Lecture 2

Gradient Clock Synchronization

In the previous lesson, we proved essentially matching upper and lower bounds on the worst-case global skew for the clock synchronization problem. We saw that during an execution of the Max algorithm (Algorithm 1.2), all logical clocks in all executions eventually agree up to an additive term of $\mathcal{O}(uD)$ (ignoring other parameters). The lower bound we proved in Section 1.3 shows that global skew of $\Omega(uD)$ is unavoidable for any algorithm in which clocks run at an amortized constant rate, at least in the worst case. In our lower bound construction, the two nodes v and w that achieved the maximal skew were distance D apart. However, the lower bound did not preclude neighboring nodes from remaining closely synchronized throughout an execution. In fact, this is straightforward if one is willing to slow down clocks arbitrarily (or simply stop them), even if the amortized rate is constant.

Today, we look into what happens if one requires that clocks progress at a constant rate at all times. In many applications, it is sufficient that neighboring clocks are closely synchronized, while nodes that are further apart are only weakly synchronized. To model this situation, we introduce the *gradient clock synchronization (GCS) problem*. Intuitively, this means that we want to ensure a small skew between neighbors despite maintaining "proper" clocks. That is, we minimize the *local skew* under the requirement that logical clocks always run at least at rate 1.

2.1 Formalizing the Problem

Let $G = (V, E)$ be a network. As in the previous lecture, each node $v \in V$ has a hardware clock $H_v : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ that satisfies for all $t, t' \in \mathbb{R}_0^+$ with $t' < t$

$$t - t' \leq H_v(t) - H_v(t') \leq \vartheta(t - t') \ .$$

Again, we denote by $h_v(t)$ the rate of $H_v(t)$ at time t, i.e., $1 \leq h(t) \leq \vartheta$ for all $t \in \mathbb{R}_0^+$. Recall that each node v computes a logical clock $L_v : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ from its hardware clock and messages received from neighbors. During an execution $\mathcal{E}$, for each edge $e = \{v, w\} \in E$, we define the *local skew* of e at time t to be

$\mathcal{L}_e(t) = |L_v(t) - L_w(t)|$. The *gradient skew at time* t in the network, denoted $\mathcal{L}(t)$, is the largest local skew across any edge: $\mathcal{L}(t) = \max_{e \in E} \mathcal{L}_e(t)$. Finally, the *gradient skew over an execution* $\mathcal{E}$ is defined to be

$$\mathcal{L} = \sup_{t \in \mathbb{R}_0^+} \{\mathcal{L}(t)\} \ .$$

The goal of the *gradient clock synchronization problem* is to minimize $\mathcal{L}$ for any possible execution $\mathcal{E}$.

Attention: In order to simplify our presentation of the gradient clock synchronization problem, we abstract away from the individual messages and message delays from the previous chapter. Instead, we assume that throughout an execution, each node v maintains an estimate of its neighbors' logical clocks. Specifically, for each neighbor $w \in N_v$, v maintains a variable $\tilde{L}_w^v(t)$. The parameter δ represents the *error* in the estimates: for all $\{v, w\} \in E$ and $t \in \mathbb{R}_0^+$, we have

$$L_w(t) \geq \tilde{L}_w^v(t) > L_w(t) - \delta \, . \tag{2.1}$$

When the node v is clear from context, we will omit the superscript v, and simply write $\tilde{L}_w$.

In order to obtain the estimates $\tilde{L}_w^v(t)$, each node w periodically broadcasts its logical clock value to its neighbors. Each neighbor v then computes $\tilde{L}_w^v(t)$ using the known bounds on message delays, and increases $\tilde{L}_w^v$ at rate h_v/ϑ between messages from w. Thus, an upper bound on the error parameter δ can be computed as a function of u (the uncertainty in message delay), ϑ (the maximum clock drift), T (the frequency of broadcasts), and μ (a parameter determining how fast logical clocks may run, see below); you do this in the exercises.

To focus on the key ideas, we make another simplifying abstraction: Instead of analyzing the global skew, we assume that it is taken care of and plug in $\mathcal{G}$ as a parametrized upper bound. You will address this issue as an exercise, too.

2.2 Averaging Protocols

In this section, we consider a natural strategy for achieving gradient clock synchronization: trying to bring the own logical clock to the average value between the neighbors whose clocks are furthest ahead and behind, respectively. Specifically, each node can be in either *fast mode* or *slow mode*. If a node v detects that its clock is behind the average of its neighbors, it will run in fast mode, and increase its logical clock at a rate faster than its hardware clock by a factor of $1 + \mu$, where μ is some appropriately chosen constant. On the other hand, if v's clock is at least the average of its neighbors, it will run in slow mode, increasing its logical clock only as quickly as its hardware clock. Note that this strategy results in logical clocks that behave like "real" clocks of drift $\vartheta' = \vartheta(1 + \mu) - 1$. If $\mu \in \mathcal{O}(\vartheta)$, these clocks are roughly as good as the original hardware clocks.

