13 Self-Stabilizing Lynch-Welch Algorithm

Chapter Contents

13.1 Overview

In previous chapters we considered having a bound on the ratio of faulty nodes ($n > 3f$). In Chapter 12 we extended our fault model, so far comprising only permanently damaged (Byzantine) nodes, to include transient faults. We studied two algorithms: one self-stabilizing, but unable to withstand a single permanent fault; the other resilient to $f < n/3$ Byzantine faults (whether transient or not), but unable to recover if there are ever more than f faults at the same time. So, is it possible to recover after facing more than f concurrent faults?

Overcoming such a case, and allowing for having up to f permanent faults, requires a different approach. In the current chapter, we present such an approach, albeit with a caveat: we assume that there is an underlying algorithm that eventually enables us to obtain *some* synchronization among the correct nodes. In later chapters, we will present algorithms providing such rough synchronization without additional assumptions. Given such an underlying algorithm, we present Algorithm 17, which converges from an arbitrary initial state—the result of transient faults—to a globally consistent state, despite any $f < n/3$ nodes remaining faulty or failing again.

Recall that the Lynch-Welch algorithm achieves asymptotically optimal skew, tolerates the maximum possible number of $\lceil n/3 \rceil - 1$ Byzantine faults, and is straightforward to implement. As the algorithm we construct is going to be a variant of the Lynch-Welch algorithm from Chapter 10, we would add self-stabilization to this list, resulting in an extremely robust synchronization primitive!

Alas, we will not get self-stabilization largely "for free," as with the GCS algorithm. The Lynch-Welch algorithm relies on some initial degree of synchronization to maintain the abstraction of rounds it uses. It is simulating synchronous execution, but self-stabilization requires that we can deal with a complete (initial) lack of synchrony! It turns out that this is an incredibly hard problem, and we will take one step at a time. Our first step is to reduce the task to finding an (efficient) self-stabilizing solution to pulse synchronization with a much weaker bound on the skew, which we perform in this chapter.

The execution suffixes meeting our task specification for the Lynch-Welch algorithm (Algorithm 11) are easily identified: There should be a time t from which on the algorithm behaves just like expected, i.e., as if it was initialized at this time and thus exhibits the skew and period bounds from Theorem 10.9. We will address the challenge to reliably achieve this behavior in stages, which showcase the tools we use to achieve the end result presented in Algorithm 17.

13.2 First Attempt: Reset on Heartbeats

In the following, we assume that we already have an underlying self-stabilizing pulse synchronization algorithm with skew σ_h in place (such an algorithm is presented later in the book) Thus, as we assume, there is some time t when it stabilized from which on the correct nodes generate pulses $h_{v,i}$, $v \in V_g$, $i \in \mathbb{N}$, satisfying that $\max_{i \in \mathbb{N}, v, w \in V_g} \{|h_{v,i} - h_{w,i}|\} \leq \sigma_h$. Moreover, we have lower and upper bounds on the time between pulses. Observe that the index $i \in \mathbb{N}$ is counted after time t as a notational reference. Nodes are not aware of the specific index and will not exchange such an index. We will refer to these pulses as *heartbeats,* or simply beats. They are supposed to be fairly slow in comparison to the pulses of the (modified) Lynch-Welch algorithm. From here on, when we refer to pulses, we will refer to those of the Lynch-Welch algorithm only.

There is a single hurdle keeping the Lynch-Welch algorithm from being self-stabilizing: the need for a (known) bound on the initial deviation between correct nodes' local times. Recall that the bound is being used explicitly in Algorithm 11. The heartbeats provide exactly that—they are at most σ_h apart from each other. So we could simply reset the Lynch-Welch algorithm on every

heartbeat, setting $\mathcal{S} := \sigma_h$ for the initialization of the algorithm. This is going to work splendidly, as we will not even have to change the analysis—until the next beat comes along and messes things up.

E13.1 Spend a few moments to think about the issues that could be caused by the interference of (unsynchronized!) heartbeats.

As the beats are not as well-synchronized as the Lynch-Welch pulses (otherwise we would not go through all this trouble), the reset will destroy the better synchronization guarantee again. Even worse, it may interrupt the Lynch-Welch algorithm generating a pulse!

Remark 13.1. *Actually, one needs to be slightly more careful when resetting, in that any messages sent by a node just before it is reset by its beat should not be confused with his "round* 1*"-message following the reset. This is easily addressed by offsetting the first round by* ϑd *local time compared to* $h_{v,i}$*, or by using the last message received during the time window in which receivers listen for messages from other nodes.*

13.3 Second Attempt: Adding Feedback

The naive solution does not work, because heartbeats may arrive at inconvenient times. However, the "first" beat (in our analysis) establishes a timing relation between the Lynch-Welch instance and the instance of the self-stabilizing pulse synchronization algorithm generating the beats. If we add the additional requirement that the pulse synchronization algorithm generating the heartbeats accepts some external input that can shift the time when the next beat occurs (within certain bounds), we could align them with the pulses generated by the Lynch-Welch instance.

