock at a rate of at least $1/2$,

$$L_u^\alpha(t) \geq L_u^\alpha(t') + \frac{1}{2}(t - t') = L_u^\beta(t') + \frac{1}{2}\mathcal{T} \cdot \mathrm{dist}_M(u, v), \tag{5}$$

and subtracting (4) from (5) yields

$$L_u^\alpha(t) - L_v^\alpha(t) \geq L_u^\beta(t') - L_v^\beta(t') + \frac{1}{2}\mathcal{T} \cdot \mathrm{dist}_M(u,v). \tag{6}$$

This implies that either $|L_u^\alpha(t) - L_v^\alpha(t)| \geq \frac{1}{4}\mathcal{T}\cdot\mathrm{dist}_M(u,v)$ or $\left|L_u^\beta(t) - L_v^\beta(t)\right| \geq \frac{1}{4}\mathcal{T}\cdot\mathrm{dist}_M(u,v)$. That is, in one of the two executions, u and v have at least the required skew. Since α is M-constrained by construction and we showed that β is also M-constrained, this proves the claim.

□

Lemma 4.3. *Let $X = x_1, \ldots, x_n$ be a sequence of numbers where $x_1 \leq x_n$ and for all $1 \leq i < n$, $|x_i - x_{i+1}| \leq d$ for some $d > 0$. Then for any $c > d$, there is a subsequence $X' = x_{i_1}, \ldots, x_{i_m} \subseteq X$ such that*

1. *$m \leq \frac{x_n - x_1}{c-d} + 1$, and*
2. *for all $1 \leq j \leq m-1$ we have $\left|x_{i_{j+1}} - x_{i_j}\right| \in [c-d, c]$.*

Proof. We construct a sequence $i_1, i_2, \ldots$ inductively, starting with $i_1 := 1$. Given i_j, we define

$$i_{j+1} := \min\left(\{n\} \cup \left\{\ell \mid i_j < \ell < n \text{ and } x_\ell - x_{i_j} \geq c - d \text{ and } x_\ell \leq x_n\right\}\right) \tag{7}$$

The sequence $i_1, i_2, \ldots$ is strictly increasing, and eventually it must reach n and stay at n forever. Let $m = \max\{j \mid i_j < n\}$. The sequence we return is $X' = x_{i_1}, \ldots, x_{i_m}$.

By construction, $x_1 = x_{i_1} \leq x_{i_2} \leq \ldots \leq x_{i_m} \leq x_n$, and for all $1 \leq i \leq m-1$ we have $x_{i_{j+1}} - x_{i_j} \geq c - d > 0$. It remains to prove the following.

1. $m \leq \frac{x_n - x_1}{c-d} + 1$: because

$$x_n - x_1 \geq x_{i_m} - x_{i_1} = \sum_{1 \leq j \leq m-1} \left(x_{i_{j+1}} - x_{i_j}\right) \geq (m-1)\cdot(c-d).$$

2. For all $1 \leq j \leq m-1$ we have $\left|x_{i_{j+1}} - x_{i_j}\right| \in [c-d, c]$: since $x_{i_{j+1}} - x_{i_j} \geq c - d > 0$, we need only to show that $x_{i_{j+1}} - x_{i_j} \leq c$. We consider two cases.

   I. $i_{j+1} = i_j + 1$: in this case we already know that $|x_{i_{j+1}} - x_{i_j}| \leq d$. Since $c > d$ the claim follows.

   II. $i_{j+1} > i_j + 1$: let $\ell > i_j$ be the minimal index such that $x_\ell - x_{i_j} \geq c - d$. By construction, $i_{j+1} > i_j$ is the minimal index that satisfies both $x_{i_{j+1}} - x_{i_j} \geq c - d$ and $x_{i_{j+1}} \leq x_n$; hence, $i_{j+1} \geq \ell$, and if $i_{j+1} > \ell$ then $x_\ell > x_n$. It follows that $x_{i_{j+1}} \leq x_\ell$. Since ℓ is the minimal index for which $x_\ell - x_{i_j} \geq c - d$, for index $\ell - 1$ we have $x_{\ell-1} - x_{i_j} < c - d$. In addition, $x_\ell - x_{\ell-1} \leq d$. Together we have $x_{i_{j+1}} - x_{i_j} \leq x_\ell - x_{i_j} = x_\ell - x_{\ell-1} + x_{\ell-1} - x_{i_j} \leq d + c - d = c$, as required.

□

Full Proof of Theorem 4.1. Let $\delta = \frac{\mathcal{T}}{128}$ and $\xi = 1 + \frac{\mathcal{T}}{3\bar{s}(n)}$, and define $k = \delta \frac{n}{\bar{s}(n)}$. We assume that n is large enough that the following requirements are satisfied.

- $k \geq 1$: since $\bar{s}(n) = o(n)$, we can choose n large enough so that $\bar{s}(n) \leq \delta n$ and $k \geq 1$.
- $\bar{s}(n) \geq \mathcal{T}$: since $\bar{s}(n) = \Omega(\mathcal{T} \cdot \log n)$ in a network with diameter $\Omega(n)$, for sufficiently large n we have $\bar{s}(n) \geq \mathcal{T}$.
- $\xi \in (1, \frac{4}{3}]$: this follows from the previous requirement.
- $n - 2(k+1) > 0$: it is sufficient to require $\bar{s}(n) > 4\delta$, which is implied by $\bar{s}(n) \geq \mathcal{T}$.

▷ **Part I: setup**

Consider the network N shown in Fig. 1(a), over nodes $V = \{w_0, w_n\} \cup (I_A \times \{A\}) \cup (I_B \times \{B\})$, where

$$\begin{aligned} I_A &= \{1, \ldots, \lfloor n/2 \rfloor - 1\}\,, \text{ and} \\ I_B &= \{1, \ldots, \lceil n/2 \rceil - 1\}\,. \end{aligned}$$

For the sake of convenience we also use $\langle 0, A\rangle$ and $\langle 0, B\rangle$ to refer to node w_0, and $\langle \lfloor n/2 \rfloor, A\rangle$ and $\langle \lceil n/2 \rceil, B\rangle$ to refer to node w_n.

Using this notation, the initial set of edges is given by

$$\begin{aligned} E = \{(\langle i, A\rangle, \langle i+1, A\rangle) \mid i \in I_A \text{ or } i+1 \in I_A\} \cup \\ \cup \{(\langle i, B\rangle, \langle i+1, B\rangle) \mid i \in I_B \text{ or } i+1 \in I_B\}\,. \end{aligned}$$

Let $u = \langle \lceil k \rceil, A\rangle$ and $v = \langle \lfloor n/2 - k \rfloor, A\rangle$. The distance between u and v is at least $n/2 - 2(k+1)$, and the distance between nodes 0 and u and between nodes v and n is at least k.

We use E^{block} to denote the set of edges on the shortest path between nodes 0 and u and between nodes v and n (these edges are covered by the double-sided arrows in Fig. 1(a)). Formally,

$$E^{\text{block}} = E \cap \{\langle i, A\rangle \mid 0 \leq i \leq \lceil k \rceil \text{ or } \lfloor n/2 - k \rfloor \leq i \leq n\}^{(2)}$$

Let $S = \xi \cdot \bar{s}(n)$. By definition, $\bar{s}(n) = \lim_{t \to \infty} s(n, 0, t)$. In particular, there is some T_1 such that for all $t \geq T_1$ we have $s(n, 0, t) \leq S$.

Let $T_2 = T_1 + k\frac{\mathcal{T}}{1+\rho}$. Consider a delay mask $M = (E^{\text{block}}, \mathcal{T})$. By Lemma 4.2, there is an M-constrained execution α in which

$$|L_u(T_2) - L_v(T_2)| \geq \frac{1}{4}\mathcal{T} \cdot \operatorname{dist}_M(u, v) \geq \frac{1}{4}\mathcal{T}\left(\frac{n}{2} - 2(k+1)\right). \tag{8}$$

▷ **Part II: the skew between nodes 0 and n**

We will now show that the large skew between u and v implies a large skew between nodes 0 and n. Let $S_2 = |L_0(T_2) - L_n(T_2)|$. We proceed to bound S_2 from below.

