Gradient Clock Synchronization Using Reference Broadcasts

Fabian Kuhn[1] and Rotem Oshman[2]

[1] Faculty of Informatics, University of Lugano, Switzerland
[2] Computer Science and Artificial Intelligence Laboratory, MIT, USA

Abstract. Reference-Broadcast Synchronization (RBS) is a technique that allows a set of receivers in a broadcast network to accurately estimate each others' clock values. RBS provides a relative time-frame for conversion between the local clocks of different nodes, and can be used to synchronize nodes to an external time-source such as GPS. However, RBS by itself does not output a logical clock at every node, and so it does not solve internal clock synchronization.

In this work we study the theoretical properties of RBS in the worst-case model, in which the performance of a clock synchronization algorithm is measured by the worst-case skew it can incur. We suggest a method by which RBS can be incorporated in standard internal clock synchronization algorithms. This is achieved by separating the task of estimating the clock values of other nodes in the network from the task of using these estimates to output a logical clock value.

The separation is modelled using a virtual *estimate graph*, overlaid on top of the real network graph, which represents the information various nodes can obtain about each other. RBS estimates are represented in the estimate graph as edges between nodes at distance 2 from each other in the original network graph. A clock synchronization algorithm then operates on the estimate graph as though it were the original network.

To illustrate the merits of this approach, we modify a recent optimal gradient clock synchronization algorithm to work in this setting. The modified algorithm transparently takes advantage of RBS estimates. Its quality of synchronization depends on the diameter of the estimate graph, which is typically much smaller than the diameter of the original network graph.

Keywords: Gradient Clock Synchronization, Wireless Networks.

1 Introduction

The evolving field of wireless networks poses new and interesting challenges to time synchronization, leading to renewed attention to this venerable problem in recent years. Sensor networks in particular are subject to constraints on computation power and energy consumption, and often require a greater degree of synchronization than traditional distributed applications.

In a multi-hop sensor network it is frequently the case that neighboring nodes must be closely synchronized, while far-apart nodes can tolerate greater clock

T. Abdelzaher, M. Raynal, and N. Santoro (Eds.): OPODIS 2009, LNCS 5923, pp. 204–218, 2009.

skew: neighboring nodes interfere with each other when they try to transmit, and are also more likely to cooperate for the purpose of some local computation. This gives rise to the problem of *gradient clock synchronization*, in which the synchronization between two nodes improves the closer they are to each other. The problem was first formulated in [6], where it is shown that in a network of diameter D, no algorithm can guarantee a skew that is better than $\Omega(\log D/\log\log D)$ even between adjacent nodes. Subsequent work has improved the lower bound to $\Omega(\log D)$, and come up with algorithms that match it [9,10].

The wireless broadcast medium also offers opportunities for better synchronization. Although contention may cause unpredictable delays before a message is broadcast, once a message is transmitted, it is received by all nodes in the sender's neighborhood almost instantaneously. Reference broadcast synchronization (RBS) [4] takes advantage of this to let the *neighbors* of the sender estimate each other's clock values with great accuracy. RBS can be extended to multi-hop networks, to allow any node in the network to estimate the clock value of any other node. However, by itself, RBS does not output a *logical clock* at every node, and so it is not a clock synchronization algorithm in the traditional sense.

In this paper we suggest an approach by which RBS, or any other estimation method (including external time sources), can be seamlessly incorporated in many clock synchronization algorithms, in order to reduce the effective diameter of the network and achieve better synchronization. We suggest a separation between the *estimate layer*, which is responsible for estimating other nodes' clock values, and the algorithm that uses these estimates to compute a local logical clock. The estimate layer runs underneath the algorithm and provides it with an *estimate graph* G^{est}. Each edge $\{u, v\}$ of G^{est} represents an estimate that node u can get for node v's clock value (and vice-versa), along with an associated *uncertainty*. RBS estimates are represented in G^{est} as edges between nodes at distance 2 from each other in the original network graph.

Almost any clock synchronization algorithm can be used on top of the estimate layer, as long as the algorithm can handle networks with non-uniform uncertainty on the links. The resulting synchronization between nodes u, v depends on their *effective distance* $\text{dist}(u, v)$, and on the *effective diameter* of the network graph. These are defined by the corresponding distances in the estimate graph G^{est}. Using RBS it is possible to reduce the effective diameter to $O((\rho\cdot\mathcal{T}+u_{\text{rcv}})\cdot D+\mathcal{T})$, where D is the diameter of the original network, $\mathcal{T}$ is a bound on the message delay, ρ is a bound on clock drift (typically very small), and u_{rcv} is a bound on the receiver uncertainty (also very small [4]), which bounds the time it takes a node to process a message it receives.

Our main contributions are as follows. In Section 4 we define the estimate layer, and show how to incorporate point-to-point messages and RBS. In Section 5, we illustrate the applicability of our approach by modifying the algorithm of [10] to work on top of the estimate layer. Significantly, this involves extending it to a heterogeneous network; in [10] it is assumed that all links are subject to the same bounds on message delay. Finally, in Section 6 we prove that the algorithm achieves gradient clock synchronization, with the skew between nodes

u and v bounded by $O(\text{dist}(u,v) \cdot \log_{1/\rho} \mathcal{D})$ in networks with effective diameter $\mathcal{D}$ and drift bounded by ρ. This is asymptotically optimal. The proof is based on the proof in [10], but in our view it is cleaner and somewhat simpler.

2 Related Work

The problem of establishing a common notion of time is at the core of many distributed systems and applications and has been widely studied, from both theoretical and a practical points of view. In most of the existing work on clock synchronization, the nodes of a network compute estimates about each others' clock values by exchanging messages. Based on the information obtained, each node computes a local logical clock. Typically, the accuracy of clock estimates is determined by the uncertainty about the propagation delay of messages. In [12], it is shown that even if hardware clocks experience no drift, no clock synchronization algorithm can prevent a clock skew of $\Omega(D)$ in a network of diameter D. This lower bound on the maximum clock skew between any two nodes is matched by an algorithm described in [20] and by many subsequent algorithms (e.g. [2,14,5,10,9,15,16]). Clock synchronization algorithms and lower bounds that accommodate non-uniform uncertainties are described, for example, in [1,3,7].