More specifically, after a (suitably chosen) fixed number of Lynch-Welch pulses, nodes will issue a NEXT signal to their local instance of the pulse algorithm generating the heartbeats. Thus, the beat generation mechanism needs only be "responsive" to the NEXT signal within a specific time window in relation to the previous beat. Under some mild conditions on ϑ, this will turn out to be a fairly harmless constraint. We use this to trigger the next beat, aligned up to $O(\sigma_h + \mathcal{S})$ time with when the nodes issue the NEXT signals (where $\mathcal{S}$ is the skew of the Lynch-Welch algorithm). This can be kept within a single round of the Lynch-Welch algorithm (without affecting more than constants), as both $\sigma_h \in O(d)$ and $\mathcal{S} \in O(d)$, and the round duration of the Lynch-Welch algorithm $T \in \Omega(d)$ anyway.

E13.2 Spend a few moments to think about whether there any issues remain for the revised approach.

Is this good enough? Not yet, as resets may cause large skew every time — unless, in addition, we require that well-synchronized NEXT signals result in an equally well-synchronized heartbeat. Instead of adding even more constraints on the self-stabilizing algorithm (not knowing whether they can be satisfied), we use a different approach.

13.4 Third Attempt: Reset on Unexpected Heartbeats Only

The final adjustment is to *not* perform a reset when a beat arrives on schedule, i.e., within a time window of size $O(\sigma_h + \mathcal{S})$ around the point when it would occur in a world of perfect synchrony. The size of this window is chosen such that once the heartbeat generation has stabilized, after the first "proper" heartbeat, no resets are triggered at correct nodes any more. Yet, the reset mechanism still guarantees that a heartbeat will enforce synchronization up to a skew of $O(\sigma_h + \mathcal{S})$: either a node is not reset, defining an $O(\sigma_h + \mathcal{S})$-sized window in which the iteration of the Lynch-Welch algorithm's loop is started, or it is, aligning the timing of the node with this window.

It remains to formalize this approach and prove it correct. W.l.o.g., we assume in the following that the heartbeats stabilized by time 0, and start to reason from there. (Note, however, that there still might be arbitrary messages in transit at time 0!). Let us first specify our expectations on the feedback mechanism.

Definition 13.2 (Feedback Mechanism)**.** *Nodes $v \in V_g$ generate* beats *at times $h_{v,i} \in \mathbb{R}$, $i \in \mathbb{N}$, such that for parameters $0 < B_1 < B_2 < B_3 \in \mathbb{R}$ and σ_h (a skew bound) the following properties hold, for all $i \in \mathbb{N}$.*

1. *For all $v, w \in V_g$, we have that $|h_{v,i} - h_{w,i}| \leq \sigma_h$.*
2. *If no $v \in V_g$ triggers its NEXT signal during $[\min_{w \in V_g}\{h_{w,i}\} + B_1, t]$ for some $t < \min_{w \in V_g}\{h_{w,i}\} + B_3$, then $\min_{w \in V_g}\{h_{w,i+1}\} > t$.*
3. *If all $v \in V_g$ trigger their NEXT signals during $[\min_{w \in V_g}\{h_{w,i}\} + B_2, t]$ for some $t \leq \min_{w \in V_g}\{h_{w,i}\} + B_3$, then $\max_{w \in V_g}\{h_{w,i+1}\} \leq t + \sigma_h$.*

B_1, B_2, and B_3 and the skew bound σ_h cannot be chosen arbitrarily for our approach to work. We will determine sufficient constraints from the analysis.

In order to describe the algorithm, we assume that each node is running an instance of the Lynch-Welch Algorithm, the beat generation algorithm, and some additional high-level control we present now. We restate the (loop of the) Lynch-Welch Algorithm 11 in Algorithm 16.

Algorithm 16 The loop of Algorithm 11, which is run alongside the local instances of the beat generation algorithm and Algorithm 17. Note that Algorithm 17 may reset the loop for stabilization purposes.