By choice of T_2 we have $s(n, 0, T_2) \leq S$. Since s is non-decreasing in the initial skew, this implies that the skew on each of the edges between nodes 0 and u and between nodes v and n is at most S. There are at most $k+1$ edges between each pair, and hence

$$|L_0(T_2) - L_u(T_2)| \leq S \cdot (k+1) = \xi \bar{s}(n)(k+1) \tag{9}$$

and

$$|L_v(T_2) - L_n(T_2)| \leq S \cdot (k+1) = \xi \bar{s}(n)(k+1). \tag{10}$$

Combining with the known skew between u and v (Eq. 8) yields

$$\begin{aligned}
S_2 = |L_0(T_2) - L_n(T_2)| &\geq |L_u(T_2) - L_v(T_2)| - |L_0(T_2) - L_u(T_2)| - |L_v(T_2) - L_n(T_2)| \geq \\
&\geq \frac{1}{8} n\mathcal{T} - \frac{1}{2}\mathcal{T}(k+1) - 2\xi\bar{s}(n)(k+1) \geq && (\xi\bar{s}(n) \geq \mathcal{T}) \\
&\geq \frac{1}{8} n\mathcal{T} - 3\xi\bar{s}(n)(k+1) \geq && (k \geq 1, \xi \leq \tfrac{4}{3}) \\
&\geq \frac{1}{8} n\mathcal{T} - 8k\bar{s}(n) = && (k = \delta \tfrac{n}{\bar{s}(n)}) \\
&= \left(\frac{1}{8}\mathcal{T} - 8\delta\right) \cdot n = && (\delta = \tfrac{\mathcal{T}}{128}) \\
&= \frac{1}{16} n\mathcal{T}.
\end{aligned}$$

▷ **Part III: adding new edges**

We now construct another execution β, in which new edges E^{new} appear at time $T_1 = T_2 - k \cdot \frac{\mathcal{T}}{1+\rho}$. Formally, the network in execution β is defined by

$$E^{\beta}(t) = \begin{cases} E^{\alpha}(t) & t < T_1 \\ E^{\alpha}(t) \cup E^{\text{new}} & t \geq T_1 \end{cases}$$

In β, a $\mathsf{discover}(\{u, v\})$ event occurs at time $T_1 + \mathcal{D}$ at every node u such that $\{u, v\} \in E^{\text{new}}$ for some $v \in V$. All message delays on edges in E and all hardware clock rates are the same in α and in β. Message delays on edges in E^{new} in β are chosen arbitrarily. Note that since α is M-constrained, β is M-constrained as well.

The new edges E^{new} are chosen between nodes on the B-chain using Lemma 4.3. For any adjacent nodes x, y on the B-chain we have $|L_x(T_1) - L_y(T_1)| \leq S$. Therefore, by Lemma 4.3, there is a sequence $X' = x_1, \ldots, x_m$ of B-chain nodes such that

1. For all $1 \leq i \leq m-1$ we have $\left|L_{x_i}(T_1) - L_{x_{i+1}}(T_1)\right| \in [I - S, I]$, and

2. $m \leq \frac{|L_0(T_1)-L_n(T_1)|}{I-S} + 1$.

Set $E^{\text{new}} = \{\{x_i, x_{i+1}\} \mid 1 \leq i \leq m-1\}$. Then

$$|E^{\text{new}}| = m - 1 \leq \frac{|L_0(T_1) - L_n(T_1)|}{I - S} \leq \frac{\bar{\mathcal{G}}(n)}{I - S},$$

where in the last step we used the fact that the global skew is bounded by $\bar{\mathcal{G}}(n)$.

$\triangleright$ **Part IV: indistinguishability of α and β**

We show by induction on i that for all $0 \leq i \leq k$, executions α and β are indistinguishable up to time $t_i := T_1 + i \cdot \frac{\mathcal{T}}{1+\rho} + \mathcal{D}$, exclusive, in the eyes of all nodes in the set

$$Y_i = \{\langle j, A\rangle \mid i \leq j \leq \lfloor n/2 \rfloor - i\}.$$

- (Base.) For $i = 0$ the claim follows from the fact that α and β are identical up to time T_1 (exclusive), and no node finds out about the new edges until time $T_1 + \mathcal{D}$.

- (Step.) Suppose that up to time t_i, exclusive, executions α and β are indistinguishable in the eyes of all nodes in the set $Y_i = \{\langle j, A\rangle \mid i \leq j \leq \lfloor n/2 \rfloor - i\}$. Let $u \in Y_{i+1}$. From the definition of Y_i and Y_{i+1}, node u and its neighbors are in Y_i. Thus, at any time $t < t_i$, neither u nor its neighbors can distinguish between α and β.

  Since message delays and the hardware clocks of all nodes are the same in α and in β, and no nodes in Y_i experience link formations or failures, the only way a node in Y_i could distinguish between executions α and β is by receiving a message from a node that previously could distinguish between α and β. We show that no node in Y_{i+1} can receive a message from a node that distinguishes α from β until time t_{i+1} (exclusive).

  Consider first messages sent by a node $v \in Y_i \setminus Y_{i+1}$ and received by $u \in Y_{i+1}$ at some time $t_r < t_{i+1}$. Let t_s be the time at which v sent the message. Because $i + 1 \leq k$, the edge $\{u, v\}$ must be in E^{block}, and since β is M-constrained this means that $t_s \leq t_r - \frac{\mathcal{T}}{1+\rho} < t_{i+1} - \frac{\mathcal{T}}{1+\rho} = t_i$. Thus, the message was sent prior to time t_i, and node v could not distinguish between α and β when it sent the message.

  As for messages sent between nodes in Y_{i+1}, it is easy to show by induction on the number of such messages received that neither sender nor recipient can distinguish between α and β.

Since $u, v \in Y_k$ and $T_2 = T_1 + k\frac{\mathcal{T}}{1+\rho} < T_1 + k\frac{\mathcal{T}}{1+\rho} + \mathcal{D}$, nodes u and v cannot distinguish between α and β at any time $t \leq T_2$. It follows that u and v will have the same logical clocks at time T_2 in β as they do in α, and the skew between them will be S_2.

$\triangleright$ **Part V: the skew on the new edges at time T_2**

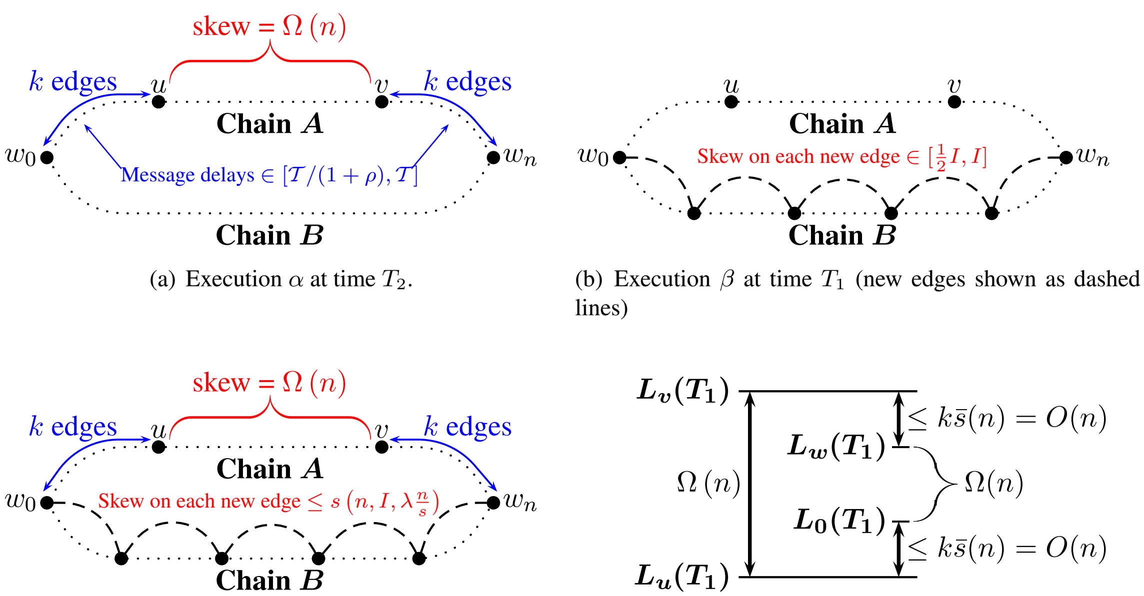
At time T_2, every edge in E^{new} carries a skew of no more than $s(n, I, T_2 - T_1)$, since the initial skew on every edge was no more than I and s is non-decreasing in the initial skew. Consequently, the total skew between the endpoints at time T_2 satisfies $S_2 \le |E^{\text{new}}| \cdot s(n, I, T_2 - T_1)$. However, we have shown that $S_2 \ge \frac{1}{16} n\mathcal{T}$, and hence

$$\frac{1}{16} n\mathcal{T} \le S_2 \le |E^{\text{new}}| \cdot s(n, I, T_2 - T_1) \le \frac{\bar{\mathcal{G}}(n)}{I - S} \cdot s(n, I, k \cdot \frac{\mathcal{T}}{1+\rho}).$$

Rearranging the terms and substituting $k = \delta \frac{n}{\bar{s}(n)}$ and $\delta = \frac{\mathcal{T}}{128}$ yields

$$s(n, I, \frac{\mathcal{T}}{128(1+\rho)} \cdot \frac{n}{\bar{s}(n)} \mathcal{T}) \ge \frac{n}{16\bar{\mathcal{G}}(n)} \mathcal{T}(I - S) \ge \frac{n}{32\bar{\mathcal{G}}(n)} \mathcal{T} \cdot I.$$

For the bound to be meaningful we must have $\frac{n}{32\bar{\mathcal{G}}(n)} \mathcal{T} \cdot I > \bar{s}(n)$, that is, $I > \frac{32\bar{\mathcal{G}}(n)\bar{s}(n)}{\mathcal{T} n}$. $\square$



(a) Execution α at time T_2.