In [6], Fan and Lynch introduced the gradient clock synchronization problem. It is shown that even on a path of length D, no algorithm can guarantee a clock skew smaller than $\Omega(\log D / \log\log D)$ between adjacent nodes. This bound has been improved to $\Omega(\log D)$ in [10] and it is shown in [9,10] that the new bound in indeed tight.

The special properties, constraints, and requirements of wireless ad hoc and sensor networks make clock synchronization especially challenging. There is a considerable amount of work on the problem (e.g. [5,18,19,21,13,17]). Particularly interesting is the work on reference broadcast synchronization [4,8], which exploits the property of sharing a single communication channel to obtain high accuracy clock estimates of nearby nodes.

3 Preliminaries

In the sequel we use $\mathbb{R}^{\geq 0}$ to denote the set of non-negative reals and $\mathbb{N}^{>0}$ to denote the positive integers.

We model a wireless network as an undirected graph $G = (V, E)$, where V is the set of nodes, and $\{u, v\} \in E$ iff u is in reception range of v and vice-versa. We abstract away low-level details of contention management, message loss and so on, by assuming reliable message delivery with message delays bounded by a parameter $\mathcal{T}$.

Each node v in the network has access to a local hardware clock H_v, which is subject to drift bounded by $\rho < 1$. We assume that for all $t_1 \leq t_2$,

$$(1-\rho)(t_2 - t_1) \leq H_v(t_2) - H_v(t_1) \leq (1+\rho)(t_2 - t_1).$$

The hardware clock increases continuously, and for the analysis we assume it is differentiable.

The goal of gradient clock synchronization is to output a local logical clock L_v at every node v, which is closely-synchronized with all the other logical clocks. Formally, an algorithm is said to achieve f-gradient clock synchronization, for a function $f : \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$, if it satisfies the following requirement.

Requirement 31. *For all $u, v \in V$ and times t we have*

$$L_v(t) - L_u(t) \leq f\left(\text{dist}(u, v)\right).$$

Here $\text{dist}(u, v)$ stands for the distance between u and v, which informally corresponds to the accuracy of information u and v can acquire about each other. Traditionally, $\text{dist}(u, v)$ is defined as the minimal sum of the uncertainties regarding message delay on any path between u and v (see, e.g., [2]). In the next section we redefine $\text{dist}(u, v)$ to incorporate reference broadcast synchronization.

In addition to f-gradient synchronization, we require the logical clocks to behave like "real" clocks. Specifically, the logical clocks should be strictly increasing, and they should always be within a linear envelope of real time. In particular, the logical clocks are continuous. This is captured by the following requirement.

Requirement 32. *There exist $\alpha \in (0, 1)$ and $\beta \geq 0$ such that for all $t_1 \leq t_2$,*

$$(1 - \alpha)(t_2 - t_1) \leq L_u(t_2) - L_u(t_1) \leq (1 + \beta)(t_2 - t_1).$$

4 The Estimate Layer

The estimate layer encapsulates point-to-point messages, reference broadcast synchronization, and any other means the nodes in the network have of obtaining information about the logical clock values of other nodes. The estimate layer provides an undirected *estimate graph* $G^{\text{est}} = (V, E^{\text{est}})$, where each edge $u, v \in E^{\text{est}}$ represents some method by which nodes u and v can estimate each others' logical clock values. Note that G^{est} can be different from the underlying network graph G; for example, RBS is represented in G^{est} as edges connecting nodes at distance 2 from each other in G. We use $N(u) := \{v \in V \mid u, v \in E^{\text{est}}\}$ to denote u's neighborhood in G^{est}.

The estimate layer provides each node $u \in V$ with a set of local variables $\left\{\tilde{L}_u^v : v \in N(u)\right\}$, which represent u's current estimates for the logical clock values of its neighbors in G^{est}. Since the estimates are typically inaccurate, we associate with every edge $e \in E^{\text{est}}$ an *uncertainty* ϵ_e. The estimate layer guarantees the following property.

Property 1 (Estimate quality). For any edge $(u, v) \in E^{\text{est}}$ and time t, we have

$$L_v(t) - \epsilon_{\{u,v\}} \leq \tilde{L}_u^v(t) \leq L_v(t) + \epsilon_{\{u,v\}}.$$

Two methods of obtaining logical clock estimates are described below. We describe each method and bound the error associated with it, and then show how to combine multiple methods.

Direct estimates. Following the style of algorithms suggested in [11,9,10], we assume that every node broadcasts its logical clock value to all its neighbors once every subjective ΔH time units (that is, after its hardware clock has increased by ΔH), where ΔH is a parameter. These messages provide a direct estimate of the node's logical clock value. When node u receives a message from v at time t, it sets $\tilde{L}_u^{v,\text{direct}} \leftarrow L$. Between messages from v, node u increases $\tilde{L}_u^{v,\text{direct}}$ at the rate of its own hardware clock.

The error of a direct estimate can be shown to be bounded by

$$-(\alpha+\rho)\left(\frac{\Delta H}{1-\rho}+\mathcal{T}\right) \le L_v(t) - \tilde{L}_u^{v,\text{direct}}(t) \le (\beta+\rho)\left(\frac{\Delta H}{1-\rho}+\mathcal{T}\right) + (1-\rho)\mathcal{T}.$$

Note that at this point, our error bound is asymmetric. It is straightforward to obtain a symmetric guarantee in the style of Prop. 1. Specifically, if $\beta = O(1)$, we have $\left|L_v(t) - \tilde{L}_u^{v,\text{direct}}(t)\right| = O(\Delta H + \mathcal{T})$.

RBS estimates. An RBS estimate is obtained by comparing the logical clock values that various nodes record when some common event occurs; in our case, a broadcast by a shared neighbor. We give a simple way to obtain RBS estimates, which is optimal as regards worst-case analysis, but differs from the more practical treatment in [4].