1: **while** true **do**
2: generate pulse // assume that $r \in \mathbb{N}$ is the pulse index
3: $h \leftarrow \text{getH}()$
4: wait until $\text{getH}() = h + \vartheta\mathcal{S}$ // all nodes are in round r
5: broadcast empty message to all nodes (including self)
6: wait until $\text{getH}() = h + (\vartheta^2 + \vartheta)\mathcal{S} + \vartheta d$
// denote this time by $\tau_{v,r}$
// correct nodes' messages should have arrived
7: **for** each node $w \in V$ **do**
8: compute $\Delta(w) \in [p_{w,r} - p_{v,r}, p_{w,r} - p_{v,r} + \delta]$
// denote $p_r := \max_{w \in V_g}\{p_{w,r}\}$
9: **end for**
10: $U \leftarrow \{\Delta(w) \mid w \in V\}$ (as multiset, i.e., values may repeat)
11: $\Delta \leftarrow \left(U^{(f+1)} + U^{(n-f)}\right)/2$
12: wait until $\text{getH}() = h + \Delta + T$
13: **end while**

E13.3 Take a look at Algorithm 16. Why is it ok that we removed the first line of Algorithm 11?

E13.4 Can we do the same in Algorithm 12 from Chapter 12?

The high-level control may (re-)initialize the instance of Algorithm 16, which is described in the subroutine `reset(`τ`)` it may call. It has a few parameters:

M: The pulses of Algorithm 16 are counted modulo M. Every M pulses, a heartbeat is expected, which is checked by Algorithm 17.

R^-: If a beat arrives at time t and the pulse number is 0 mod M, it should take at least R^- local time before the node generates the next pulse. Instead of trying to compute upfront when Algorithm 16 would generate a pulse, we simply "catch" the event and perform a reset if the pulse would be generated too early.

R^+: This is the matching upper bound, i.e., under the same conditions, it should take at most R^+ local time before the node generates the next pulse.

Algorithm 17 Interface algorithm, actions for node $v \in V_g$ in response to a local event at time t. Runs in parallel to local instances of the beat generation algorithm and Algorithm 16.

```
1:  ▷ The algorithm maintains local variable i ∈ [M]
2:  if v generates a pulse at time t then
3:      i := i + 1 mod M
4:      if i = 0 then
5:          wait until local time H_v(t) + ϑS(M)
6:          trigger NEXT signal
7:      end if
8:  end if
9:  if v generates a beat at time t then
10:     if i ≠ 0 then
                ▷ beats should align with every M^th pulse, hence reset
11:         reset(R^+)
12:     else if Algorithm 16 requires generating a pulse before H_v(t)+R^- then
13:                 ▷ reset at pulse time t' to avoid early pulse or message
14:         reset(R^+ − (H_v(t') − H_v(t))), where t' is the current time
15:     else if next pulse is not generated by local time H_v(t) + R^+ then
16:                                         ▷ reset to avoid late pulse and
17:                             ▷ start listening for other nodes' pulses on time
18:         reset(0)
19:     end if
20: end if
21: Function(reset(τ))
22: stop local instance of Algorithm 16
23: wait for τ local time
24: i := 0
25: initialize a new local instance of Algorithm 16
```

$\mathcal{S}(r)$: This denotes the skew bound guaranteed by Algorithm 16 for pulse $1 < r \in \{1, \dots, M\}$, provided the algorithm is initialized with skew $\mathcal{S}(1) := \mathcal{S}$. Algorithm 17 needs to make use of $\mathcal{S}(1)$ and $\mathcal{S}(M)$ only.

Algorithm 17 triggers the NEXT signal $\vartheta\mathcal{S}(M)$ local time after generating a beat (i.e., at the earliest time when certainly all nodes have generated the beat), and checks whether pulse 1 modulo M occurs between R^- and R^+ local time after the beat (which necessitates that the beat occurs after pulse 0 modulo M). If this is not the case, the algorithm generates a pulse and restarts the loop of Algorithm 16 exactly R^+ local time after the beat was generated. Moreover, it

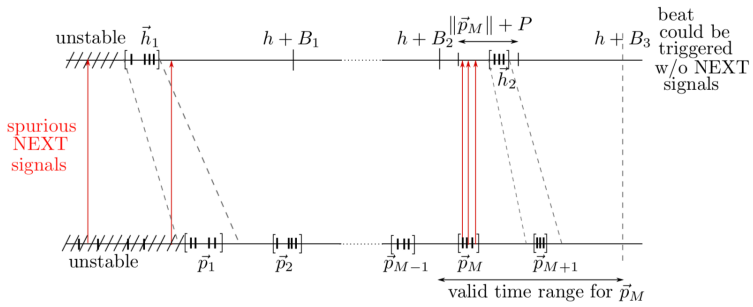

Figure 13.1
Interaction of the beat generation and Algorithm 16 in the stabilization process, controlled by Algorithm 17. Beat $\vec{h}_1$ forces pulse $\vec{p}_1$ to be roughly synchronized. The approximate agreement steps then result in tightly synchronized pulses. By the time the nodes trigger beat $\vec{h}_2$ by providing NEXT signals based on $\vec{p}_M$, synchronization is tight enough to guarantee that the beat results in no resets.

ensures that no other pulse is generated between the beat and then. Figure 13.1 illustrates how the control algorithm ensures stabilization.