The idea of switching between fast and slow modes gives a well-defined protocol if neighboring clock values are known precisely,[1] however ambiguity

[1]There is one issue of pathological behavior in which nodes could switch infinitely quickly between fast and slow modes. This can be avoided by introducing a small threshold δ so that a node only changes, say, from slow to fast mode if it detects that its clock is δ time units behind the average.

arises in the presence of uncertainty.

We consider two natural ways of dealing with the uncertainty. Set $L_{N_v}^{\max}(t) := \max_{w \in N_v}\{L_w\}$ and $L_{N_v}^{\min}(t) := \min_{w \in N_v}\{L_w\}$.

Aggresive strategy: each v computes an *upper bound* on the average between $L_{N_v}^{\max}$ and $L_{N_v}^{\min}$, and determines whether to run in fast or slow mode based on this upper bound;

Conservative strategy: each v computes a *lower bound* on the average between $L_{N_v}^{\max}$ and $L_{N_v}^{\min}$ and determines the mode accordingly.

We will see that, in fact, both strategies yield terrible results, but for opposite reasons. In Section 2.3, we will derive an algorithm that strikes an appropriate balance between both stragies, with impressive results!

Aggressive Averaging

Here we analyze the aggressive averaging protocol described above. Specifically, each node $v \in V$ computes an upper bound on the average of its neighbors' logical clock values:

$$\tilde{L}_v^{\text{up}}(t) = \frac{\max_{w \in N_v}\{\tilde{L}_w\} + \min_{w \in N_v}\{\tilde{L}_w\}}{2} + \delta \geq \frac{L_{N_v}^{\max} + L_{N_v}^{\min}}{2}.$$

The algorithm then increases the logical clock of v at a rate of $h_v(t)$ if $L_t(t) > \tilde{L}_v^{\text{up}}(t)$, and a rate of $(1+\mu)h_v(t)$ otherwise. We show that the algorithm performs poorly for any choice of $\mu \geq 0$.

Claim 2.1. *Consider the aggressive averaging protocol on a path network of diameter D, i.e., $V = \{v_i \,|\, i \in [D+1]\}$ and $E = \{v_i, v_{i+1}\} \,|\, i \in [D]$. Then there exists an execution $\mathcal{E}$ such that the gradient skew satisfies $\mathcal{L} \in \Omega(\delta D)$.*

Proof Sketch. Throughout the execution, we will assume that all clock estimates are correct: for all $v \in V$ and $w \in N_v$, we have $\tilde{L}_v^w(t) = L_w(t)$. This means for all $i \in [D]\setminus\{0\}$ that $\tilde{L}_{v_i}^{\text{up}}(t) = (L_{v_{i-1}}(t)+L_{v_{i+1}}(t))/2+\delta$, whereas $\tilde{L}_{v_0}^{\text{up}}(t) = L_{v_1}(t)+\delta$ and $\tilde{L}_{v_D}^{\text{up}} = L_{v_{D-1}}(t)+\delta$. Initially, the hardware clock rate of node v_i is $1+\frac{i(\vartheta-1)}{D}$. Thus, even though all nodes immediately "see" that skew is building up, they all set their clock rates to fast mode in order to catch up in case they underestimate their neighbors' clock values.

Now let's see what happens to the logical clocks in this execution. While nodes are running fast, skew keeps building up, but the property that $L_{v_i}(t) = (L_{v_{i+1}}(t) - L_{v_{i-1}}(t))$ is maintained at nodes $i \in [D] \setminus \{0\}$. In this state, v_0 — despite running fast — has no way of catching up to v_1. However, at time $\tau_0 := \frac{\delta D}{(1+\mu)(\vartheta-1)}$ we would have that $L_{v_D}(\tau_0) = L_{v_{D-1}}(\tau_0) + \delta = \tilde{L}_{v_D}^{\text{up}}(\tau_0)$ and v_D would stop running fast. We set $t_0 := \tau_0 - \varepsilon$ for some arbitrarily small $\varepsilon > 0$ and set $h_{v_D}(t) := h_{v_{D-1}}(t)$ for all $t \geq t_0$. Thus, all nodes would remain in fast mode until the time $\tau_1 := t_0 + \frac{\delta D}{(1+\mu)(\vartheta-1)}$ when we had $L_{v_{D-1}}(\tau_1) = \tilde{L}_{v_{D-1}}^{\text{up}}(\tau_1)$. We set $t_1 := \tau_1 - \varepsilon$ and proceed with this construction inductively. Note that, with every hop, the local skew increases by (almost) 2δ, as this is the additional skew that L_{v_i} must build up to $L_{v_{i-1}}$ when $L_{v_{i+1}} = L_{v_i}$ in order to increase $\tilde{L}_{v_i}^{\text{up}} - L_{v_i}$ by δ, i.e., for v_i to stop running fast. As ε is arbitrarily small, we build up a local skew that is arbitrarily close to $(2D-1)\delta$. $\square$

Remarks:

- The algorithm is also bad in that the above execution results in a global skew of $\Omega(\delta D^2)$.
- This could be fixed fairly easily, but without further changes still a large local skew could build up.
- The above argument can be generalized to arbitrary graphs, by taking two nodes $v, w \in V$ in distance D and using the function $d(x) = d(x,v) - d(x,w)$, just as in Lemma 1.5.