We use $\mathcal{H}_u$ to denote node u's *history*, a set of triplets (x, L, H) where x is a unique event identifier and L, H record node u's logical and hardware clock values when it observed the event. After recording event x, node u sends a $\mathsf{report}(u, x, L)$ message, which is propagated until it reaches all other nodes that observed the same event. In our case, $\mathsf{report}(\cdot)$ messages need to be re-broadcast only once, so that they reach the 2-neighborhood of the node that originated the report.

The accuracy of RBS depends on two factors.

1. *Receiver uncertainty* is the time required for nodes to process the common event and record their logical clock value. The receiver uncertainty is bounded by u_{rcv} if whenever an event x occurs at real time t, there is some $t_x \in [t, t+u_{\text{rcv}}]$ such that for all $t' \ge t_x$ we have $(x, L_u(t_x), H_u(t_x)) \in \mathcal{H}_u(t')$.
2. *Propagation delay* is the time it takes for nodes that observe an event to receive $\mathsf{report}(\cdot)$ messages from other nodes that observed it. This delay contributes to the inaccuracy of the estimate, because while the report is propagated the clocks continue to drift apart. We say that the propagation delay is bounded by $\mathcal{P}$ if whenever a node u experiences an event x at real time t, every node $v \in N^2(u)$ receives a $\mathsf{report}(u, x, L)$ message no later than time $t + \mathcal{P}$.
   In our case, because $\mathsf{report}(\cdot)$ messages need to be re-broadcast only once, the propagation delay is bounded by $\mathcal{P} \le u_{\text{rcv}} + 2\left(\frac{\Delta H}{1-\rho} + \mathcal{T}\right)$: after observing the event, node u waits at most $\frac{\Delta H}{1-\rho}$ time units and then broadcasts the message, which takes at most $\mathcal{T}$ time units to arrive; its neighbors do the same.

When node u receives a $\mathsf{report}(v, x, L)$ message at time t, it looks up the corresponding triplet (x, H', L') recorded in its own history. It uses $H_u - H'$ to estimate the time that has passed since x occurred, and sets

$$\tilde{L}_u^{v,\mathrm{rbs}} \leftarrow L + H_u - H'.$$

Every broadcast by a node is an event that its neighbors use to get estimates of each others' logical clock values. RBS estimates are accurate up to the following bound.

$$-(\alpha+\rho)\left(\frac{\Delta H}{1-\rho}+\mathcal{P}\right) - (1-\alpha)u_{\mathrm{rcv}} \le L_v(t) - \tilde{L}_u^{v,\mathrm{rbs}}(t) \le \\ \le (\beta+\rho)\left(\frac{\Delta H}{1-\rho}+\mathcal{P}\right) + (1-\rho)u_{\mathrm{rcv}}.$$

Assuming $u_{\mathrm{rcv}} \ll \Delta H + \mathcal{T}$, the RBS estimates are a significant improvement over direct estimates between nodes at distance 2 as long as the clock synchronization algorithm guarantees that $\alpha, \beta \ll 1$. In particular, if $\alpha, \beta = O(\rho)$, we obtain $\left|L_v(t) - \tilde{L}_u^{v,\mathrm{rbs}}(t)\right| = O(u_{\mathrm{rcv}} + \rho(\Delta H + \mathcal{T}))$.

Combining multiple estimates. As we have seen, each node may have multiple ways of estimating the clock values of its neighbors in G^{est}. Let $\tilde{L}_u^{v,1}, \ldots, \tilde{L}_u^{v,m}$ be the various estimates that u has for v's logical clock value, and let $\epsilon_{\mathrm{low}}^1, \ldots, \epsilon_{\mathrm{low}}^m$ and $\epsilon_{\mathrm{high}}^1, \ldots, \epsilon_{\mathrm{high}}^m$ be error bounds such that for all $i \in \{1, \ldots, m\}$ and time t we have $-\epsilon_{\mathrm{low}}^i \le L_v(t) - \tilde{L}_u^{v,i}(t) \le \epsilon_{\mathrm{high}}^i$. Node u computes a combined estimate with symmetric error, given by

$$\tilde{L}_u^v(t) := \frac{\min_i\left(\tilde{L}_u^{v,i}(t) + \epsilon_{\mathrm{high}}^i\right) - \max_i\left(\tilde{L}_u^{v,i}(t) - \epsilon_{\mathrm{low}}^i\right)}{2}. \tag{1}$$

The uncertainty of the combined estimate is bounded by

$$\epsilon_{\{u,v\}} := \min_i\left\{\frac{\epsilon_{\mathrm{low}}^i + \epsilon_{\mathrm{high}}^i}{2}\right\}.$$

Effective distance and diameter. Let $\mathcal{P}$ denote the set of all paths in the graph G^{est} (including non-simple paths), and let $\mathcal{P}(v) \subseteq \mathcal{P}$ denote the set of paths that start at node v. Given a path $P = v_0, \ldots, v_k \in \mathcal{P}$, we denote $\epsilon_P := \sum_{i=0}^{k-1} \epsilon_{\{v_i, v_{i+1}\}}$. Given two nodes $u, v \in V$, the distance between u and v is defined by

$$\mathrm{dist}(u,v) := \min_{P=u,\ldots,v} \epsilon_P, \tag{2}$$

and the diameter of the graph G^{est} is defined by

$$\mathcal{D} := \max_{u,v} \mathrm{dist}(u,v). \tag{3}$$

In the following assume that we have a clock synchronization algorithm that guarantees $\alpha, \beta = O(\rho)$. Let $d_e(u,v)$ be the length of the shortest even-length path and let $d(u,v)$ be the length of the shortest path between two nodes u and v in G. When using RBS estimates, we have $\mathrm{dist}(u,v) = O\big(d_e(u,v) \cdot (\rho(\Delta H + \mathcal{T}) +$

$u_{\text{rcv}})\big)$. Further, when using both direct and RBS estimates, we have $\text{dist}(u,v) = O\big(d(u,v)\cdot(\rho(\Delta H+\mathcal{T})+u_{\text{rcv}})+\Delta H+\mathcal{T}\big)$. Consequently, we obtain

$$\mathcal{D} = O\big((1+\rho D)(\Delta H+\mathcal{T})+u_{\text{rcv}}D\big),$$

where D is the diameter of the underlying network G. As the receiver uncertainty u_{rcv} is typically small, this is a significant improvement over the "true" diameter of G.