(b) Execution β at time T_1 (new edges shown as dashed lines)

(c) Execution β at time T_2

(d) The logical clocks of w_0, u, v, w_n at time T_2 in executions α and β (assuming $L_u(T_1) \le L_v(T_1)$)

Figure 1: Illustrations for the proof of Theorem 4.1

5 A Dynamic Clock Synchronization Algorithm

Next we present a simple DCSA that achieves the tradeoff demonstrated in the previous section.

The algorithm is based on nodes sending each other periodic updates containing their own logical clock value and their estimate for the maximal logical clock in the network. Updates are sent to all neighbors every ΔH subjective time units; that is, if node u sends an update to all its

neighbors at real time t, the next time it will send an update is real time t' such that $H_u(t') = H_u(t) + \Delta H$. Since the hardware clock of u progresses at a rate of at least $1 - \rho$, updates are sent at least once every $\frac{\Delta H}{1-\rho}$ real time units.
Define

$$\Delta\mathcal{T} := \mathcal{T} + \frac{\Delta H}{1-\rho}, \qquad \Delta\mathcal{T}' := (1+\rho)\,\Delta\mathcal{T}.$$

Since every node sends messages to all its neighbors at least once every $\frac{\Delta H}{1-\rho}$ time units, and messages take at most $\mathcal{T}$ time units to arrive, $\Delta\mathcal{T}$ is the longest amount of real time that may pass between the receipt of two messages along an edge, provided the edge exists throughout the interval. Since nodes do not have access to real time, they use $\Delta\mathcal{T}'$ to conservatively estimate the time they have to wait between receiving two messages from a neighbor. If $\Delta\mathcal{T}'$ subjective time has passed and a message was not received, the link to that neighbor must have failed.

The algorithm we present here is event-based: nodes react to messages they receive and to $\mathsf{discover}(X)$ events, where $X \in \{\mathsf{add}(\{u,v\}), \mathsf{remove}(\{u,v\}) \mid v \in V\}$. In addition, each node can set a timer to trigger a delayed event by calling $\mathsf{set\_timer}(\Delta t, \text{timer-ID})$. If $\mathsf{set\_timer}(\Delta t, \text{timer-ID})$ is called by u at real time t, then at real time t' such that $H_u(t') = H_u(t) + \Delta t$, an $\mathsf{alarm}(\text{timer-ID})$ event is triggered at node u. A delayed event can be cancelled using the function $\mathsf{cancel}(\text{timer-ID})$.

The algorithm uses two types of timers: the tick timer is set to go off every subjective ΔH time, and a $\mathsf{lost}(v)$ timer is set to go off $\Delta\mathcal{T}'$ subjective time units after a message from v is received.

Throughout the run of the algorithm each node u maintains two sets Γ_u, Υ_u such that $\Gamma_u \subseteq \Upsilon_u$. The set Υ_u contains all the nodes v such that u believes the edge $\{u,v\}$ exists; that is, all the nodes v such that a $\mathsf{discover}(\mathsf{add}(\{u,v\}))$ event occurred at u and was not yet followed by a $\mathsf{discover}(\mathsf{remove}(\{u,v\}))$ event. The criterion for membership in Γ_u is more restrictive: the nodes in Γ_u are those nodes of Υ_u that u has heard from at most $\Delta\mathcal{T}'$ subjective time units ago. If $\Delta\mathcal{T}'$ subjective time units pass and u does not receive a message from v, then v is removed from Γ_u (but not from Υ_u). The nodes in Γ_u are the only ones used to determine u's logical clock value, since they are the ones for which u has an accurate estimate. However, u sends (or tries to send) periodic updates to all nodes in Υ_u.

In addition to Γ_u and Υ_u, node u maintains the following local variables.

L_u	Node u's logical clock
$L_u^{\max}$	Node u's estimate for the maximal logical clock in the network
C_u^v for $v \in \Gamma_u$	The value of node u's hardware clock when v was last added to Γ_u
L_u^v for $v \in \Gamma_u$	Node u's estimate for node v's current logical clock

The local variables are modified upon processing the various events as shown in Algorithm 2. Between events, the variables L_u, $L_u^{\max}$ and L_u^v for all $v \in \Gamma_u$ are increased at the rate of u's hardware clock.

Node u uses a non-increasing function $B : \mathbb{R}^+ \to \mathbb{R}^+$ to determine how much perceived skew it is willing to tolerate on the edge $\{u,v\}$ for $v \in \Gamma_u$. The parameter to the function is $(H_u - C_u^v)$, the subjective amount of time that has passed since u discovered the edge. For a parameter B_0, the function B is given by

$$B(\Delta t) := \max\left\{B_0, 5\mathcal{G}(n) + (1+\rho)\tau + B_0 - \frac{B_0}{(1+\rho)\tau}\Delta t\right\},$$

where

$$\tau := \frac{1+\rho}{1-\rho}\Delta\mathcal{T} + \mathcal{T} + \mathcal{D}, \quad B_0 > 2(1+\rho)\tau,$$

and where $\mathcal{G}(n)$ is the bound on the global skew derived in Theorem 6.9. The parameter B_0 can be interpreted as the local skew that the algorithm is trying to maintain on every edge, once the edge is in the system for a sufficiently long time.

The logical clock of each node is adjusted after every event. In each adjustment, node u increases L_u to the largest value that it can, subject to the following constraints: (1) L_u is never decreased, (2) L_u cannot exceed $L_u^{\max}$, and (3) the perceived skew on every edge $\{u, v\}$ such that $v \in \Gamma_u$ cannot exceed the value of B for that edge. That is, for all $v \in \Gamma_u$ we require $L_u - L_u^v \leq B\left(H_u - C_u^v\right)$. If the constraints cannot be met (e.g., if u has a neighbor that is very far behind), node u cannot make a discrete increase to its logical clock. However, the logical clock continues to increase at the rate of u's hardware clock. The update rule is given by

Procedure `AdjustClock`

1 $L_u \leftarrow \max\left\{L_u, \min\left\{L_u^{\max}, \min_{v\in\Gamma_u}\left\{L_u^v + B(H_u - C_u^v)\right\}\right\}\right\}$

For simplicity, we assume that all nodes know (upper bounds on) the maximum hardware clock drift ρ, the propagation delay $\mathcal{T}$, as well as the bound $\mathcal{D}$ on the time between topology changes and the nodes discovering these changes. Depending on how edge insertions and deletions are discovered, $\mathcal{D}$ typically is a function of ρ, $\mathcal{T}$, as well as the parameter ΔH. Throughout the remainder of the paper, we assume that $\mathcal{D} > \max\{\mathcal{T}, \Delta H/(1-\rho)\}$. We also assume that all nodes know n, the number of nodes participating in the system. With these assumptions, each node u knows enough to compute the value of B_u^v for every $v \in \Gamma_u$. In particular, all nodes can compute the bound $\mathcal{G}(n)$ on the global skew. Note that the same asymptotic results can be achieved if all nodes know n up to a constant factor. This would allow to generalize the setting and also adapt to nodes joining and leaving the system as long as n only changes at a constant rate.

Algorithm 2: Responses to events that occur at node u

when $\mathsf{discover}(\mathsf{add}(\{u, v\}))$ occurs at u
  $\mathsf{send}(u, v, \langle L_u, L_u^{\max} \rangle)$
  $\Upsilon_u \leftarrow \Upsilon_u \cup \{v\}$
  `AdjustClock()`
end
when $\mathsf{discover}(\mathsf{remove}(\{u, v\}))$ occurs at u
  $\Gamma_u \leftarrow \Gamma_u \setminus \{v\}$
  $\Upsilon_u \leftarrow \Upsilon_u \setminus \{v\}$
  `AdjustClock()`
end
when $\mathsf{alarm}(\mathsf{lost}(v))$ occurs at u
  $\Gamma_u \leftarrow \Gamma_u \setminus \{v\}$
  `AdjustClock()`
end
when $\mathsf{receive}(u, v, \langle L_v, L_v^{\max} \rangle)$ occurs at u
  $\mathsf{cancel}(\mathsf{lost}(v))$
  if $v \notin \Gamma_u$ **then**
    $\Gamma_u \leftarrow \Gamma_u \cup \{v\}$
    $C_u^v \leftarrow H_u$
  $L_u^v \leftarrow L_v$
  $L_u^{\max} \leftarrow \max\{L_u^{\max}, L_v^{\max}\}$
  `AdjustClock()`
  $\mathsf{set\_timer}(\Delta \mathcal{T}', \mathsf{lost}(v))$
end
when $\mathsf{alarm}(\mathsf{tick})$ occurs at u
  forall $v \in \Upsilon_u$ **do**
    $\mathsf{send}(u, v, \langle L_u, L_u^{\max} \rangle)$
  `AdjustClock()`
  $\mathsf{set\_timer}(\Delta H, \mathsf{tick})$
end

6 Analysis of the Algorithm

6.1 Basic properties

To analyze the algorithm it is important to understand what conditions prevent nodes from making discrete changes to their logical clocks. The following definitions and lemmas characterize these conditions and describe basic properties of the algorithm.
Let

$$B_u^v(t) := B(H_u(t) - C_u^v(t))$$

be the amount of perceived skew node u is willing to tolerate on the edge $\{u, v\}$ at real time t.