This properly "initializes" Algorithm 16 with skew $\mathcal{S} := \mathcal{S}(1) := R^+ + \sigma_h - R^-/\vartheta$, which then ensures that the skew has been reduced to $\mathcal{S}(M)$ by the time the next beat is due. By choosing all parameters right, we ensure that the M^{th} pulse (after stabilization) falls in the time window provided by Definition 13.2 for making use of the NEXT signals, which then triggers the next beat such that no $v \in V_g$ performs another abrupt reset. From there, inductive reasoning shows that no $v \in V_g$ ever performs an abrupt reset again (so as long as there are no more transient faults), and the analysis of Algorithm 11 from Chapter 10 yields a bound on the skew achieved. Figure 13.2 illustrates how the nodes locally check whether they should perform a reset or not.

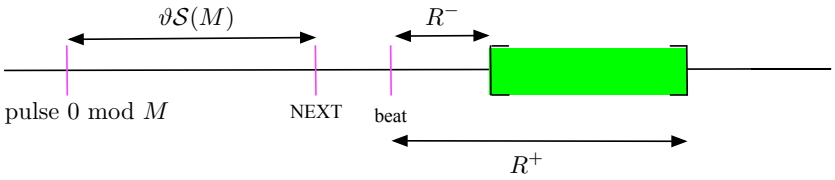

Figure 13.2
After M pulses a node v waits for $\mathcal{S}(M)$ local time and then generates the NEXT signal. After stabilization, the next heartbeat occurs shortly after. If the next pulse (which is going to be generated by the Lynch-Welch algorithm), with number $i = 1$, is not going to be generated at least R^- and at most R^+ local time after the heartbeat (i.e., within the green box), the node resets the Lynch-Welch algorithm, restarting its loop R^+ local time after the beat.

13.5 Analysis of The Self-Stabilizing Lynch-Welch Algorithm

In the following, we assume that in Algorithm 16, estimates are computed according to Lemma 10.8 (yielding $\delta = u + (\vartheta - 1)d + (\vartheta^2 + \vartheta - 2)\mathcal{S}$), $7 - 6\vartheta^2 > 0$, and set $T = (\vartheta^2 + \vartheta + 1)\mathcal{S} + \vartheta d$. Under certain constraints, we then can show analogously to Theorem 10.9 that all pulses will have skew at most $\mathcal{S}$. However, we will in fact use that with each pulse, we get a stronger bound on the skew, i.e., $\mathcal{S} = \mathcal{S}(1) \geq \mathcal{S}(2) \geq \ldots \geq \mathcal{S}(M)$, which follows from a minor adjustment to the analysis, see Corollary 13.4.

For the outlined approach to work, in addition to the above, the following constraints need to be satisfied; note that we will not mention explicitly using the first inequality, but it is certainly necessary for the algorithm to operate as

intended.

$$R^+ \geq R^- \tag{13.1}$$
$$\mathcal{S} = R^+ + \sigma_h - R^-/\vartheta \tag{13.2}$$
$$\mathcal{S} \geq \frac{2(2\vartheta - 1)\delta + 2(\vartheta - 1)T}{2 - \vartheta} \tag{13.3}$$
$$\frac{R^-}{\vartheta} \geq \sigma_h + \vartheta\mathcal{S} + d \tag{13.4}$$
$$\frac{B_2}{\vartheta} > \sigma_h + R^+ + T + 3\mathcal{S} \tag{13.5}$$
$$B_1 > \sigma_h + R^+ \tag{13.6}$$
$$B_3 > R^+ + (M-1)(T + 3\mathcal{S}) + (\vartheta + 1)\mathcal{S}(M) + \sigma_h \tag{13.7}$$
$$B_2 \leq \frac{R^-}{\vartheta} + (M-1)\left(\frac{T - (\vartheta + 1)\mathcal{S}}{\vartheta}\right) + \mathcal{S}(M) \tag{13.8}$$
$$\frac{R^+}{\vartheta} \geq (\vartheta + 1)\mathcal{S}(M) + \sigma_h \tag{13.9}$$
$$\mathcal{S}(M) < \frac{\vartheta\mathcal{S} - \sigma_h}{\vartheta + 1} \tag{13.10}$$

We will worry about satisfying all of these constraints later. For now, we assume that they hold; what follows is conditional on this assumption.