5 An Optimal Gradient Clock-Synchronization Algorithm

In this section we modify the algorithm of [10] to work on top of the estimation layer presented in the previous section.

To satisfy Requirement 32, the algorithm increases the logical clock in a continuous manner, with no discrete jumps. At each point during the execution a node is either in *fast mode* or in *slow mode*. In slow mode, u increases its logical clock at a rate of $\frac{d}{dt}H_u(t)$; in fast mode, the logical clock rate is $(1+\mu)\frac{d}{dt}H_u(t)$, where μ is a parameter.

Each node continually examines its estimates for the logical clock values of its neighbors in G^{est}. To compensate for the uncertainty on edge e we use a parameter κ_e, which is defined as

$$\kappa_e := \frac{2}{\lambda}\cdot\epsilon_e \tag{4}$$

for some constant $0<\lambda<1/4$ [1]. For a path $P\in\mathcal{P}$, we define $\kappa_P := \frac{2}{\lambda}\cdot\epsilon_P$.

If a node u finds that it is too far behind, it goes into fast mode and uses the fast rate of $(1+\mu)\frac{d}{dt}H_u(t)$. The following rule is used to determine when to go into fast mode; informally, it states that some neighbor is far ahead, and no neighbor is too far behind.

Definition 1 (Fast condition FC). *At time t, a node $u\in V$ satisfies the* fast condition*, denoted* FC*, if there is some integer $s\in\mathbb{N}$ for which following conditions are satisfied:*

(FC1) $\exists v\in N(u): \tilde{L}_u^v(t) - L_u(t) \geq (s-1-\lambda)\,\kappa_{\{u,v\}}$, *and*
(FC2) $\forall v\in N(u): L_u(t) - \tilde{L}_u^v(t) \geq (s-1+\lambda)\,\kappa_{\{u,v\}}$.

Conversely, if a node is far behind some neighbor, and no other neighbor is too far ahead of it, it enters slow mode and uses the slow rate. The rule for entering slow mode is as follows.

Definition 2 (Slow condition SC). *At time t, a node $u\in V$ satisfies the* slow condition*, denoted* SC*, if there is an integer $s\in\mathbb{N}^{>0}$ for which the following conditions are satisfied:*

(SC1) $\exists v\in N(u): L_u(t) - \tilde{L}_u^v(t) \geq \big(s-\frac{1}{2}-\lambda\big)\cdot\kappa_{\{u,v\}}$, *and*
(SC2) $\forall v\in N(u): \tilde{L}_u^v(t) - L_u(t) \leq \big(s-\frac{1}{2}+\lambda\big)\cdot\kappa_{\{u,v\}}$.

[1] The choice of 2 in the definition of κ_e is arbitrary; it is sufficient to have $\kappa_e > \frac{1}{\lambda}\cdot\epsilon_e$.

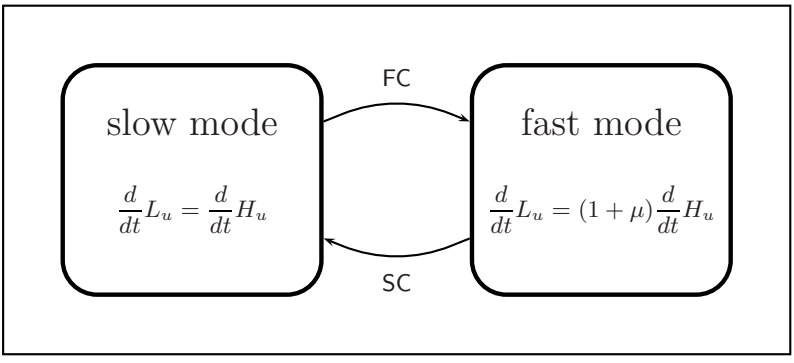

Fig. 1. A possible concrete implementation of the algorithm

The specification of the algorithm is nondeterministic. Whenever SC or FC are satisfied for some node, that node must be in slow or fast mode, respectively; this part of the specification is deterministic. However, when neither SC nor FC are satisfied, the node's behavior is nondeterministic, and the node can be in either slow or fast mode.

To show that the algorithm is realizable, we show that the two conditions are disjoint, which ensures that no node is required to be in both fast mode and slow mode at the same time. The proof is technical and we omit it here.

Lemma 1. *No node can satisfy* SC *and* FC *at the same time.* □

One possible implementation of the algorithm is shown in Fig. 1. In this implementation, the nondeterminism in the specification is resolved by having a node stay in its current state until SC or FC are satisfied, and then transition to the appropriate state. When neither SC nor FC are satisfied the node simply stays in its current state. We stress that this is only one possible choice; the algorithm performs correctly regardless of what nodes do when SC and FC are not satisfied.

6 Analysis

In this section we show that the algorithm achieves $O\left(\mathrm{dist}(u,v)\cdot\log\mathcal{D}\right)$-gradient synchronization. The proofs of some lemmas are omitted; they appear in the full version of this paper.

We define a parameter $\sigma \geq 2$, which serves as the base for the logarithm in the gradient skew bound. The correctness of the algorithm relies on the following assumption, which (informally) states that μ is large enough to allow nodes that are behind to catch up.

Property 2 (Requirement on μ). We require

$$\mu > 4\sigma\frac{\rho}{1-\rho}. \tag{5}$$

We show that the following invariant, which we denote $\boldsymbol{\mathcal{L}}$, is maintained throughout any execution of the algorithm.

Definition 3 (Legal State). *We say that the network is in a* legal state *at time t if and only if for all $s \in \mathbb{N}^{>0}$ and all paths $P = v_0, \ldots, v_k$, if*

$$\kappa_P(t) \geq C_s := \frac{4}{\lambda} \cdot \frac{\mathcal{D}}{\sigma^s},$$

then

$$L_{v_k}(t) - L_{v_0}(t) \leq s \cdot \kappa_P.$$

In particular, if the network is legal at time t, then for every two nodes u, v and integer $s \geq 1$ such that $\mathrm{dist}(u,v) \geq C_s$, we have $L_u(t) - L_v(t) \leq s \cdot \frac{2}{\lambda} \cdot \mathrm{dist}(u,v)$. The gradient synchronization property follows (see Corollaries 1, 2).