Conservative Averaging

Let's be more careful. Now each node $v \in V$ computes a *lower* bound on the average of its neighbors' logical clock values:

$$\tilde{L}_v^{\mathrm{up}}(t) = \frac{\max_{w\in N_v}\{\tilde{L}_w\} + \min_{w\in N_v}\{\tilde{L}_w\}}{2} \leq \frac{L_{N_v}^{\max} + L_{N_v}^{\min}}{2} .$$

The algorithm then increases the logical clock of v at a rate of $h_v(t)$ if $L_v(t) > \tilde{L}_v^{\mathrm{up}}(t)$, and a rate of $(1+\mu)h_v(t)$ otherwise. Again, the algorithm fails to achieve a small local skew.

Claim 2.2. *Consider the conservative averaging protocol on a path network of diameter D. Then there exists an execution $\mathcal{E}$ such that the gradient skew satisfies $\mathcal{L} \in \Omega(\delta D)$.*

Proof Sketch. We do the same as for the aggressive strategy, except that now for each $v \in V$, $w \in N_w$, and time t, we rule that $\tilde{L}_w(t) = L_w(t) - \delta + \varepsilon$ for some arbitrarily small $\varepsilon > 0$. Thus, all nodes are initially in slow mode. We inductively change hardware clock speeds just before nodes would switch to fast mode, building up the exact same skews between logical clocks as in the previous execution. The only difference is that now it does not depend on μ how long this takes! □

Remarks:

- It seems as if we just can't do things right. Both the aggressive and the conservative strategy do not result in a proper response to the gobal distribution of clock values.
- Surprisingly, mixing the two strategies works! We study this during the remainder of the lecture.

2.3 GCS Algorithm

The high-level strategy of the algorithm is as follows. As above, at each time each node can be either in *slow mode* or *fast mode.* In slow mode, a node v will increase its logical clock at rate $h_v(t)$. In fast mode, v will increase its logical clock at rate $(1 + \mu)h_v(t)$. The parameter μ will be chosen large enough for nodes whose logical clocks are behind to be able to catch up to other nodes. The conditions for a node to switch from slow to fast or vice versa are simple, but perhaps unintuititve. In what follows, we first describe "ideal" conditions to switch between modes. In the ideal behavior, each node knows exactly the logical clock values of its neighbors. Since the actual algorithm only has access to estimates of neighboring clocks, we then describe fast and slow triggers for switching between modes that can be implemented in our model for GCS. We conclude the section by proving that the triggers do indeed implement the conditions.

Fast and Slow Conditions

Definition 2.3 (**FC**: Fast Mode Condition)**.** *We say that a node $v \in V$ satisfies the* fast mode condition (**FC**) *at time $t \in \mathbb{R}_0^+$ if there exists $s \in \mathbb{N}$ such that:*

***FC** 1: $\exists x \in N_v : L_x(t) - L_v(t) \geq 2s\delta$;*

***FC** 2: $\forall y \in N_v : L_v(t) - L_y(t) \leq 2s\delta$.*

Informally, **FC** 1 says that v has a neighbor x whose logical clock is significantly ahead of $L_v(t)$, while **FC** 2 stipulates that none of v's neighbors' clocks is too far behind $L_v(t)$. In particular, if **FC** is satisfied with $x \in N_v$ satisfying **FC** 1, then the local skew across $\{v, x\}$ is at least $2s\delta$, where L_x is at least $2s\delta$ time units ahead of L_v. Since none of v's neighbors are running more than $2s\delta$ units behind L_v, v can decrease the maximum skew with its neighbors by increasing its logical clock.

The slow mode condition below is dual to **FC**. It essentially gives conditions under which v could decrease the maximum skew in its neighborhood by decreasing its logical clock.

Definition 2.4 (**SC**: Slow Mode Condition)**.** *We say that a node $v \in V$ satisfies the* slow mode condition *(or **SC**) at time $t \in \mathbb{R}_0^+$ if there exists $s \in \mathbb{N}$ such that:*

***SC** 1: $\exists x \in N_v : L_v(t) - L_x(t) \geq (2s-1)\delta$;*

***SC** 2: $\forall y \in N_v : L_y(t) - L_v(t) \leq (2s-1)\delta$.*

Substracting an additional δ in **SC** 1 and **SC** 2 ensures that conditions **FC** and **SC** are mutually exclusive. Together, the conditions mean that, if in doubt, the algorithm alternates between aggressively seeking to reduce skew towards neighbors that are ahead (**FC**) and conservatively avoiding to build up additional skew to neighbors that are behind (**SC**), depending on the currently observed average skew.

Fast and Slow Triggers

While the fast and slow mode conditions described in the previous section are well-defined (and mutually exclusive), uncertainty on neighbors' clock values prevents an algorithm from checking the conditions directly. Here we define corresponding *triggers* that our computational model does allow us to check.

The separation of δ between the conditions is just enough for this purpose. As we assumed that clock values are never overestimated, but may be underestimated by δ, the fast mode trigger needs to shift its thresholds by δ.