Definition 6.1 (Blocked nodes)**.** *We say that a node u is* blocked by node v *at time t if $L_u^{\max}(t) > L_u(t)$ and $v \in \Gamma_u(t)$ and $L_u(t) - L_u^v(t) > B_u^v(t)$. In this case we also say that node v blocks node u at time t and that node u is blocked at time t.*

Property 6.1. *If $v \in \Gamma_u(t)$, then by time t node u has received at least one message that node v sent at time $t_s \geq t - \tau$.*

Proof. If $v \in \Gamma_u(t)$ then u has received a message from v at some time t_r such that $H_u(t) - H_u(t_r) \leq (1 + \rho)\Delta\mathcal{T}$, otherwise u would have removed v from Γ_u prior to time t. Since the hardware clock rate of u is at least $1 - \rho$,

$$H_u(t) - H_u(t_r) \geq (1 - \rho)(t - t_r).$$

Thus, $t - t_r \leq \frac{1+\rho}{1-\rho}\Delta\mathcal{T}$. The message was sent at some time $t_s \geq t_r - \mathcal{T} \geq t - \frac{1+\rho}{1-\rho}\Delta\mathcal{T} - \mathcal{T} \geq t - \tau$, so the property holds. ☐

Property 6.2. *If edge $\{u, v\}$ exists throughout the interval $[t_1, t_2]$ where $t_2 \geq t_1 + \mathcal{D}$, then for any time t such that $t_1 + \Delta\mathcal{T} + \mathcal{D} \leq t \leq t_2$,*

1. *$u \in \Gamma_v(t)$ and $v \in \Gamma_u(t)$,*
2. *Node u receives at least one message from v in the interval $[t - \Delta\mathcal{T}, t]$, and*
3. *Node v receives at least one message from u in the interval $[t - \Delta\mathcal{T}, t]$.*

Proof. Since the edge $\{u, v\}$ exists throughout the interval $[t_1, t_2]$ where $t_2 \geq t_1 + \mathcal{D}$, it is discovered by u and v at times t_u^0, t_v^0 respectively such that $t_u^0, t_v^0 \leq t_1 + \mathcal{D}$.

Upon discovering the edge nodes u and v add each other to Υ_u and Υ_v respectively. No $\mathsf{discover}(\mathsf{remove}(u, v))$ event can occur at u or at v between times t_u^0, t_v^0 (respectively) and time t_2, because the edge exists throughout the interval $[t_1, t_2]$. Therefore, for all $t \in [t_1 + \mathcal{D}, t_2]$ we have $v \in \Upsilon_u$ and $u \in \Upsilon_v$. It follows that nodes u and v send each other updates every subjective ΔH time units at most throughout the interval $[t_1 + \mathcal{D}, t_2]$. This implies that u and v send each other updates every objective $\frac{\Delta H}{1-\rho}$ time units at most throughout this interval.

Let t be a time such that $[t - \Delta\mathcal{T}, t] \subseteq [t_1 + \mathcal{D}, t_2]$. Since v sends u a message at least once every $\frac{\Delta H}{1-\rho}$ time units throughout the interval $[t_1 + \mathcal{D}, t_2]$, there is some $t_s \in [t - \Delta\mathcal{T}, t - \Delta\mathcal{T} + \frac{\Delta H}{1-\rho}]$ such that v sends u a message at time t_s. The message is received by u at time t_r such that

$$t - \Delta\mathcal{T} \leq t_s \leq t_r \leq t_s + \mathcal{T} \leq t - \Delta\mathcal{T} + \frac{\Delta H}{1-\rho} + \mathcal{T} = t.$$

Therefore, condition 2 of the lemma is satisfied. Condition 3 is similar.

Condition 1 of the lemma follows from conditions 2 and 3: from lines 6–22 of the algorithm, if u received a message from v at time t_r such that $H_u(t) - H_u(t_r) \leq (1+\rho)\Delta\mathcal{T}$ and no $\mathsf{discover}(\mathsf{remove}(\{u,v\}))$ event occurs during the interval $[t_r, t]$, then $v \in \Gamma_u(t)$. Let t be a time such that $[t - \Delta\mathcal{T}, t] \subseteq [t_1 + \mathcal{D}, t_2]$. Condition 2 of the lemma shows that node u receives a message from node v at some time $t_r \in [t - \Delta\mathcal{T}, t]$. In particular, $H_u(t) - H_u(t_r) \leq (1+\rho)(t - t_r) \leq (1+\rho)(t - (t - \Delta\mathcal{T})) = (1+\rho)\Delta\mathcal{T}$. Finally, we know that no $\mathsf{discover}(\mathsf{remove}(\{u,v\}))$ event occurs during the interval $[t_r, t]$, because the edge $\{u,v\}$ exists throughout the interval $[t_1, t_2]$ and $[t_r, t] \subseteq [t - \Delta\mathcal{T}, t] \subseteq [t_1 + \mathcal{D}, t_2]$. □

Property 6.3 (Max estimates)**.** *For all $u \in V$ and times $t \geq 0$,*

$$L_u^{\max}(t) \geq L_u(t).$$

Proof. The variables $L_u^{\max}$ and L_u are modified in three circumstances: in line 21 of the algorithm, which is executed when u receives a message; in procedure `AdjustClock()`, which is called after every event; and in between discrete events. It is sufficient to show that all of these preserve the invariant $L_u^{\max} \geq L_u$.

Between processing discrete events, $L_u^{\max}$ and L_u are both increased at the rate of u's hardware clock, and the invariant is preserved. Suppose then that $L_u^{\max} \geq L_u$ prior to the execution of line 21 or of procedure `AdjustClock()`. In line 21 the value of $L_u^{\max}$ can only be increased, so the invariant is preserved. In `AdjustClock()`, node u sets

$$L_u \leftarrow \max\left\{L_u, \min\left\{L_u^{\max}, \ldots\right\}\right\}.$$

Since we assume that $L_u^{\max} \geq L_u$ prior to the execution of `AdjustClock()`, both terms in the $\max$ are no greater than $L_u^{\max}$. Following the assignment we still have $L_u \leq L_u^{\max}$.

□

Property 6.4 (Sufficient condition to be blocked)**.** *If $L_u^{\max}(t) > L_u(t)$, then node u is blocked at time t.*

Proof. Let $t' \leq t$ be the most recent time a discrete event occurs at node u up to (and including) time t.

Between time t' and time t node u increases $L_u^{\max}$ and L_u at the rate of its hardware clock, and therefore $L_u^{\max}(t') - L_u(t') = L_u^{\max}(t) - L_u(t)$. Since we assume that $L_u^{\max}(t) > L_u(t)$ it follows that $L_u^{\max}(t') > L_u(t')$.

Node u must be blocked following the last event that occurs at time t', otherwise it would have set $L_u(t') \leftarrow L_u^{\max}(t')$ in Procedure `AdjustClock()` after processing the last event. Thus, there is some neighbor $v \in \Gamma_u(t')$ such that $L_u(t') - L_u^v(t') > B_u^v(t')$.

Between time t' and time t node v was not removed from Γ_u, because nodes are only removed from Γ_u following discrete events and no discrete event occurs at node u between the last event that occurs at time t' and time t. Thus, $v \in \Gamma_u(t)$. Also, between times t' and t, the values L_u and L_u^v were both increased at the rate of u's hardware clocks, and hence $L_u(t) - L_u^v(t) = L_u(t') - L_u^v(t') > B_u^v(t') \geq B_u^v(t)$. This shows that node v blocks node u at time t. □

Lemma 6.5 (Estimate quality)**.** *If $v \in \Gamma_u(t)$ then $L_u^v(t) \geq L_v(t - \tau)$.*

Proof. Let t_s be the latest time that node v sends a message $\langle L_v(t_s), L_v^{\max}(t_s) \rangle$ which is received by u at some time $t_r \leq t$. Upon receiving the message node u sets $L_u^v \leftarrow L_v(t_s)$ (line 20).

Since messages are delivered in FIFO fashion, node u does not receive another message from v during the interval $(t_r, t]$; during this interval L_u^v is increased at the rate of u's hardware clock, and in particular, L_u^v is not decreased. Finally, from Property 6.1 we have $t_s \geq t - \tau$, and therefore

$$L_u^v(t) \geq L_u^v(t_r) \geq L_v(t_s) \geq L_v(t - \tau).$$

□

Each node u decides whether or not to increase its clock based on its estimates of its neighbors' clocks, aiming to keep the skew on edge $\{u, v\}$ no greater than B_u^v. Since the estimate may be larger than the real value of the neighbor's clock, node u may overshoot the mark, but the following lemma shows that it does not overshoot it by much.