We first establish that the first beat guarantees to "initialize" the synchronization algorithm such that it will run correctly from this point on (neglecting for the moment the possible intervention by further beats). We use this do define the "first" pulse times $p_{v,1}$, $v \in V_g$, as well; we enumerate consecutive pulses accordingly.

Lemma 13.3. *Let $h := \min_{v \in V_g}\{h_{v,1}\}$. We have that*

1. *Each $v \in V_g$ generates a pulse at a unique time $p_{v,1} \in [h + R^-/\vartheta, h + \sigma_h + R^+]$.*
2. *$\|\vec{p}(1)\| \leq \mathcal{S}$.*
3. *At time $p_{v,1}$, $v \in V_g$ sets $i := 1$.*
4. *At the time $\min_{v \in V_g}\{p_{v,1}\}$, no message (of Algorithm 16) sent by node $v \in V_g$ before time $p_{v,1}$ is in transit any more.*

Proof. Assume for the moment that $\min_{v \in V_g}\{h_{v,2}\}$ is sufficiently large, i.e., no second beat will occur at any correct node for the times relevant to the proof of the lemma; we will verify this at the end of the proof.

From the pseudocode given in Algorithm 17, it is straightforward to verify that $v \in V_g$ generates a pulse at a local time from $[H_v(h_{v,1}) + R^-, H_v(h_{v,1}) + R^+]$, and does not generate a pulse at a local time from $[H_v(h_{v,1}), H_v(h_{v,1}) + R^-)$.

Denote by $p_{v,1}$ the time when $v \in V_g$ generates its first pulse after time $h_{v,1}$. Following Algorithm 16, no $v \in V_g$ will send a message or generate another pulse during $(p_{v,1}, p_{v,1} + \mathcal{S})$, where

$$\begin{aligned} p_{v,1} + \mathcal{S} &\geq h_{v,1} + R^-/\vartheta + \mathcal{S} \\ &= h + \sigma_h + R^+. \end{aligned} \tag{13.2}$$

Because $h_{v,1} \in [h, h + \sigma_h]$ for all $v \in V_g$ by Definition 13.2, this implies that for each $v \in V_g$, $p_{v,1} \in [h + R^-/\vartheta, h + \sigma_h + R^+]$ and $p_{v,1}$ is indeed unique. The second claim is now immediate from Definition 13.2 and Equation (13.2).

Concerning the third claim, observe that if at time $h_{v,1}$ it held that the i-variable of $v \in V_g$ was not 0, it was set to 0. Thus, when v generates its next pulse at time $p_{v,1}$, it is increased to 1. Concerning the final claim, we have established that $v \in V$ generates no pulse during $[h + \sigma_h, h + R^-/\vartheta)$; thus, it sends no message during $[h + \sigma_h + \vartheta\mathcal{S}, h + R^-/\vartheta)$ (cf. Algorithm 16), and Inequality (13.4) ensures that no message of $v \in V_g$ sent before time $h_{v,1}$ is in transit any more at time $p_{w,1}$ for any $w \in V_g$.

It remains to show that indeed $\min_{v \in V_g}\{h_{v,2}\}$ is sufficiently large to not interfere with the above reasoning. Clearly, this is the case if round 1 ends at all nodes before this time. Let H be the infimum[8] of times t such that some $v \in V_g$ executes `reset` at a time $t > p_{v,1}$. Clearly, $H \geq \min_{v \in V_g}\{h_{v,2}\}$.

By Definition 13.2 and Inequality (13.5) (and $\vartheta > 1$), we can conclude that $H \geq h + B_2 \geq h + \sigma_h + R^+ + T + 3\mathcal{S}$. All parts of the statements of this lemma that refer to times smaller than H hold. As $H > h + \sigma_h + R^+$, this implies that Algorithm 16 behaves exactly as if it was initialized with skew $\mathcal{S}$ at time $h + R^-/\vartheta$. We can thus apply all results from Chapter 10 (for times $t < H$) accordingly. In particular, we get the same results as in Theorem 10.9 (as Inequality (13.3) and our choice of T and δ make sure that we can apply all lemmas), yielding that

$$\begin{aligned} \max_{v \in V_g}\{p_{v,2}\} &\leq \min_{v \in V_g}\{p_{v,1}\} + P_{\max} \\ &\leq h + \sigma_h + R^+ + T + 3\mathcal{S} \\ &< H. \end{aligned}$$

□

Lemma 13.3 serves as induction anchor for the argument showing that all rounds of the algorithm are executed correctly. Let H be defined as in the

[8] This means "smallest upper bound," which is defined also if there is no minimum. In particular, we will see that $H = \infty$, as no such time exists.

previous proof. From the results in Chapter 10, we can bound $\mathcal{S}(r)$ for rounds $r \in \mathbb{N}$ that are complete before time H, by tightening one inequality in the proof of Lemma 10.7.