Definition 2.5 (**FT**: Fast Mode Trigger)**.** *We say that $v \in V$ satisfies the* fast mode trigger (**FT**) *at time $t \in \mathbb{R}_0^+$ if there exists an integer $s \in \mathbb{N}$ such that:*

***FT** 1:* $\exists x \in N_v : \tilde{L}_x(t) - L_v(t) > (2s-1)\delta$ *;*

***FT** 2:* $\forall y \in N_v : L_v(t) - \tilde{L}_y(t) < (2s+1)\delta$ *.*

Definition 2.6 (**ST**: Slow Mode Trigger)**.** *We say that a node $v \in V$ satisfies the* slow mode trigger *(or **ST**) at time $t \in \mathbb{R}_0^+$ if there exists $s \in \mathbb{N}$ such that:*

***ST** 1:* $\exists x \in N_v : L_v(t) - \tilde{L}_x(t) \geq (2s-1)\delta$ *;*

***ST** 2:* $\forall y \in N_v : \tilde{L}_y(t) - L_v(t) \leq (2s-1)\delta$ *.*

Before we formally describe the GCS algorithm, we give two preliminary results about the fast and slow mode triggers. The first result claims that **FT** and **ST** cannot simultaneously be satisfied by the same node. The second shows that **FT** and **ST** implement **FC** and **SC**, respectively. That is, if the fast (resp. slow) mode condition is satisfied, then the fast (resp. slow) mode trigger is also satisfied.

Lemma 2.7. *No node $v \in V$ can simultaneously satisfy **FT** and **ST**.*

Proof. Suppose v satisfies **FT**, i.e., there is $s \in \mathbb{N}$ so that there is some $x \in N_v$ such that $\tilde{L}_x(t) - L_v(t) > (2s-1)\delta$ and for all $y \in N_v$ we have $L_v(t) - \tilde{L}_y(t) < (2s+1)\delta$. Consider $s' \in \mathbb{N}$. If $s' > s$, then for all $y \in N_v$ we have that

$$L_v(t) - \tilde{L}_x(t) < (2s+1)\delta \leq (2s'-1)\delta\,,$$

so **ST** 1 is not satisfied for s'. If $s' \leq s$, then there is some $x \in N_v$ so that

$$\tilde{L}_x(t) - L_v(t) > (2s-1)\delta \geq (2s'-1)\delta\,,$$

so **ST** 2 is not satisfied for s'. Hence, **ST** is not satisfied. □

Lemma 2.8. *Suppose $v \in V$ satisfies **FC** (resp. **SC**) at time t. Then v satisfies **FT** (resp. **SC**) at time t.*

Proof. Suppose **FC** holds (at time t). Then, by (2.1), there is some $s \in \mathbb{N}$ such that

$$\exists x \in \mathbb{N}_v : \tilde{L}_x(t) - L_v(t) > L_x(t) - \delta - L_v(t) \geq (2s-1)\delta$$

and

$$\forall y \in \mathbb{N}_v : L_v(t) - \tilde{L}_y(t) < L_v(t) - L_y(t) + \delta \leq (2s+1)\delta\,,$$

i.e., **FT** holds. Similarly, if **SC** holds, (2.1) yields that

$$\exists x \in \mathbb{N}_v : L_v(t) - \tilde{L}_x(t) \geq L_v(t) - L_x(t) \geq (2s-1)\delta$$

and

$$\forall y \in \mathbb{N}_v : \tilde{L}_y(t) - L_x(t) \leq L_y(t) - L_v(t) \leq (2s-1)\delta$$

for some $s \in \mathbb{N}$, establishing **ST**. □

We now describe the GCS algorithm. Each node v initializes its logical clock to its hardware clock value. It continuously checks if the fast (resp. slow) mode trigger is satisfied. If so, it increases its logical clock at a rate of $(1+\mu)h_v(t)$ (resp. $h_v(t)$). Pseudocode is presented in Algorithm 2.1. The algorithm itself is simple, but the analysis of the algorithm (presented in the following section) is rather delicate.

Algorithm 2.1: GCS algorithm

```
1 $L_v(0) := H_v(0)$
2 $r := 1$
3 at all times $t$ do the following
4 if FT then
5 |   $r := 1 + \mu$                              // v is in fast mode
6 if ST then
7 |   $r := 1$                                    // v is in slow mode
8 increase $L_v$ at rate $rh_v(t)$
```

Remarks:

- In fact, when neither **FT** nor **ST** hold, the logical clock may run at any speed from the range $[h_v(t), (1+\mu)h_v(t)]$.
- In order for the algorithm to be implementable, δ should leave some wiggle space. We expressed this by having (2.1) include a strict inequality, but if the inequality can become arbitrarily tight, the algorithm may have to switch between slow and fast mode arbitrarily fast.
- For technical reasons, we will assume that logical clocks are differentiable. Thus, $l_v := \frac{d}{dt}L_v$ exists and is between 1 and $\vartheta(1+\mu)$ at all times. It is possible to prove the guarantees of the algorithm without this assumption, but all this does is making the math harder.
- Even with this assumption, we still need Lemma A.1. This is not a mathematics lecture, but as we couldn't find any suitable reference, the lemma and a proof is given in the appendix.