To show that the network is always in the safety region defined by the legal state condition, we show that whenever some path comes close to having illegal skew, the algorithm acts to decrease the skew, pulling the system back into the safety region.

Unfortunately, the proof is not straightforward. We cannot guarantee that a node will always "realize" when it is on a path that has too much skew: each node only has knowledge of its local neighborhood, and this local image may not reflect a large skew further down the path. We can, however, show that when the skew is close to being illegal, the nodes that are "the most behind" or "the most ahead", in a sense defined formally below, *will* realize that they must act to correct the skew. We will show that such nodes enter fast or slow mode as appropriate.

Since we can only argue about the clock rate of nodes that roughly speaking maximize some notion of weighted skew (defined below), we will use the following technical lemma.

Lemma 2. *Let $g_1, \ldots, g_n : \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$ be differentiable functions, and let $[a,b]$ be an interval such that for all $i \in \{1, \ldots, n\}$ and $x \in (a,b)$, if $g_i(x) = \max_j g_j(x)$ then $\frac{d}{dx} g_i(x) \leq r$. Then for all $x \in [a,b]$, $\max_i g_i(x) \leq \max_i g_i(a) + r \cdot (x - a)$.*

Next we define two different notions of "weighted skew": one captures how much a node v_0 is ahead of any other node, and the other captures how far behind it is. The weights in both cases are proportional to the uncertainty on the path, but use different constants. These notions correspond exactly to the the fast and slow conditions, respectively.

Definition 4. *Given an integer $s \in \mathbb{N}$, a time t, and a path $P = v_0, \ldots, v_k \in \mathcal{P}$, we define*

$$\Xi^s_P(t) := L_{v_0}(t) - L_{v_k}(t) - (s-1) \cdot \kappa_P, \quad \text{and} \quad \Xi^s_{v_0}(t) := \max_{P \in \mathcal{P}(v_0)} \Xi^s_P(t).$$

Definition 5. *Given an integer $s \in \mathbb{N}$, a time t, and a path $P = v_0, \ldots, v_k \in \mathcal{P}$, we define*

$$\Psi^s_P(t) := L_{v_k}(t) - L_{v_0}(t) - \left(s - \frac{1}{2}\right) \cdot \kappa_P, \quad \text{and} \quad \Psi^s_{v_0}(t) := \max_{P \in \mathcal{P}(v_0)} \Psi^s_P(t).$$

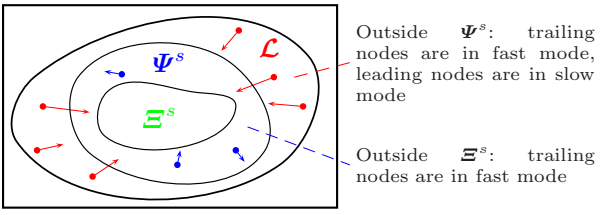

Fig. 2. Regions Ξ^s, Ψ^s and $\mathcal{L}$. Arrows illustrate the possible dynamics acting on the weighted skew in each region.

These definitions induce "inner safety regions" $\Xi^s := [\max_v \Xi_v^s \leq 0]$ and $\Psi^s := [\max_v \Psi_v^s \leq 0]$ for any $s \in \mathbb{N}^{>0}$, with $\Xi^s \subseteq \Psi^s \subseteq \mathcal{L}$ (see Fig. 2).

The next lemma can be thought of as bounding how far the system can stray outside the boundary of Ξ^s and Ψ^s while still being in a legal state.

Lemma 3. *If the network is in a legal state at time t, then for all nodes $u \in V$ and integers $s \geq 1$ we have $\Xi_u^s(t) < C_{s-1}$ and $\Psi_u^s(t) < C_{s-1}$.* □

Next we show that when the system is outside the region Ξ^s, nodes that are "the most behind" (maximize Ξ with respect to some other node) will be acting to catch up, and when the system is outside the region Ψ^s, nodes that are "the most ahead" will be held back from moving too quickly.

Lemma 4. *Let $P = v_0, \ldots, v_k \in \mathcal{P}(v_0)$ be a path starting at v_0 for which $\Xi_P^s(t) = \Xi_{v_0}^s(t)$ at some time t. If $\Xi_{v_0}^s(t) > 0$, then v_k is in fast mode at time t.*

Lemma 5. *Let $P = v_0, \ldots, v_k \in \mathcal{P}(v_0)(t)$ be a path starting at v_0 for which $\Psi_P^s(t) = \Psi_{v_0}^s(t)$ at some time t. If $\Psi_{v_0}^s(t) > 0$, then v_k is in slow mode at time t.*

The proofs of the two lemmas are similar. We give the proof of Lemma 4 here.

Proof (Lemma 4). We set out to show that v_k satisfies FC.

Consider any path $P' = v_0, \ldots, v \in \mathcal{P}(v_0)$ that ends at a neighbor v of v_k. Since $\Xi_P^s(t) = \Xi_{v_0}^s(t) = \max_{Q \in \mathcal{P}(v_0)} \Xi_Q^s(t)$, we have $\Xi_{P'}^s(t) \leq \Xi_P^s(t)$; that is,

$$L_{v_0}(t) - L_v(t) - (s-1) \cdot \kappa_{P'} \leq L_{v_0}(t) - L_{v_k}(t) - (s-1) \cdot \kappa_P.$$

Re-arranging yields

$$L_v(t) - L_{v_k}(t) \geq (s-1) \cdot (\kappa_P - \kappa_{P'}),$$

and applying Property 1 we obtain

$$\begin{aligned} \tilde{L}_{v_k}^v(t) - L_{v_k}(t) &\geq L_v(t) - \epsilon_{\{v,v_k\}} - L_{v_k}(t) \geq \\ &\geq (s-1) \cdot (\kappa_P - \kappa_{P'}) - \epsilon_{\{v,v_k\}}. \end{aligned} \tag{6}$$