Corollary 13.4. *Suppose for $r \in \mathbb{N}$ that $\max_{v \in V_g}\{p_{v,r}\} < H$. Then*

$$\begin{aligned}\|\vec{p}_r\| &\leq \mathcal{S}(r)\\ &\coloneqq \frac{\mathcal{S}}{2^{r-1}} + \left(2 - \frac{1}{2^{r-2}}\right)\left(\delta + \left(1 - \frac{1}{\vartheta}\right)(T + \mathcal{S} + \delta)\right)\\ &= \frac{\mathcal{S}}{2^{r-1}} + O(u + (\vartheta - 1)(\mathcal{S} + d)).\end{aligned}$$

Moreover, the generated pulses satisfy $P_{\min} \geq (T - (\vartheta + 1)\mathcal{S})/\vartheta$ and $P_{\max} \leq T + 3\mathcal{S}$.

Proof. Since for each $r \in \mathbb{N}_{>0}$ with $\lambda = 1/2^{r-1} \in (0, 1)$ we have that

(13.2)
$$\begin{aligned}&\vartheta\mathcal{S} \geq 2(2\vartheta - 1)\delta + 2(\vartheta - 1)(T + \mathcal{S})\\ \Leftrightarrow\ &\mathcal{S} \geq 2\delta + 2\left(1 - \frac{1}{\vartheta}\right)(T + \mathcal{S} + \delta)\\ \Leftrightarrow\ &\mathcal{S} \geq \lambda\mathcal{S} + (1 - \lambda)2\left(\delta + \left(1 - \frac{1}{\vartheta}\right)(T + \mathcal{S} + \delta)\right) = \mathcal{S}(r),\end{aligned}$$

convex combination

we have that $\mathcal{S} \geq \mathcal{S}(r)$ for each r. We reason analogously as in Chapter 10 for obtaining Theorem 10.9, but in Lemma 10.7 we do not bound $\|\vec{p}_r\|/2$ by $\mathcal{S}/2$ when bounding $\|\vec{p}_{r+1}\|$, but rather by $\mathcal{S}(r)/2$; since this bound is stronger the induction is otherwise unaffected. In the induction step, inserting the stronger induction hypothesis yields

$$\begin{aligned}\|\vec{p}_{r+1}\| &\leq \frac{\mathcal{S}(r)}{2} + \delta + \left(1 - \frac{1}{\vartheta}\right)(T + \mathcal{S} + \delta)\\ &= \frac{\mathcal{S}}{2 \cdot 2^{r-1}} + \left(2 - \frac{1}{2 \cdot 2^{r-2}}\right)\left(\delta + \left(1 - \frac{1}{\vartheta}\right)(T + \mathcal{S} + \delta)\right)\\ &= \mathcal{S}(r + 1),\end{aligned}$$

i.e., the stronger bound indeed holds. Since $\delta = O(u + (\vartheta - 1)(d + \mathcal{S}))$ and $T = O(\mathcal{S} + d)$, it follows that $\|\vec{p}_r\| = \mathcal{S}/2^{r-1} + O(u + (\vartheta - 1)(\mathcal{S} + d))$ for all $r \in \mathbb{N}_{>0}$. □

In other words, all we need to show is that $H = \infty$, i.e., no further resets occur after the first beat. In fact, it suffices to show this for the second beat, as this constitutes the necessary induction step. To this end, we first show that the NEXT signals occur within the "window of opportunity" provided by Definition 13.2.

Lemma 13.5. *For all $v \in V_g$, it holds that $h_{v,2} \in (p_{v,M} + \mathcal{S}(M), p_{v,M} + (\vartheta + 1)\mathcal{S}(M) + \sigma_h]$. In particular, no node calls the* `reset` *subroutine due to its second beat.*

Proof. Checking Algorithm 17 (and noting that by Lemma 13.3 we have that i is set to 1 at time $p_{v,1}$), we see that after time $p_{v,1}$, $v \in V_g$ will not locally trigger a NEXT signal before either time $p_{v,M} + \mathcal{S}(M)$ or H. Denote $p := \min_{v \in V_g}\{p_{v,M}\}$. As Lemma 13.3 and Inequality (13.6) show that $\max_{v \in V_g}\{p_{v,1}\} \leq h + \sigma_h + R^+ \leq h + B_1$, no NEXT signal is triggered during $[h + B_1, \min\{p + \mathcal{S}(M), H\}]$. However, by Definition 13.2, in absence of any NEXT signal, $h' := \min_{v \in V_g}\{h_{v,2}\}$ satisfies $h' \geq h + B_3$, implying that no NEXT signal is triggered during $[h + B_1, \min\{p + \mathcal{S}(M), h + B_3\}]$. By Definition 13.2, this entails that $H \geq h' \geq \min\{p + \mathcal{S}(M), h + B_3\}$, where equality can hold only if $h' = h + B_3$.