2.4 Analysis of the GCS Algorithm

We now show that the GCS algorithm (Algorithm 2.1) indeed achieves a small local skew, which is expressed by the following theorem.

Theorem 2.9. *For every network G and every execution $\mathcal{E}$ in which $H_v(0) - H_w(0) \leq \delta$ for all edges $\{v, w\} \in E$, the GCS algorithm achieves a gradient skew of $\mathcal{L} \leq 2\delta\lceil\log_\sigma \mathcal{G}/\delta\rceil$, where $\sigma := \mu/(\vartheta - 1)$.*

In order to prove Theorem 2.9, we analyze the *average* skew over paths in G of various lengths. For long paths of $\Omega(D)$ hops, we will simply exploit that $\mathcal{G}$ bounds the skew between *any* pair of nodes. For successively shorter paths, we inductively show that the average skew between endpoints cannot increase too quickly: reducing the length of a path by factor σ can only increase the skew between endpoints by an additive constant term. Thus, paths of constant length (in particular edges) can only have a skew that is logarithmic in the network diameter.

Leading Nodes

We start by showing that skew cannot build up too quickly. This is captured by the following functions.

Definition 2.10 (Ψ and Leading Nodes)**.** *For each $v \in V$, $s \in \mathbb{N}$, and $t \in \mathbb{R}_0^+$, we define*

$$\Psi_v^s(t) = \max_{w \in V}\{L_w(t) - L_v(t) - (2s-1)\delta d(v,w)\}\,,$$

where $d(v,w)$ denotes the distance between v and w in G. Moreover, set

$$\Psi^s(t) = \max_{w \in V}\{\Psi_w^s(t)\}\,.$$

Finally, we say that $w \in V$ is a leading node *if there is some $v \in V$ so that*

$$\Psi_v^s(t) = L_w(t) - L_v(t) - (2s-1)\delta d(v,w) > 0\,.$$

We will show that $\Psi^s(t) \leq \mathcal{G}/\sigma^s$ for each $s \in \mathbb{N}$ and all times t. For $s = \lceil \log_\sigma \mathcal{G}/\delta \rceil$, this yields that

$$L_v(t) - L_w(t) - (2s-1)\delta \leq \mathcal{G}/\sigma^s \leq \delta \quad \Rightarrow \quad L_v(t) - L_w(t) \leq 2\delta\lceil \log_\sigma \mathcal{G}/\delta \rceil\,.$$

The definition of Ψ_v^s is closely related to the slow mode condition **SC**. It makes sure that leading nodes are always in slow mode.

Lemma 2.11 (Leading Lemma)**.** *Suppose $w \in V$ is a leading node at time t. Then w satisfies **SC** and **ST**.*

Proof. As w is a leading node at time t, there are $s \in \mathbb{N}$ and $v \in V$ so that

$$\Psi_v^s(t) = L_w(t) - L_v(t) - (2s-1)\delta d(v,w) > 0\,.$$

In particular, $L_w(t) > L_v(t)$, so $w \neq v$. For any $y \in V$, we have that

$$L_w(t) - L_v(t) - (2s-1)\delta d(v,w) = \Psi_v^s(t) \geq L_y(t) - L_v(t) - (2s-1)\delta d(y,w)\,.$$

Rearranging this yields

$$L_w(t) - L_y(t) \geq (2s-1)\delta(d(v,w) - d(y,w))\,.$$

In particular, for any $y \in N_v$, $d(v,w) \geq d(y,w) - 1$ and hence

$$L_y(t) - L_w(t) \leq (2s-1)\delta\,,$$

i.e., **SC** 2 holds at w. Now consider $x \in N_v$ so that $d(x,w) = d(v,w) - 1$; as $v \neq w$, such a node exists. We get that

$$L_w(t) - L_y(t) \geq (2s-1)\delta\,,$$

showing **SC** 1. By Lemma 2.8, w then also satisfies **ST** at time t. □

This can readily be translated into a bound on the growth of Ψ_w^s whenever it is positive.

Lemma 2.12 (Wait-up Lemma)**.** *Suppose $w \in V$ satisfies $\Psi_w^s(t) > 0$ for all $t \in (t_0, t_1]$. Then*

$$\Psi_w^s(t_1) \leq \Psi_w^s(t_0) - (L_w(t_1) - L_w(t_0)) + \vartheta(t_1 - t_0).$$