Lemma 6.6. *If u's logical clock made a discrete jump at time t, then immediately following the jump, for all $v \in \Gamma_u(t)$ we have $L_u(t) - L_v(t) \leq B_u^v(t) + 2\rho \cdot \tau$.*

Proof. If u's logical clock made a discrete jump at time t, then following the jump in Procedure `AdjustClock()` we have

$$L_u(t) \leq \min_{v \in \Gamma_u} \left(L_u^v(t) + B_u^v(t) \right). \tag{11}$$

Let $v \in \Gamma_u$. From Property 6.1, there is a time $t_r \leq t$ such that at time t_r node u receives a message sent by node v at time $t_s \geq t - \tau$. Let t_s be the last time v sent a message that u received by time $t_r \leq t$. The message carries $L_v(t_s)$, and following the receipt of the message we have $L_u^v(t_r) = L_v(t_s)$. Between time t_r and time t, node u increases L_u^v at the rate of its own hardware clock, and hence

$$L_u^v(t) \leq L_u^v(t_r) + (t - t_r)(1 + \rho) \leq L_v(t_s) + (t - t_s)(1 + \rho). \tag{12}$$

Also, node v increases its logical clock at a rate of at least $(1-\rho)$, and consequently,

$$L_v(t_s) \le L_v(t) - (t - t_s)(1-\rho). \tag{13}$$

Combining (12) and (13) yields

$$L_u^v(t) \le L_v(t) + 2\rho(t - t_s) \le L_v(t) + 2\rho\tau, \tag{14}$$

and from (14) and (11) we obtain

$$L_u(t) - L_v(t) \le L_u(t) - L_u^v(t) + 2\rho\tau \le B_u^v(t) + 2\rho\tau.$$

□

6.2 Global Skew

The basic strategy to bound the global skew of our dynamic clock synchronization algorithm is the same as the one used in a static network (see [13]). We first show that for any two nodes u and v, the estimates $L_u^{\max}(t)$ and $L_v^{\max}(t)$ of the maximum clock value in the system are not too far apart. Second, we show that if the global skew exceeds a certain value at time t, the node v with the smallest logical clock value $L_v(t)$ cannot be blocked at time t. By Lemma 6.4, we then have $L_v(t) = L_v^{\max}(t)$ and thus the bound on the maximal difference between two estimates $L_u^{\max}(t)$ and $L_v^{\max}(t)$ also yields a bound on the global skew.

For any $t \ge 0$, define

$$L^{\max}(t) := \max_{u \in V} L_u^{\max}(t). \tag{15}$$

Property 6.7 (Rate of $L^{\max}$)**.** *The value of $L^{\max}$ increases at a rate at most $1+\rho$. That is, for all $t_2 \ge t_1 \ge 0$ we have*

$$L^{\max}(t_2) - L^{\max}(t_1) \le (1+\rho)(t_2 - t_1)$$

Proof. First observe that no discrete change to $L_u^{\max}$ made by a node u changes the value of $L^{\max}$: nodes only make discrete changes to $L_u^{\max}$ in line 21 of the algorithm, following receipt of a message from some neighbor. Suppose that at time t_r node u receives a message that was sent by node v at time $t_s \le t_r$. In line 21, node u sets $L_u^{\max} \leftarrow \max\{L_u^{\max}, L_v^{\max}(t_s)\}$; this assignment can neither increase $L^{\max}$ nor decrease it, and hence it remains unchanged. (Note that v never decreases $L_v^{\max}$, so the value of $L^{\max}$ prior to the assignment is at least $L_v^{\max}(t_s)$.)

The proof of the claim is by induction on the total number of times line 21 is executed by all nodes in the network. (We also refer to each execution of line 21 as "a discrete change".) Suppose first that during the interval $[t_1, t_2]$ no node ever executes line 21, but $L^{\max}(t_2) - L^{\max}(t_1) > (1+\rho)(t_2 - t_1)$. Let u be a node such that $L_u^{\max}(t_2) = L^{\max}(t_2)$. Throughout the interval $[t_1, t_2]$, the value of $L_u^{\max}$ is increased at u's hardware clock rate, and therefore $L_u^{\max}(t_1) \ge L_u^{\max}(t_2) - (1+\rho)(t_2 - t_1) = L^{\max}(t_2) - (1+\rho)(t_2 - t_1) > L^{\max}(t_1)$. This contradicts the definition of $L^{\max}(t_1)$.

Now suppose that the claim holds when at most k discrete changes to local $L^{\max}$ variables are made, and consider an interval $[t_1, t_2]$ during which $k+1$ discrete changes occur. Let $t \in [t_1, t_2]$ be the last time a discrete change occurs, and let u be a node such that $L_t^{\max}(t_2) = L^{\max}(t_2)$. Since discrete changes do not affect the value of $L^{\max}$, we can apply the induction hypothesis to obtain $L^{\max}(t) - L^{\max}(t_1) \leq (1+\rho)(t - t_1)$. In addition, since u makes no discrete change to $L_u^{\max}$ during the interval $(t, t_2]$, it holds that $L^{\max}(t_2) - L_u^{\max}(t) = L_u^{\max}(t_2) - L_u^{\max}(t) \leq (1+\rho)(t_2 - t)$. Finally, by definition, $L^{\max}(t) \geq L_u^{\max}(t)$. Combining the three inequalities yields $L^{\max}(t_2) - L^{\max}(t_1) \leq (1+\rho)(t_2 - t_1)$. □

The accuracy of the estimates $L_u^{\max}(t)$ can be bounded by applying the interval connectivity property of the dynamic network graph. This is stated by the following lemma.

Lemma 6.8 (Max Propagation Lemma)**.** *If the dynamic graph $G(t)$ is $(\mathcal{T}+\mathcal{D})$-interval connected, then for all $t \geq 0$ and all $u \in V$ it holds that*

$$L^{\max}(t) - L_u^{\max}(t) \leq ((1+\rho) \cdot \mathcal{T} + 2\rho \cdot \mathcal{D}) \cdot (n-1).$$

Proof Sketch. Let v be a node for which $L_v^{\max}(t-(n-1)(\mathcal{T}+\mathcal{D})) = L^{\max}(t-(n-1)(\mathcal{T}+\mathcal{D}))$. We want to show that v's estimate $L_v^{\max}$ at time $t-(n-1)(\mathcal{T}+\mathcal{D})$ or a larger estimate has reached every node u by time t. On the way from v to u, the estimate is increased at at the rate of some hardware clock whenever it is not in transit between two nodes. Because the total transit time between nodes is at most $(n-1)\mathcal{T}$, the estimate has been increased by at least $(n-1)(1-\rho)\mathcal{D}$ by time t.

The estimate can reach every node within time $(n-1)(\mathcal{T}+\mathcal{D})$ for the following reason. The $(\mathcal{T}+\mathcal{D})$-interval connectivity implies that throughout every interval of length $\mathcal{T}+\mathcal{D}$, there is an edge over every cut in the network. In particular, there always is such an edge over the cut separating all nodes that have already received the estimate from all other nodes. Because this edge is discovered within $\mathcal{D}$ time units and because transmitting a message over the edge takes at most $\mathcal{T}$ time, the number of nodes that have received the estimate grows by at least 1 after every $\mathcal{D}+\mathcal{T}$ time units. □

Full Proof. All hardware clocks and max-estimates are initialized to 0 at time 0, and hence $L^{\max}(0) - L_u^{\max}(0) = 0$. The max clock $L^{\max}$ increases at a rate of no more than $1+\rho$, and the max-estimate $L_u^{\max}(t)$ of any node u increases at a rate of at least $1-\rho$. Consequently, the difference $L^{\max}(t) - L_u^{\max}(t)$ grows at a rate of no more than $(1+\rho)-(1-\rho) = 2\rho$, and because $\rho < 1$, the claim holds at least until time

$$t = \frac{(1+\rho)\mathcal{T} + 2\rho \cdot \mathcal{D}}{2\rho} \cdot (n-1) > (\mathcal{T}+\mathcal{D}) \cdot (n-1).$$

Thus, it is sufficient to consider times t such that $t > (\mathcal{T}+\mathcal{D}) \cdot (n-1)$.

For $i \in \{1, \ldots, n\}$, define

$$t_i := t - (n-i)(\mathcal{T}+\mathcal{D})$$

and

$$V_i := \{v \in V \mid L_v^{\max}(t_i) \geq L^{\max}(t_1) + (i-1)(1-\rho)\mathcal{D}\}\,.$$

We prove by induction on i that for all $i \in \{1, \ldots, n\}$ we have $|V_i| \geq i$.

- (Base) By definition, $V_1 = \{v \in V \mid L_v^{\max}(t_1) \geq L^{\max}(t_1)\}$. There exists some node v such that $L_v^{\max}(t_1) = L^{\max}(t_1)$, and consequently $|V_1| \geq 1$.