Next, we show that $h' < h + B_3$. Assuming the contrary, we have that $H \geq h' \geq h + B_3$, and get from Lemma 13.3 and Corollary 13.4 that

$$\begin{aligned}
&p + (\vartheta + 1)\mathcal{S}(M) + \sigma_h \\
&\leq \min\left\{\max_{v \in V_g}\{p_{v,1}\} + (M-1)(T + 3\mathcal{S}) + (\vartheta + 1)\mathcal{S}(M) + \sigma_h, H\right\} \\
&\leq \min\left\{h + \sigma_h + R^+ + (M-1)(T + 3\mathcal{S}) + (\vartheta + 1)\mathcal{S}(M) + \sigma_h, h + B_3\right\} \\
&< h + B_3\,,
\end{aligned}$$

where the last step uses Inequality (13.7). Thus, as H is larger than this time, each $v \in V_g$ triggers its NEXT signal before time $h + B_3 - \sigma_h$, because the corollary also shows that $\max_{v \in V_g}\{p_{v,M}\} \leq p + \mathcal{S}(M)$, and nodes wait for $\vartheta\mathcal{S}(M)$ local time before triggering the signal. On the other hand, Lemma 13.3, Corollary 13.4, and Inequality (13.8) show that

$$\begin{aligned}
p + \mathcal{S}(M) &\geq \min_{v \in V_g}\{p_{v,1}\} + (M-1)\frac{T - (\vartheta + 1)\mathcal{S}}{\vartheta} + \mathcal{S}(M) \\
&\geq h + \frac{R^-}{\vartheta} + (M-1)\frac{T - (\vartheta + 1)\mathcal{S}}{\vartheta} + \mathcal{S}(M) \\
&\geq h + B_2\,,
\end{aligned}$$

i.e., all of these NEXT signals are triggered no earlier than time $h + B_2$. By Definition 13.2, this entails that $h' \leq p + (\vartheta + 1)\mathcal{S}(M) + \sigma_h < h + B_3$, contradicting the assumption that $h' \geq h + B_3$.

Knowing that $h' < h + B_3$, we can conclude that $\max_{v \in V_g}\{p_{v,M}\} \leq p + \mathcal{S}(M) < h' \leq H$. As we can derive the same bounds as above, we also get that $\max_{v \in V_g}\{h_{v,2}\} \leq p + (\vartheta + 1)\mathcal{S}(M) + \sigma_h = \min_{v \in V_g}\{p_{v,M}\} + (\vartheta + 1)\mathcal{S}(M) + \sigma_h$,

provided that no node performs a reset before triggering its NEXT signal, i.e., $H > p + (\vartheta+1)\mathcal{S}(M) + \sigma_h$. Recall that we already established that $H \geq h' > \max_{v\in V_g}\{p_{v,M}\}$, i.e., the local i variables have been set to 0 mod M again and will not change before the next pulse. Checking Algorithm 17, we see that such a reset thus would either occur R^+ local time after the (local) beat or due to the next pulse occuring before local time $h_{v,2} + R^-$. As $R^+/\vartheta \geq (\vartheta+1)\mathcal{S}(M)+\sigma_h$ by Inequality (13.9), the former cannot happen.

Observe that if the latter does not take place either, it would indeed follow that no node performs a reset on its second beat. Therefore, we conclude that $H \geq \min_{v\in V_g}\{p_{v,M+1}, p+(\vartheta+1)\mathcal{S}(M)+\sigma_h\}$ (where we slightly abuse notation in that if v *would* generate pulse $M+1$, but Algorithm 17 prevents this and performs a reset instead, we still denote this time by $p_{v,M+1}$). Finally, assume for contradiction that $H < p + (\vartheta+1)\mathcal{S}(M) + \sigma_h$. Thus, there is some $v \in V_g$ so that $H = p_{v,M+1} < p + (\vartheta+1)\mathcal{S}(M) + \sigma_h$. However, as v is the first node performing a reset, the period bound applies, i.e.,

$$\begin{aligned} p_{v,M+1} &\geq p + \frac{T - (\vartheta+1)\mathcal{S}}{\vartheta} \\ &= p + \vartheta\mathcal{S} + d \\ &> p + (\vartheta+1)\mathcal{S}(M) + \sigma_h\,, \end{aligned}$$

where the last step uses Inequality (13.10). Thus all possible cases lead to the desired bounds on $h_{v,2}$ for all $v \in V_g$. □

We summarize the above discussions in the following theorem.