Proof. Fix $w \in V$, $s \in \mathbb{N}$ and $(t_0, t_1]$ as in the hypothesis of the lemma. For $v \in V$ and $t \in (t_0, t_1]$, define the function $f_v(t) = L_v(t) - (2s-1)\delta d(v,w)$. Observe that

$$\max_{v \in V}\{f_v(t)\} - L_w(t) = \Psi_w^s(t)\,.$$

Moreover, for any v satisfying $f_v(t) = L_w(t) + \Psi_w^s(t)$, we have that $L_v(t) - L_w(t) - (2s-1)\delta d(v,w) = \Psi_w^s(t) > 0$. Thus, Lemma 2.11 shows that v is in slow mode at time t. As (we assume that) logical clocks are differentiable, so is f_v, and it follows that $\frac{d}{dt} f_v(t) \leq \vartheta$ for any $v \in V$ and time $t \in (t_0, t_1]$ satisfying that $f_v(t) = \max_{x \in V}\{f_x(t)\}$. By Lemma A.1, it follows that $\max_{v \in V}\{f_v(t)\}$ grows at most at rate ϑ:

$$\max_{v \in V}\{f_v(t_1)\} \leq \max_{v \in V}\{f_v(t_0)\} + \vartheta(t_1 - t_0)\,.$$

We conclude that

$$\begin{aligned}\Psi_w^s(t_1) - \Psi_w^s(t_0) &= \max_{v \in V}\{f_v(t_1)\} - L_w(t_1) - (\max_{v \in V}\{f_v(t_0)\} - L_w(t_0)) \\ &\leq -(L_w(t_1) - L_w(t_0)) + \vartheta(t_1 - t_0)\,,\end{aligned}$$

which can be rearranged into the claim of the lemma. □

Trailing Nodes

As $L_w(t_1) - L_w(t_0) \geq t_1 - t_0$ at all times, Lemma 2.15 shows that Ψ^s cannot grow faster than at rate $\vartheta - 1$ when it is positive. This buys us some time, but we need to show that w will make sufficient progress before Ψ^s grows larger than the desired bound. The approach to showing this is very similar to the one for Lemma 2.12, where now we need to exploit the fast mode condition **FC**.

Definition 2.13 (Trailing Nodes)**.** *We say that $w \in V$ is a* trailing node *at time t, if there is some $s \in \mathbb{N}$ and a node v such that*

$$L_v(t) - L_w(t) - 2s\delta d(v,w) = \max_{x \in V}\{L_v(t) - L_x(t) - 2s\delta d(v,x)\} > 0\,.$$

Lemma 2.14 (Trailing Lemma)**.** *Suppose $w \in V$ is a trailing node at time t. Then w satisfies **FC** and **FT**.*

Proof. Let s and v be such that

$$L_v(t) - L_w(t) - 2s\delta d(v,w) = \max_{x \in V}\{L_v(t) - L_x(t) - 2s\delta d(v,x)\} > 0\,.$$

In particular, $L_v(t) > L_w(t)$, implying that $v \neq w$. For $y \in V$, we have that

$$L_v(t) - L_w(t) - 2s\delta d(v,w) \geq L_v(t) - L_y(t) - 2s\delta d(v,y)$$

and thus for all neighbors $y \in N_w$ that

$$L_y(t) - L_w(t) + 2s\delta(d(v,y) - d(v,w)) \geq 0\,.$$

It follows that

$$\forall y \in N_v\colon L_w(t) - L_y(t) \leq 2s\delta\,,$$

i.e., **FC** 2 holds. As $v \neq w$, there is some node $x \in N_v$ with $d(v,x) = d(v,w) - 1$. We obtain that

$$\exists x \in N_v\colon L_y(t) - L_w(t) \geq 2s\delta\,,$$

showing **FC** 1. By Lemma 2.8, w thus also satisfies **FT** at time t. □

Using this, we can show that if $\Psi^s_w(t_0) > 0$, w will eventually catch up. How long this takes can be expressed in terms of $\Psi^{s-1}(t_0)$, or, if $s = 1$, $\mathcal{G}$.

Lemma 2.15 (Catch-up Lemma)**.** *Let $s \in \mathbb{N}$ and t_0, t_1 be times. If $s = 1$, suppose that $t_1 \geq t_0 + \mathcal{G}/\mu$; otherwise, suppose that $t_1 \geq t_0 + \Psi^{s-1}(t_0)/\mu$. Then, for any $w \in V$,*

$$L_w(t_1) - L_w(t_0) \geq t_1 - t_0 + \Psi^s_w(t_0)\,.$$

Proof. Choose $v \in V$ such that

$$\Psi^s_w(t_0) = L_v(t_0) - L_w(t_0) - (2s-1)\delta d(v, w) > 0\,.$$

Define $f_x(t) := L_v(t_0) + (t - t_0) - L_x(t) - (2s-2)\delta d(v, x)$ for $x \in V$ and observe that $\Psi^s_w(t_0) \leq f_w(t_0)$. Hence, if $\max_{x \in V}\{f_x(t)\} \leq 0$ for some $t \in [t_0, t_1]$, then

$$\begin{aligned}
L_w(t_1) - L_w(t) - (t_1 - t) \geq 0 \geq f_w(t)&\\
&= L_v(t_0) + (t - t_0) - L_w(t) - (2s-2)\delta d(v, x)\\
&= f_w(t_0) + (t - t_0) - (L_w(t) - L_w(t_0))\\
&\geq \Psi^s_w(t_0) + (t - t_0) - (L_w(t) - L_w(t_0))\,,
\end{aligned}$$

which can be rearranged into the claim of the lemma.