To show (FC1) is satisfied, let P' be the subpath $v_0, \ldots, v_{k-1}$ of P, where $v_{k-1} \in N(v)$. Note that since $\Xi_P(t) > 0$ it must be that $k > 0$, and thus v_{k-1} is well-defined. For this choice of P', (6) yields

$$\tilde{L}_{v_k}^{v_{k-1}}(t) - L_{v_k}(t) \geq (s-1) \cdot (\kappa_P - \kappa_{P'}) - \epsilon_{\{v_{k-1},v_k\}} = \\ = (s-1) \cdot \kappa_{\{v_{k-1},v_k\}} - \epsilon_{\{v_{k-1},v_k\}} \overset{(4)}{>} (s-1-\lambda)\,\kappa_{\{v_{k-1},v_k\}}.$$

This shows that (FC1) is satisfied. To show that (FC2) holds, let $v \in N(v_k)$ be any neighbor of v_k, and let $P' = v_0, \ldots, v_k, v$ be the path obtained by appending v to the path P. In this case (6) yields

$$L_{v_k}(t) - \tilde{L}_{v_k}^{v}(t) \leq (s-1) \cdot (\kappa_{P'} - \kappa_P) + \epsilon_{\{v,v_k\}} = \\ = (s-1) \cdot \kappa_{\{v,v_k\}} + \epsilon_{\{v,v_k\}} \overset{(4)}{<} (s-1+\lambda) \cdot \kappa_{\{v,v_k\}}.$$

Hence, the second condition is satisfied as well, and node v_k is in fast mode. □

Suppose that at time t, node v has $\Xi_v^s(t) > 0$. From Lemma 4, all the nodes that maximize Ξ_v^s are in fast mode, trying to catch up to v, and their logical clock rate is at least $(1-\rho)(1+\mu)$. Thus, whenever it is positive, Ξ_v^s decreases at an average rate of at least $(1-\rho)(1+\mu)$, *minus* the rate by which v increases its own logical clock. To formalize this observation, define

$$\mathcal{I}_v(t_1, t_2) := L_v(t_2) - L_v(t_1) \tag{7}$$

to be the amount by which v increases its logical clock over the time interval $[t_1, t_2]$. Since $\frac{d}{dt} L_v(t) \geq \frac{d}{dt} H_v(t) \geq 1 - \rho$ we have the following property.

Property 3. For all nodes v and times t_1, t_2 we have $\mathcal{I}_v(t_1, t_2) \geq (1-\rho)(t_2 - t_1)$.

Now we can state the following lemma.

Lemma 6 (Catch-Up Lemma). *Let v_0 be a node and let $[t_0, t_1]$ be a time interval such that for all $t \in (t_0, t_1)$ we have $\Xi_{v_0}^s(t) > 0$. Then for all $t \in [t_0, t_1]$,*

$$\Xi_{v_0}^s(t) \leq \Xi_{v_0}^s(t_0) + \mathcal{I}_{v_0}(t_0, t) - (1-\rho)(1+\mu)(t - t_0).$$

Similarly, whenever $\Psi_v^s(t) > 0$, the nodes that maximize Ψ_v^s are in slow mode, and their logical clocks increase at a rate of at most $1 + \rho$. Thus, whenever it is positive, $\Psi_v^s(t)$ increases at an average rate of at most $1 + \rho$, again minus v's increase to its own logical clock. This is captured by the following lemma.

Lemma 7 (Waiting Lemma). *Let v_0 be a node and let $[t_0, t_1]$ be a time interval such that for all $t \in (t_0, t_1)$ we have $\Psi_{v_0}^s(t) > 0$. Then for all $t \in [t_0, t_1]$,*

$$\Psi_{v_0}^s(t) \leq \Psi_{v_0}^s(t_0) - \mathcal{I}_{v_0}(t_0, t) + (1+\rho)(t - t_0).$$

The proofs of Lemmas 6 and 7 involve a straightforward application of Lemma 2.

We have so far argued that if v_0 is too far ahead of other nodes then those nodes will be in fast mode, and if v_0 is too far behind other nodes then those nodes will be in slow mode. What does v_0 *itself* do when it is too far behind? Observe that if there is some path $P = v_0, \ldots, v_k$ such that $\Psi^s_P(t) > 0$, then for the inverted path $P' = v_k, \ldots, v_0$ we have $\Xi^s_{P'}(t) > \Psi^s_P(t) > 0$. Thus, informally speaking, whenever v_0 is too far behind some other node it will be "pulled forward" at the fast rate. The next lemma quantifies how much ground v_0 makes up during an interval in which it is far behind: it states that given sufficient time, the node makes up all the initial weighted skew Ψ^s_v, *in addition* to its minimal rate of progress $(1-\rho)$.

Lemma 8. *For any node v_0, integer $s \in \mathbb{N}^{>0}$ and time interval $[t_0, t_1]$ where $t_1 \geq t_0 + \frac{C_{s-1}}{(1-\rho)\mu}$, if the network is in a legal state at time t_0, then*

$$\mathcal{I}_{v_0}(t_0, t_1) \geq \Psi^s_{v_0}(t_0) + (1-\rho)(t_1 - t_0).$$

Proof (Lemma 8). If $\Psi^s_{v_0}(t_0) \leq 0$, the claim follows immediately from Property 3. Thus, assume that $\Psi^s_{v_0}(t_0) > 0$, and let $P = v_0, \ldots, v_k$ be a path such that $\Psi^s_P(t_0) = \Psi^s_{v_0}(t_0)$. From the definitions of Ψ and Ξ, for the inverted path $P' = v_k, \ldots, v_0$ we have $\Xi^s_{P'}(t_0) > \Psi^s_P(t_0)$, and therefore, $\Xi^s_{v_k}(t_0) > \Psi^s_{v_0}(t) > 0$. If there is a time $t \in [t_0, t_1]$ such that $\Xi^s_{v_k}(t) \leq 0$, let $\bar{t}$ be the infimum of such times. Otherwise, let $\bar{t} = t_1$. Observe that

$$\begin{aligned}\mathcal{I}_{v_0}(t_0, \bar{t}) &= L_{v_0}(\bar{t}) - L_{v_0}(t_0) = \Xi^s_{P'}(t_0) - \Xi^s_{P'}(\bar{t}) + \mathcal{I}_{v_k}(t_0, \bar{t}) \\ &> \Psi^s_P(t_0) - \Xi^s_{v_k}(\bar{t}) + \mathcal{I}_{v_k}(t_0, \bar{t}) = \Psi^s_{v_0}(t_0) - \Xi^s_{v_k}(\bar{t}) + \mathcal{I}_{v_k}(t_0, \bar{t}).\end{aligned}$$

Since $\bar{t} \leq t_1$ and $\mathcal{I}_{v_0}(t_0, \cdot)$ is increasing and interval-additive, to prove the claim it is sufficient to show that $\mathcal{I}_{v_k}(t_0, \bar{t}) \geq \Xi^s_{v_k}(\bar{t}) + (1-\rho)(\bar{t} - t_0)$.