- (Step) Suppose that $|V_{i-1}| \geq i-1$. By definition, for all $v \in V_{i-1}$ we have

$$L_v^{\max}(t_{i-1}) \geq L^{\max}(t_1) + (i-2)(1-\rho)\mathcal{D}. \tag{16}$$

  The max estimate of each node increases at least at the rate of its hardware clock. Consequently, for all $v \in V_{i-1}$,

$$\begin{aligned} L_v^{\max}(t_i) &\geq L_v^{\max}(t_{i-1}) + (t_i - t_{i-1})(1-\rho) \geq && \text{(From (16))} \\ &\geq L^{\max}(t_1) + (i-2)(1-\rho)\mathcal{D} + (t_i - t_{i-1})(1-\rho) \geq \\ &\geq L^{\max}(t_1) + (i-1)(1-\rho)\mathcal{D}, \end{aligned}$$

  and hence $V_{i-1} \subseteq V_i$.

  If $V \setminus V_{i-1} = \emptyset$, then $|V_i| \geq |V_{i-1}| = n$ and we are done. Otherwise by $(\mathcal{T} + \mathcal{D})$-interval connectivity of $G(t)$ there exists an edge $e = \{v, w\}$, where $v \in V_{i-1}$ and $w \in V \setminus V_{i-1}$, such that e exists throughout the interval $[t_{i-1}, t_i]$. By Property 6.2, there are times $t_{\text{snd}} \geq t_{i-1}$ and $t_{\text{rcv}} \leq t_i$ such that node v sends node w a message containing $L_v^{\max}(t_{\text{snd}})$ at time t_{snd}, and node w receives the message at time t_{rcv} and updates its max estimate. Thus we have

$$\begin{aligned} L_w^{\max}(t_i) &\geq L_w^{\max}(t_{\text{rcv}}) + (1-\rho)(t_i - t_{\text{rcv}}) \geq \\ &\geq L_v^{\max}(t_{\text{snd}}) + (1-\rho)(t_i - t_{\text{rcv}}) \geq \\ &\geq L_v^{\max}(t_{i-1}) + (1-\rho)(t_i - t_{\text{rcv}}) + (1-\rho)(t_{\text{snd}} - t_{i-1}) \geq \\ &\geq L_v^{\max}(t_{i-1}) + (1-\rho)(t_i - t_{i-1} - \mathcal{T}) = \\ &= L_v^{\max}(t_{i-1}) + (1-\rho)\mathcal{D} \geq && \text{(From (16))} \\ &\geq L^{\max}(t_1) + (i-1)(1-\rho)\mathcal{D}. \end{aligned}$$

  It follows that $w \in V_i$. Since $w \notin V_{i-1}$ and $V_{i-1} \cup \{w\} \subseteq V_i$ we have $|V_i| \geq |V_{i-1}| + 1 \geq i$.

The claim we proved implies that $V_n = V$; that is, for all $v \in V$, at time $t_n = t$ we have

$$L_v^{\max}(t) \geq L^{\max}(t_1) + (n-1)(1-\rho)\mathcal{D}. \tag{17}$$

From Property 6.7,

$$L^{\max}(t) \leq L^{\max}(t_1) + (1+\rho)(t - t_1) = L^{\max}(t_1) + (1+\rho)(n-1)(\mathcal{T} + \mathcal{D}), \tag{18}$$

and combining (17) and (18) yields

$$L^{\max}(t) - L_v^{\max}(t) \leq (n-1)\left((1+\rho)\mathcal{T} + 2\rho \cdot \mathcal{D}\right).$$

□

Using the approach sketched above, Lemma 6.8 allows us to prove the following theorem, which bounds the global skew of our algorithm.

Theorem 6.9 (Global skew)**.** *The algorithm guarantees a global skew of*

$$\mathcal{G}(n) := ((1+\rho)\cdot\mathcal{T} + 2\rho\cdot\mathcal{D})\cdot(n-1).$$

Proof. We show the stronger statement that at all times t,

$$\forall v \in V \quad : \quad L^{\max}(t) - L_v(t) \leq \mathcal{G}(n)$$

and the claim then follows from Property 6.3 and the definition of $L^{\max}$.

For the sake of contradiction, assume that this is not the case. Then there is some time t, node $v \in V$ and $\varepsilon > 0$ such that

$$L^{\max}(t) - L_v(t) \geq \mathcal{G}(n) + \varepsilon \tag{19}$$

Let $\bar{t}$ be the infimum of times when (19) holds for some node v. By Lemma 6.8, we have $L^{\max}(\bar{t}) - L_v^{\max}(\bar{t}) \leq \mathcal{G}(n)$ and thus $L_v(\bar{t}) < L_v^{\max}(\bar{t})$. Hence, as a consequence of Lemma 6.4, v is blocked at time $\bar{t}$. Therefore by Definition 6.1, there is a node $u \in \Gamma_v(\bar{t})$ such that $L_v(\bar{t}) - L_v^u(\bar{t}) > B_v^u(\bar{t}) \geq B_0$. By Lemma 6.5, it therefore holds that $L_u(\bar{t}-\tau) < L_v(\bar{t}) - B_0$. By Property 6.7, we have $L^{\max}(\bar{t}-\tau) \geq L^{\max}(\bar{t}) - (1+\rho)\tau$. We therefore obtain

$$L^{\max}(\bar{t}-\tau) - L_u(\bar{t}-\tau) > L^{\max}(\bar{t}) - L_v(\bar{t}) - (1+\rho)\tau + B_0.$$

Because we assume that $B_0 \geq (1+\rho)\tau$, this is a contradiction to the assumption that $\bar{t}$ is the infimum of times when (19) is satisfied for the first time for some node v. □

6.3 Local Skew

The local skew guarantee of the algorithm hinges on the fact that the constraint imposed by a newly formed edge is so weak that *no* edge can violate it: for a long time after edge $\{u, v\}$ is detected, the value of B_u^v stays greater than the global skew $\mathcal{G}(n)$. Since no edge carries a skew that is greater than $\mathcal{G}(n)$, the requirement is trivially satisfied. In fact, only after $\Theta\left(\frac{\mathcal{G}(n)}{B_0}\right)$ time can a node be blocked by a new neighbor, and this is formalized by the following lemma.

Lemma 6.10. *If node v blocks node u at time t, then $v \in \Gamma_u(t')$ for all $t' \in [t-W, t]$, where*

$$W := \left(4\frac{\mathcal{G}(n)}{B_0} + 1\right)\tau.$$

(Informally, the interval $[t-W, t]$ corresponds to the time required according to Theorem 4.1 for information about the new edge to spread throughout the network.)

Proof. For node v to block node u at time t, we must have

$$L_u(t) - L_u^v(t) > B_u^v(t). \tag{20}$$

Using Lemma 6.5, Property 6.7 and the global skew guarantee (Thm. 6.9), we obtain

$$\begin{aligned} L_u(t) - L_u^v(t) \le L^{\max}(t) - L_v(t-\tau) \le L^{\max}(t-\tau) - L_v(t-\tau) + (1+\rho)\tau \le \\ \le \mathcal{G}(n) + (1+\rho)\tau. \end{aligned}$$

Substituting the definition of B_u^v yields

$$\begin{aligned} B_u^v(t) = \max\left\{B_0, 5\mathcal{G}(n) + (1+\rho)\tau + B_0 - \frac{B_0}{(1+\rho)\tau}\left(H_u(t) - C_u^v(t)\right)\right\} < \\ < L_u(t) - L_u^v(t) \le \mathcal{G}(n) + (1+\rho)\tau, \end{aligned}$$

and in particular,

$$5\mathcal{G}(n) + (1+\rho)\tau + B_0 - \frac{B_0}{(1+\rho)\tau}\left(H_u(t) - C_u^v(t)\right) < \mathcal{G}(n) + (1+\rho)\tau.$$

Rearranging we obtain

$$H_u(t) - C_u^v(t) > \left(4\frac{\mathcal{G}(n)}{B_0} + 1\right)(1+\rho)\tau. \tag{21}$$

Let $t_1 \le t$ be the most recent time node u adds node v to Γ_u (lines 17–19). Such a time must exist, because $v \in \Gamma_u(t)$. To prove the lemma it is sufficient to show that $t - t_1 \ge W$. Observe that by choice of t_1 we have $C_u^v(t) = H_u(t_1)$, and therefore, $H_u(t) - C_u^v(t) = H_u(t) - H_u(t_1) \le (1+\rho)(t - t_1)$. Using (21) we obtain

$$t - t_1 \ge \frac{H_u(t) - C_u^v(t)}{1+\rho} > \left(4\frac{\mathcal{G}(n)}{B_0} + 1\right)\tau.$$

□

Lemma 6.11 (Edge reversal)**.** *If node v blocks node u at time t then for all $t' \in [t-W+\Delta\mathcal{T}, t-\mathcal{D}]$ we have $L_v^{\max}(t') \ge L_u^{\max}(t'-\tau)$.*

Proof. From Lemma 6.10, if v blocks u at time t, then for all $t' \in [t-W, t]$ we have $v \in \Gamma_u(t')$. Since $\Gamma_u(t') \subseteq \Upsilon_u(t')$, this implies that $v \in \Upsilon_u$ throughout the interval. Hence, throughout the interval $[t-W, t]$, node u sends node v an update every $\frac{\Delta H}{1-\rho}$ real time units at most.