To show this, consider any time $t \in [t_0, t_1]$ when $\max_{x \in V}\{f_x(t)\} > 0$ and let $y \in V$ be any node such that $\max_{x \in V}\{f_x(t)\} = f_y(t)$. Then y is trailing, as

$$\begin{aligned}
&\max_{x \in V}\{L_v(t) - L_x(t) - (2s-2)\delta d(v, x)\}\\
&= L_v(t) - L_v(t_0) - (t - t_0) + \max_{x \in V}\{f_x(t)\}\\
&= L_v(t) - L_v(t_0) - (t - t_0) + f_y(t)\\
&= L_v(t) - L_y(t) - (2s-2)\delta d(v, y)
\end{aligned}$$

and

$$L_v(t) - L_v(t_0) - (t - t_0) + \max_{x \in V}\{f_x(t)\} > L_v(t) - L_v(t_0) - (t - t_0) \geq 0\,.$$

Thus, by Lemma 2.14 y is in fast mode. As logical clocks are (assumed to be) differentiable, we get that $\frac{d}{dt} f_y(t) = 1 - l_y(t) \leq -\mu$.

Now assume for contradiction that $\max_{x \in V}\{f_x(t)\} > 0$ for all $t \in [t_0, t_1]$. Then, applying Lemma A.1 again, we conclude that

$$\max_{x \in V}\{f_x(t_0)\} > -(\max_{x \in V}\{f_x(t_1)\} - \max_{x \in V}\{f_x(t_0)\}) \geq \mu(t_1 - t_0)\,.$$

If $s = 1$, $\mu(t_1 - t_0) \geq \mathcal{G}$, contradicting the fact that

$$f_x(t_0) = L_v(t_0) - L_x(t_0) \leq \mathcal{G}$$

for all $x \in V$. If $s > 1$, then $\mu(t_1 - t_0) \geq \Psi^{s-1}(t_0)$. However, we have that

$$f_x(t_0) \leq L_v(t_0) - L_x(t_0) - (2s-3)\delta d(v, x) \leq \Psi^{s-1}(t_0)$$

for all $x \in V$. As this is a contradiction as well, the claim of the lemma follows. $\square$

Putting Things Together

Theorem 2.16. *Assume that $H_v(0) - H_w(0) \leq \delta$ for all $\{v, w\} \in E$. Then, for all $s \in \mathbb{N}$, Algorithm 2.1 guarantees $\Psi^s(t) \leq \mathcal{G}/\sigma^s$, where $\sigma = \mu/(1-\vartheta)$.*

Proof. Suppose for contradiction that the statement of the theorem is false. Let $s \in \mathbb{N}$ be minimal such that there is a time t_1 for which $\Psi^s(t_1) = \mathcal{G}/\sigma^s + \varepsilon$ for some $\varepsilon > 0$. Thus, there is some $w \in V$ such that

$$\Psi^s_w(t_1) = \Psi^s(t_1) = \frac{\mathcal{G}}{\sigma^s} + \varepsilon\,.$$

Set $t_0 := \max\{t - \mathcal{G}/(\mu\sigma^{s-1}), 0\}$. Consider the time $t' \in [t_0, t_1]$ that is minimal with the property that $\Psi^s_w(t) > 0$ for all $t \in (t', t_1]$ (by continuity of Ψ^s_w such a time exists). Thus, we can apply Lemma 2.12 to this interval, yielding that

$$\Psi^s_w(t_1) \leq \Psi^s_w(t') + \vartheta(t_1 - t') - (L_w(t_1) - L_w(t')) \leq \Psi^s_w(t') + (\vartheta - 1)(t_1 - t')\,.$$

$\Psi^s_w(t')$ cannot be 0, as otherwise

$$\Psi^s_w(t_1) \leq (\vartheta - 1)(t_1 - t') \leq \frac{(\vartheta-1)}{\mu} \cdot \frac{\mathcal{G}}{\sigma^{s-1}} = \frac{\mathcal{G}}{\sigma^s}\,,$$

contradicting $\Psi^s_w(t_1) = \mathcal{G}/\sigma^s + \varepsilon$.

On the other hand, if $\Psi^s_w(t') > 0$, we must have $t' = t_0$ from the definition of t', and $t_0 \neq 0$ because

$$\begin{aligned}
&\max_{v,w \in V}\{L_v(0) - L_w(0) - (2s-1)\delta d(v,w)\}\\
=\ &\max_{v,w \in V}\{H_v(0) - H_w(0) - (2s-1)\delta d(v,w)\}\\
\leq\ &\max_{v,w \in V}\{H_v(0) - H_w(0) - \delta d(v,w)\} \leq 0\,,
\end{aligned}$$

as $H_v(0) - H_w(0) \leq \delta$ for all neighbors v, w by assumption. Hence, $t' = t_0 = t_1 - \mathcal{G}/(\mu\sigma^{s-1})$. If $s > 1$, the minimality of s yields that $\Psi^s(t_0) \leq \mathcal{G}/\sigma^{s-1}$. We apply Lemma 2.15 to level s, node w, and time $t' = t_0$, yielding that

$$\Psi^s_w(t_1) \leq \Psi^s_w(t_0) + \vartheta(t_1 - t_0) - (L_w(t_1) - L_w(t_0)) \leq (\vartheta - 1)(t_1 - t_0) \leq \frac{\mathcal{G}}{\sigma^s}\,,$$

again contradicting $\Psi^s_w(t_1) = \mathcal{G}/\sigma^s + \varepsilon$. Reaching a contradiction in all cases, we conclude that the statement of the theorem must indeed hold. □

Our main result, Theorem 2.9, is now immediate.