Consider first the case where $\bar{t} < t_1$. In this case $\bar{t}$ is the infimum of times t where $\Xi^s_{v_k}(t) \leq 0$. Since $\Xi^s_{v_k}(\cdot)$ is continuous, it follows that $\Xi^s_{v_k}(\bar{t}) = 0$, and using Property 3 we obtain $\mathcal{I}_{v_k}(t_0, \bar{t}) \geq \Xi^s_{v_k}(\bar{t}) + (1-\rho)(\bar{t} - t_0)$.

Otherwise, if $\bar{t} = t_1$, then for all $t \in [t_0, t_1)$ we have $\Xi^s_{v_k}(t) > 0$. Applying Lemma 6 to the interval $[t_0, t_1]$ we obtain

$$\begin{aligned}\Xi^s_{v_k}(t_1) &\leq \Xi^s_{v_k}(t_0) + \mathcal{I}_{v_k}(t_0, t_1) - (1-\rho)(1+\mu)(t_1 - t_0) \leq \\ &\overset{\text{Lemma 3}}{\leq} C_{s-1} + \mathcal{I}_{v_k}(t_0, t_1) - (1-\rho)\mu \cdot \frac{C_{s-1}}{(1-\rho)\mu} - (1-\rho)(t_1 - t_0) = \\ &= \mathcal{I}_{v_k}(t_0, t_1) - (1-\rho)(t_1 - t_0),\end{aligned}$$

which yields the desired result. □

Now we are ready to put all the pieces together and prove the main theorem:

Theorem 1. *The network is always in a legal state.*

Proof. Suppose for the sake of contradiction that this is not the case, and let $\bar{t}$ be the infimum of times when the legal state condition is violated. Then there is some path $P = v_0, \ldots, v_k$ and some $s \geq 1$ such that $\kappa_P \geq C_s$ but

$$L_{v_0}(\bar{t}) - L_{v_k}(\bar{t}) \geq s \cdot \kappa_P. \tag{8}$$

For the legal state condition to be violated, the system must be far outside the boundary of $\boldsymbol{\Psi}^s$:

$$\Psi^s_{v_k}(\bar{t}) \geq L_{v_0}(\bar{t}) - L_{v_k}(\bar{t}) - \left(s - \frac{1}{2}\right) \cdot \kappa_P \overset{(8)}{\geq} \frac{1}{2}\kappa_P \geq \frac{1}{2}C_s = \frac{1}{2\sigma}C_{s-1}. \quad (9)$$

However, Lemma 7 tells us that whenever $\Psi^s_{v_k}$ is large it cannot increase quickly, which gives v_k time to catch up. Specifically, if t_0 is the supremum of times $t \leq \bar{t}$ such that $\Psi^s_{v_k}(t) \leq 0$, then Lemma 7 shows that

$$\Psi^s_{v_k}(\bar{t}) \leq \Psi^s_{v_k}(t_0) - \mathcal{I}_{v_k}(t_0, \bar{t}) + (1+\rho)(\bar{t} - t_0) \overset{\text{(Prop. 3)}}{\leq} 2\rho(\bar{t} - t_0). \quad (10)$$

Let $t_1 := \bar{t} - \frac{C_{s-1}}{(1-\rho)\mu}$. Combining (9) and (10), we see that $t_0 \leq \bar{t} - \frac{C_{s-1}}{4\sigma\rho} \overset{(5)}{\leq} \bar{t} - \frac{C_{s-1}}{(1-\rho)\mu} = t_1$. Thus, by Lemma 8, the interval $[t_1, \bar{t}]$ is sufficient for v_k to increase its clock by

$$\mathcal{I}_{v_k}(t_1, \bar{t}) \geq \Psi^s_{v_k}(t_1) + (1-\rho)(\bar{t} - t_1). \quad (11)$$

Applying Lemma 7 again, we obtain

$$\Psi^s_{v_k}(\bar{t}) \overset{\text{(Lemma 7)}}{\leq} \Psi^s_{v_k}(t_1) - \mathcal{I}_{v_k}(t_1, \bar{t}) + (1+\rho)(\bar{t} - t_1) \overset{(11)}{\leq} 2\rho\frac{C_{s-1}}{(1-\rho)\mu} \overset{(5)}{<} \frac{1}{2\sigma}C_{s-1},$$

in contradiction to (9).

As an easy corollary we obtain the following.

Theorem 2. *The global skew of the algorithm is bounded by* $\frac{4}{\lambda} \cdot \mathcal{D}$. □

There is some flexibility in setting the parameter μ, which governs the maximal speed of the logical clocks. We illustrate two possible choices.

By (5), the choice of μ limits the choice of σ, the parameter introduced in Definition 3. The value of σ serves as the base of the logarithm in the gradient function. If we set $\mu \approx 1$, we can set $\sigma = \Theta(1/\rho)$, achieving optimal gradient clock-synchronization and matching the lower bound of [10]. However, for this choice of μ we get $\beta \approx 1$, meaning that in fast mode logical clocks progress at almost twice the rate of real time. An alternative is to choose μ as $\Theta(\rho/(1-\rho))$, which yields $\beta = O(\rho)$. The cost is choosing σ as a constant, which is no longer optimal. This is formalized below.