The model guarantees that if a message sent by u to v at time t' is not delivered, node u experiences a $\mathsf{discover}(\mathsf{remove}(\{u,v\}))$ event no later than time $t + \mathcal{D}$, which leads u to remove v from Γ_u (line 7). Since $v \in \Gamma_u$ throughout the interval $[t-W, t]$, all messages sent from u to v during the interval $[t-W, t-\mathcal{D}]$ are delivered. It follows that during the interval $[t-W+\Delta\mathcal{T}, t-\mathcal{D}]$, node v receives a message from u at least once every $\Delta\mathcal{T}$ time units, and hence throughout the interval we have $u \in \Gamma_v$. Property 6.1 implies that $L_v^{\max}(t') \ge L_u^{\max}(t'-\tau)$ for all $t' \in [t-W+\Delta\mathcal{T}, t-\mathcal{D}]$. □

The local skew guarantee of the algorithm is as follows.

Theorem 6.12. *For any two nodes u, v and time t such that $v \in \Gamma_u(t)$,*

$$L_u(t) - L_v(t) \le B_u^v(t - W) + 2\rho W = B_u^v(t - W) + 2\rho\tau \left(4\frac{\mathcal{G}(n)}{B_0} + 1\right)$$

Proof. Suppose by way of contradiction that at time t there are two nodes $u, v \in V$ such that $v \in \Gamma_u$ but

$$L_u(t) - L_v(t) > B_u^v(t - W) + 2\rho W.$$

Since B_u^v is non-increasing, for all $t' \in [t - W, t]$ we have

$$B_u^v(t') \le B_u^v(t - W). \tag{22}$$

From Lemma 6.6 and Lemma 6.10, at any time $t' \in [t - W, t]$ node u's logical clock cannot jump to a value that exceeds $L_v(t') + B_u^v(t') + 2\rho\tau \le L_v(t') + B_u^v(t - W) + 2\rho\tau$. Thus, the excess skew of $2\rho W - 2\rho\tau$ was built up by increasing u's logical clock at the rate of u's hardware clock, which is at most $1 + \rho$, while v's clock increased at a rate of at least $1 - \rho$. In other words, as long as the skew is greater than $B_u^v(t - W) + 2\rho\tau$ it increases at a rate of at most 2ρ, which implies that u's clock cannot make a discrete jump throughout the interval $[t - W + \tau, t]$.

Let $\ell = \lfloor \frac{W-\tau}{2\tau} \rfloor$. For all $0 \le i \le 2\ell$, define $t_i = t - i \cdot \tau$. In particular, $t_1 = t - \tau$, and

$$t_{2\ell} = t - 2\ell\tau \ge t - 2\tau \cdot \frac{W - \tau}{2\tau} = t - W + \tau.$$

It follows that for all $t' \in [t_{2\ell}, t]$,

$$L_u(t) - L_u(t') \le (1 + \rho)(t - t'), \tag{23}$$

since u cannot make a discrete change to its clock during this interval.

We show that there exists a chain of nodes $u_1, \ldots, u_\ell$ such that for all $1 \le i \le \ell$,

(C1) $L_u(t) - L_{u_i}(t_i) > i \cdot B_0$, and

(C2) For all $t' \in [t_{2\ell-i+1}, t_i]$ we have $L_{u_i}^{\max}(t') \ge L_u(t' - i\tau)$, and

(C3) Node u_i is blocked at time t_i.

The proof is by induction on i, the length of the chain.

- (Base.) For $i = 1$ we choose $u_1 = v$. We need to show that conditions (C1)–(C3) are satisfied.

  (C1) Since $t_1 = t - \tau < t$ and the logical clocks are strictly increasing we have $L_u(t) - L_v(t_1) > L_u(t) - L_v(t) > B_u^v(t) > B_0$.

(C2) Let $t' \in [t_{2\ell}, t_1]$. Since v blocks u at time t and $[t_{2\ell}, t_1] \subseteq [t - W + \Delta\mathcal{T}, t - \mathcal{D}]$, we can use Lemma 6.11 with t' to obtain

$$L_v^{\max}(t') \geq L_u(t' - \tau), \tag{24}$$

as required.

(C3) From Lemma 6.4, to show that v is blocked at time t_1 it suffices to show that $L_v^{\max}(t_1) > L_v(t_1)$. Applying (24) and (C1) with $t' = t_1 = t - \tau$ yields

$$\begin{aligned} L_v^{\max}(t_1) \geq L_u(t_1 - \tau) \geq& \\ &\overset{(23)}{\geq} L_u(t) - 2(1+\rho)\tau > L_v(t_1) + B_0 - 2(1+\rho)\tau > \\ &> L_v(t_1). \end{aligned}$$

- (Step.) Suppose that there is a node u_i such that

(IH1) $L_u(t) - L_{u_i}(t_i) > i \cdot B_0$, and

(IH2) For all $t' \in [t_{2\ell-i+1}, t_i]$ we have $L_{u_i}^{\max}(t') \geq L_u(t' - i\tau)$.

(IH3) Node u_i is blocked at time t_i,

Let u_{i+1} be a node that blocks u_i at time t_i. We show that conditions (C1)–(C3) are satisfied for u_{i+1}.

(C1) Since u_{i+1} blocks u_i at time t_i,

$$L_{u_i}(t_i) - L_{u_i}^{u_{i+1}}(t_i) > B_{u_i}^{u_{i+1}}(t_i) \geq B_0. \tag{25}$$

Using Lemma 6.5 we obtain

$$L_{u_i}^{u_{i+1}}(t_i) \geq L_{u_{i+1}}(t_i - \tau) = L_{u_{i+1}}(t_{i+1}), \tag{26}$$

and combining (25), (26) and (IH1) yields

$$\begin{aligned} L_u(t) - L_{u_{i+1}}(t_{i+1}) &\geq L_u(t) - L_{u_i}^{u_{i+1}}(t_i) > L_u(t) - L_{u_i}(t_i) + B_0 > \\ &\overset{\text{IH1}}{>} i \cdot B_0 + B_0 = (i+1)B_0. \end{aligned}$$

(C2) Let $t' \in [t_{2\ell-i}, t_{i+1}]$. Since $t_{2\ell-i} = t_{2\ell-i+1} + \tau$ and $t_{i+1} = t_i - \tau$, we have $t' - \tau \in [t_{2\ell-i+1}, t_i - 2\tau]$, and we can apply IH2 at time $t' - \tau$. In addition we have $t' \in [t_i - W + \Delta\mathcal{T}, t_i - \tau]$, and since u_{i+1} blocks u_i at time t_i, Lemma 6.11 shows that

$$L_{u_{i+1}}^{\max}(t') \geq L_{u_i}^{\max}(t' - \tau) \geq \tag{27}$$

$$\overset{\text{IH2}}{\geq} L_u(t' - \tau - i\tau) = L_u(t' - (i+1)\tau), \tag{28}$$

as required.

(C3) It remains to verify that node u_{i+1} is blocked at time t_{i+1}. Applying (28) with $t' = t_{i+1}$ yields

$$\begin{aligned} L_{u_{i+1}}^{\max}(t_{i+1}) &\geq L_u(t_{i+1} - (i+1)\tau) = L_u(t_{2(i+1)}) \geq \\ &\geq L_u(t) - (1+\rho)(t - t_{2(i+1)}) = L_u(t) - (1+\rho)(2i+1)\tau > \\ &> L_{u_{i+1}}(t_{i+1}) + (i+1)B_0 - (1+\rho)(2i+1)\tau \\ &\geq L_{u_{i+1}}(t_{i+1}). \end{aligned}$$

From Lemma 6.4, node u_{i+1} is blocked at time t_{i+1}.

This completes the induction.

The claim shows that there is a node u_ℓ such that

- Node u_ℓ is blocked at time t_ℓ, and
- $L_u(t) - L_{u_\ell}(t_\ell) > \ell \cdot B_0$.

Let $u_{\ell+1}$ be a node that blocks u_ℓ at time t_ℓ. As we already showed, it follows that $L_{u_\ell}(t_\ell) - L_{u_{\ell+1}}(t_{\ell+1}) > B_0$, which implies that

$$L_u(t) - L_{u_{\ell+1}}(t_{\ell+1}) > (\ell+1)B_0. \tag{29}$$

Since $t_{\ell+1} > t_{2(\ell+1)}$ we can use (23) to obtain

$$L_u(t_{\ell+1}) \geq L_u(t) - (1+\rho)(\ell+1)\tau, \tag{30}$$

and combining (29) and (30) yields

$$\begin{aligned} L_u(t_{\ell+1}) - L_{u_{\ell+1}}(t_{\ell+1}) &\geq L_u(t) - L_{u_{\ell+1}}(t_{\ell+1}) - (1+\rho)(\ell+1)\tau > \\ &> (\ell+1)B_0 - (1+\rho)(\ell+1)\tau \geq \\ &\geq (\ell+1)\left(B_0 - (1+\rho)\tau\right) \geq \qquad (B_0 \geq 2(1+\rho)\tau) \\ &\geq (\ell+1)\frac{B_0}{2} \geq \frac{W-\tau}{2\tau} \cdot \frac{B_0}{2} \geq \frac{2\mathcal{G}(n)}{B_0} \cdot \frac{B_0}{2} = \mathcal{G}(n), \end{aligned}$$

which is a contradiction to the global skew guaranteed by the algorithm.