Proof of Theorem 2.9. We apply Theorem 2.16 and consider $s := \lceil \log_\sigma(\mathcal{G}/\delta)\rceil$. For any $\{v, w\} \in E$ and any time t, we thus have that

$$L_v(t) - L_w(t) - (2s-1)\delta = L_v(t) - L_w(t) - (2s-1)\delta d(v,w) \leq \Psi^s(t) \leq \frac{\mathcal{G}}{\sigma^s} \leq \delta\,.$$

Rearranging this and exchanging the roles of v and w, we obtain

$$\mathcal{L}(t) = \max_{\{v,w\} \in E}\{|L_v(t) - L_w(t)|\} \leq 2s\delta = 2\delta\lceil \log_\sigma(\mathcal{G}/\delta)\rceil\,.$$ □

What to Take Home

- A very simple algorithm achieves a surprisingly good local skew, even if clocks must advance at all times.
- The base of the logarithm in the bound is typically large. A cheap quartz oscillator guarantees $\vartheta - 1 \leq 10^{-5}$, while typically $u/d \geq 10^{-2}$. With a base of roughly 10^3, the logarithmic term usually remains quite small.
- The algorithmic idea is surprisingly versatile. It works if δ is different for each link, and with some modifications (to algorithm and analysis), adversarial changes in the graph can be handled.

Bibliographic Notes

Gradient clock synchronization was introduced by Fan and Lynch [FL06], who show a lower bound of $\Omega(\log(uD)/\log\log(uD))$ on the local skew. Some researchers found this result rather counter-intuitive, and it triggered a line of research seeking to resolve the question what precisely can be achieved. The first non-trivial upper bound was provided by Locher and Wattenhofer [LW06]. Their *blocking algorithm* bounds the local skew by $\mathcal{O}(\sqrt{\delta D})$. The first logarithmic bound on the local skew was given in [LLW08] and soon after improved to the algorithm presented here [LLW10]. However, the elegant way of phrasing it in terms of the fast and slow modes and conditions is due to Kuhn and Oshman [KO09].

The algorithmic idea underlying the presented solution turns out to be surprisingly robust and versatile. Essentially the same algorithm works for different uncertainties on the edges [KO09]. With a suitable method of carefully incorporating newly appearing edges, it can handle dynamic graphs [KLLO10] (this problem is introduced in [KLO11]), in the sense that edges that were continuously present for sufficiently long satisfy the respective guarantee on the skew between their endpoints. Recently, the approach has been independently discovered (twice!) for solving load balancing tasks that arise in certain packet routing problems [DLNO17, PR17].

Bibliography

[DLNO17] Stefan Dobrev, Manuel Lafond, Lata Narayanan, and Jaroslav Opatrny. Optimal local buffer management for information gathering with adversarial traffic. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017, Washington DC, USA, July 24-26, 2017*, pages 265–274, 2017.

[FL06] Rui Fan and Nancy Lynch. Gradient Clock Synchronization. *Distributed Computing*, 18(4):255–266, 2006.

[KLLO10] Fabian Kuhn, Christoph Lenzen, Thomas Locher, and Rotem Oshman. Optimal Gradient Clock Synchronization in Dynamic Networks. *CoRR*, abs/1005.2894, 2010.

[KLO11] Fabian Kuhn, Thomas Locher, and Rotem Oshman. Gradient Clock Synchronization in Dynamic Networks. *Theory Comput. Syst.*, 49(4):781–816, 2011.

[KO09] Fabian Kuhn and Rotem Oshman. Gradient Clock Synchronization Using Reference Broadcasts. In *Proc. 13th Conference on Principles of Distributed Systems (OPODIS)*, pages 204–218, 2009.

[LLW08] Christoph Lenzen, Thomas Locher, and Roger Wattenhofer. Clock Synchronization with Bounded Global and Local Skew. In *Proc. 49th Symposium on Foundations of Computer Science (FOCS)*, pages 509–518, 2008.

[LLW10] Christoph Lenzen, Thomas Locher, and Roger Wattenhofer. Tight Bounds for Clock Synchronization. *J. ACM*, 57(2):8:1–8:42, 2010.

[LW06] Thomas Locher and Roger Wattenhofer. Oblivious Gradient Clock Synchronization. In *Proc. 20th Symposium on Distributed Computing (DISC)*, pages 520–533, 2006.

[PR17] Boaz Patt-Shamir and Will Rosenbaum. The space requirement of local forwarding on acyclic networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*, pages 13–22, 2017.