Corollary 1. *If* $\mu = \Theta(1/(1-\rho))$, *the algorithm achieves* $O\left(\operatorname{dist}(u,v) \cdot \log_{1/\rho} \mathcal{D}\right)$*-gradient synchronization, with a global skew of* $O(\mathcal{D})$. □

Corollary 2. *If* $\mu = \Theta(\rho/(1-\rho))$, *the algorithm achieves* $O\left(\operatorname{dist}(u,v) \cdot \log \mathcal{D}\right)$*-gradient synchronization, with a global skew of* $O(\mathcal{D})$. *When using direct and RBS estimates, we get* $\mathcal{D} = O\big((1+\rho D)(\Delta H + \mathcal{T}) + u_{\mathrm{rcv}} \cdot D\big)$ *in this case, where* D *is the diameter of* G. □

7 Conclusion

In this work we introduced a method of seamlessly incorporating reference broadcast synchronization (RBS) into the theoretical study of internal clock synchronization. We argue that by separating the task of estimating other nodes' clock values from the task of combining these estimates to come up with a logical clock, one obtains a cleaner and more general framework.

The approach taken here is reminiscent of, e.g., [2], [14] and [15]; in these works, various assumptions on message delay are represented as bounds on the difference between times at which a pair of events occur (e.g., the sending and receipt of a message). Using this abstract representation, general algorithms are presented which can handle many forms of delay assumptions. We take one step further by decoupling the physical network graph from the estimate graph, which is used to represent all information that nodes can acquire about each other. The approach extends to handle not only RBS but any means of acquiring clock estimates. For example, if certain nodes in the network have access to an external time source such as GPS, the estimate graph would contain edges between such nodes, since they can use the external time source to estimate each others' clock values (similar to RBS).

The estimate graph serves as a base layer on which classical clock synchronization algorithms can run, including, with minor modifications, those of [2] and [15] (which provide only global clock synchronization). In this work we further modified a recent gradient clock synchronization algorithm [10], to obtain an algorithm that provides gradient synchronization in broadcast networks.

References

1. Attiya, H., Hay, D.C., Welch, J.L.: Optimal clock synchronization under energy constraints in wireless ad-hoc networks. In: Anderson, J.H., Prencipe, G., Wattenhofer, R. (eds.) OPODIS 2005. LNCS, vol. 3974, pp. 221–234. Springer, Heidelberg (2006)
2. Attiya, H., Herzberg, A., Rajsbaum, S.: Optimal clock synchronization under different delay assumptions. SIAM Journal on Computing 25(2), 369–389 (1996)
3. Cristian, F.: Probabilistic clock synchronization. Distributed Computing 3, 146–158 (1989)
4. Elson, J., Girod, L., Estrin, D.: Fine-grained network time synchronization using reference broadcasts. ACM SIGOPS Operating Systems Review 36(SI), 147–163 (2002)
5. Fan, R., Chakraborty, I., Lynch, N.: Clock synchronization for wireless networks. In: Higashino, T. (ed.) OPODIS 2004. LNCS, vol. 3544, pp. 400–414. Springer, Heidelberg (2005)
6. Fan, R., Lynch, N.: Gradient clock synchronization. Distributed Computing 18(4), 255–266 (2006)
7. Halpern, J., Megiddo, N., Munshi, A.: Optimal precision in the presence of uncertainty. In: Proc. of 17th Symp. on Theory of Computing (STOC), pp. 346–355 (1985)

8. Karp, R., Elson, J., Papadimitriou, C., Shenker, S.: Global synchronization in sensornets. In: Farach-Colton, M. (ed.) LATIN 2004. LNCS, vol. 2976, pp. 609–624. Springer, Heidelberg (2004)
9. Lenzen, C., Locher, T., Wattenhofer, R.: Clock synchronization with bounded global and local skew. In: Prof. of 49th IEEE Symp. on Foundations of Computer Science (FOCS), pp. 500–510 (2008)
10. Lenzen, C., Locher, T., Wattenhofer, R.: Tight bounds for clock synchronization. In: Proc. of the 28th ACM Symp. on Principles of Distributed Computing (PODC) (to appear, 2009)
11. Locher, T., Wattenhofer, R.: Oblivious gradient clock synchronization. In: Dolev, S. (ed.) DISC 2006. LNCS, vol. 4167, pp. 520–533. Springer, Heidelberg (2006)
12. Lundelius, J., Lynch, N.: An upper and lower bound for clock synchronization. Information and Control 62(2/3), 190–204 (1984)
13. Meier, L., Thiele, L.: Brief announcement: gradient clock synchronization in sensor networks. In: Proc. of 24th ACM Symp. on Principles of Distributed Computing (PODC), p. 238 (2005)
14. Moses, Y., Bloom, B.: Knowledge, timed precedence and clocks (preliminary report). In: Proc. of the 13th ACM Symp. on Principles of Distributed Computing (PODC), pp. 294–303 (1994)
15. Ostrovsky, R., Patt-Shamir, B.: Optimal and efficient clock synchronization under drifting clocks. In: Proc. of 18th ACM Symp. on Principles of Distributed Computing (PODC), pp. 400–414 (1999)
16. Patt-Shamir, B., Rajsbaum, S.: A theory of clock synchronization. In: Proc. of 26th ACM Symp. on Theory of Computing (STOC), pp. 810–819 (1994)
17. Pussente, R.M., Barbosa, V.C.: An algorithm for clock synchronization with the gradient property in sensor networks. J. Parallel Distrib. Comput. 69(3), 261–265 (2009)
18. Römer, K.: Time synchronization in ad hoc networks. In: Proc. of 2nd Symp. on Mobile Ad Hoc Networking and Computing (MOBIHOC), pp. 173–182 (2001)
19. Sivrikaya, F., Yener, B.: Time synchronization in sensor networks: A survey. IEEE Network 18(4), 45–50 (2004)
20. Srikanth, T.K., Toueg, S.: Optimal clock synchronization. Journal of the ACM 34(3), 626–645 (1987)
21. Sundararaman, B., Buy, U., Kshemkalyani, A.: Clock synchronization for wireless sensor networks: A survey. Ad Hoc Networks 3(3), 281–323 (2005)