□

Theorem 6.12 describes the local skew guarantee from a point of view that is subjective to node u: the statement of the theorem assumes that $v \in \Gamma_u$, and the value of B_u^v depends on the local variables C_u^v and H_u. The following corollary states the "objective" local skew guarantee of the algorithm.

Corollary 6.13. *The algorithm guarantees a dynamic local skew of*

$$s(n, I, \Delta t) = s(n, \Delta t) = B\left(\max\left\{(1-\rho)(\Delta t - \Delta\mathcal{T} - \mathcal{D} - W), 0\right\}\right) + 2\rho W,$$

regardless of the initial skew I on the edge.

Proof. Let $e = \{u, v\}$ be an edge that exists throughout an interval $[t, t+\Delta t]$. If $\Delta t - \Delta\mathcal{T} - \mathcal{D} - W \leq 0$, then $s(n, \Delta t) = B(0) + 2\rho W > \mathcal{G}(n)$, and all edges carry less than $s(n, \Delta t)$ skew. Suppose then that $\Delta t - \Delta\mathcal{T} - \mathcal{D} - W > 0$, that is, $t + \Delta t - W > t + \Delta\mathcal{T} + \mathcal{D}$.

Since the edge exists throughout the interval $[t, t + \Delta t]$, from Property 6.2, at any time $t' \in [t + \Delta\mathcal{T} + \mathcal{D}, t + \Delta t]$ we have $v \in \Gamma_u(t')$. Thus, the last time v was added to Γ_u prior to time $t + \Delta t$ is some time $t_1 \leq t + \Delta\mathcal{T} + \mathcal{D} < t + \Delta t - W$, and from the algorithm, $C_u^v(t + \Delta t - W) = H_u(t_1) \leq H_u(t + \Delta\mathcal{T} + \mathcal{D})$. Since B is non-increasing,

$$\begin{aligned} B_u^v(t + \Delta t - W) &= B\left(H_u(t + \Delta t - W) - C_u^v(t + \Delta t - W)\right) \leq \\ &\leq B\left(H_u(t + \Delta t - W) - H_u(t + \Delta\mathcal{T} + \mathcal{D})\right) \leq \\ &\leq B\left((1-\rho)(t + \Delta t - t - \Delta\mathcal{T} - \mathcal{D} - W)\right) = \\ &= B\left((1-\rho)(\Delta t - \Delta\mathcal{T} - \mathcal{D} - W)\right). \end{aligned}$$

Now we can use Theorem 6.12 to obtain

$$L_u(t + \Delta t) - L_v(t + \Delta t) \leq B_u^v(t + \Delta t - W) + 2\rho W \leq s(n, \Delta t),$$

and similarly we can show that $L_v(t + \Delta t) - L_u(t + \Delta t) \leq s(n, \Delta t)$ as well. Together we have $|L_u(t + \Delta t) - L_v(t + \Delta)| \leq s(n, \Delta t)$, as required. □

Corollary 6.14. *If the parameter B_0 is chosen as $B_0 \geq \lambda\sqrt{\rho n}$ for a constant $\lambda > 0$, the stable local skew of the algorithm is $O(B_0)$. Further, the time to reach this stable skew on a new edge is $O(n/B_0)$. Hence, for this choice of B_0, the trade-off achieved by the algorithm asymptotically matches the trade-off given by the lower bound in Theorem 4.1.*

7 Conclusion

We have established fundamental trade-offs for gradient clock synchronization algorithms in dynamic networks. First, the time to adjust the skew on a newly formed edge is inversely proportional to the skew one is willing to tolerate on well-established edges. Hence, having a stronger skew requirement in stable conditions impairs the ability to adapt to dynamic changes. Second, contrary to what one might initially think, reducing the skew on edges with a small initial skew turns out to be as hard as reducing the skew on edges with a large initial skew. The time needed in both cases is linear in the global skew bound of the algorithm and is thus at least linear in n.

It will be interesting to see whether the trade-off established by our algorithm can also be achieved for smaller stable skew bounds. In particular, it will be interesting to see whether the techniques developed in [11, 12] to guarantee a local skew of $O(\log n)$ in the static case can be adapted for the dynamic setting. Note, however, that such an improved local skew bound necessarily comes at the cost of worse adaptability to topology changes.

In this paper we used a weighted-graph approach to deal with the dynamic topology: in the algorithm of Section 5, each edge carries a weight, which starts out very large when the edge first appears and decreases over time. We use the dynamic weights to gradually decrease the effective diameter of the graph, giving nodes time to adapt to the appearance of new edges. In a companion paper [9] we use a similar approach to incorporate reference broadcast synchronization in the algorithm from [12]. In that case the weight of the edge has the traditional meaning in the context of clock synchronization: it corresponds to the uncertainty along the edge. It is our hope that extending the algorithm from [12] to the weighted-graph model will serve as a first step towards a dynamic clock synchronization algorithm with $O(\log n)$ stable local skew, but this seems challenging.

An additional obvious generalization would be to incorporate node insertions and deletions in the dynamic graph model. As long as nodes join and leave at a constant rate, it might be possible to be able to adapt all the parameters used sufficiently quickly in order to still guarantee the same basic results. The details of such a protocol as well as possible limitations on how fast one can adapt to changes of the network size are fascinating open questions.

References

[1] H. Attiya, D. Hay, and J.L. Welch. Optimal clock synchronization under energy constraints in wireless ad-hoc networks. In *Proc. of 9th Int. Conf. on Principles of Distributed Systems (OPODIS)*, pages 221–234, 2005.

[2] H. Attiya, A. Herzberg, and S. Rajsbaum. Optimal clock synchronization under different delay assumptions. *SIAM Journal on Computing*, 25(2):369–389, 1996.

[3] S. Biaz and J.L. Welch. Closed form bounds for clock synchronization under simple uncertainty assumptions. *Information Processing Letters*, 80(3):151–157, 2001.

[4] D. Dolev, J.Y. Halpern, B. Simons, and R. Strong. Dynamic fault-tolerant clock synchronization. *Journal of the ACM*, 42(1):143–185, 1995.

[5] D. Dolev, J.Y. Halpern, and H.R. Strong. On the possibility and impossibility of achieving clock synchronization. *Journal of Computer and System Sciences*, 32(2):230–250, 1986.

[6] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Operating Systems Review*, 36(SI):147–163, 2002.

[7] R. Fan, I. Chakraborty, and N. Lynch. Clock synchronization for wireless networks. In *Proc of 8th Int. Conf. on Principles of Distributed Systems (OPODIS)*, pages 400–414, 2004.

[8] R. Fan and N. Lynch. Gradient clock synchronization. *Distributed Computing*, 18(4):255–266, 2006.

[9] F. Kuhn and R. Oshman. Gradient clock synchronization using reference broadcasts. arXiv:0905.3454v1, 2009. `http://arxiv.org/abs/0905.3454v1`.

[10] L. Lamport and P.M. Melliar-Smith. Synchronizing clocks in the presence of faults. *Journal of the ACM*, 32(1):52–78, 1985.

[11] C. Lenzen, T. Locher, and R. Wattenhofer. Clock synchronization with bounded global and local skew. In *Prof. of 49th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 500–510, 2008.

[12] C. Lenzen, T. Locher, and R. Wattenhofer. Tight bounds for clock synchronization. In *Proc. of 28th ACM Symp. on Principles of Distributed Computing (PODC)*, 2009.

[13] T. Locher and R. Wattenhofer. Oblivious gradient clock synchronization. In *Proc. of 20th Int. Symp. on Distributed Computing (DISC)*, pages 520–533, 2006.

[14] J. Lundelius and N. Lynch. An upper and lower bound for clock synchronization. *Information and Control*, 62(2/3):190–204, 1984.

[15] K. Marzullo and S. Owicki. Maintining the time in a distributed system. In *Proc. of 2nd ACM Symp. on Principles of Distributed Computing (PODC)*, pages 44–54, 1983.

[16] R. Ostrovsky and B. Patt-Shamir. Optimal and efficient clock synchronization under drifting clocks. In *Proc. of 18th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 400–414, 1999.

[17] B. Patt-Shamir and S. Rajsbaum. A theory of clock synchronization. In *Proc. of 26th ACM Symp. on Theory of Computing (STOC)*, pages 810–819, 1994.

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