Theorem 13.6. *Assume that* $7 - 6\vartheta^2 > 0$ *and* (13.1)-(13.10) *hold. Set* $T := \vartheta((\vartheta^2+\vartheta+1)\mathcal{S} + \vartheta d)$. *If the beats behave as required by Definition 13.2, Algorithm 17 running in conjunction with Algorithm 16 (where estimates are computed according to Lemma 10.8) is a self-stabilizing solution to the pulse synchronization problem. Its skew is in* $O\left(u + (\vartheta-1)(d+\mathcal{S})\right)$ *and the generated pulses satisfy* $P_{\min} \geq (T - (\vartheta+1)\mathcal{S})/\vartheta$ *and* $P_{\max} \leq T + 3\mathcal{S}$. *The stabilization time (not accounting for the beats) is* $O(MT) = O(M(\mathcal{S}+d))$.

Proof. We apply Lemma 13.5 to each beat but the first, showing that $H = \infty$. Corollary 13.4 then yields the claims. □

Satisfying the constraints

Lemma 13.7. *Suppose for some constant* $B \in \mathbb{R}_{>0}$ *we are given a family of self-stabilizing pulse synchronization algorithms* $\mathcal{A}(b)$, $b \in [B, \infty)$, *that satisfies Definition 13.2 with the following constraints:*

- $\sigma_h = O(d)$,

- $B_1 = Cd$ *for a sufficiently large constant* C,
- $B_2 = b$, *and*
- $B_3 \geq \alpha B_2 - O(d)$ *for a sufficiently large constant* α.

If $\vartheta \leq \vartheta_0$ *for a constant* $\vartheta_0 > 1$, *then there is* $b = O(d \log d/u)$ *and choices of parameters such that* (13.1)-(13.10) *hold for* $\mathcal{A}(b)$, *where* $\mathcal{S}(M) = O(u + (\vartheta - 1)d)$, $T = O(d)$, *and* $M = O(\log(1 + d/u))$.

Proof. We choose

$$R^+ := R^- + (\vartheta + 1)\mathcal{S}(M) + \sigma_h \tag{13.11}$$

and $\mathcal{S}$ in accordance with (13.2), immediately ensuring (13.1), (13.2), and (13.9). This also guarantees (13.10), because

$$\begin{aligned} \mathcal{S} &= R^+ + \sigma_h - \frac{R^-}{\vartheta} && \text{(13.2)} \\ &> (\vartheta + 1)\mathcal{S}(M) + \sigma_h && \text{(13.11), } \vartheta > 1 \end{aligned}$$

In particular,

$$\begin{aligned} \mathcal{S} &> \mathcal{S}(M) && \text{(13.10)} \\ &= \frac{\mathcal{S}}{2^{M-1}} + \left(1 - \frac{1}{2^{M-1}}\right) 2 \left(\delta + \left(1 - \frac{1}{\vartheta}\right)(T + \mathcal{S} + \delta)\right), \end{aligned}$$

which is equivalent to

$$\mathcal{S} > 2\left(\delta + \left(1 - \frac{1}{\vartheta}\right)(T + \mathcal{S} + \delta)\right).$$

Using that $2 - \vartheta > 0$, this yields that

$$\mathcal{S} > \frac{2\left(\vartheta\delta + (\vartheta - 1)\left(T + \delta\right)\right)}{2 - \vartheta} = \frac{2(2\vartheta - 1)\delta + 2(\vartheta - 1)T}{2 - \vartheta},$$

i.e., shows that (13.3) holds.

We now turn to prove (13.4). Due to the choice of R^+, (13.4) becomes

$$\begin{aligned} R^- &\geq \vartheta(\sigma_h + \vartheta\mathcal{S} + d) && \text{(13.4)} \\ &= \vartheta(\sigma_h + \vartheta(R^+ + \sigma_h) - R^- + d) && \text{(13.2)} \\ &= \vartheta(\sigma_h + \vartheta(R^- + (\vartheta + 1)\mathcal{S}(M) + 2\sigma_h) - R^- + d). && \text{(13.11)} \end{aligned}$$

Since ϑ is small enough, $1 + \vartheta - \vartheta^2 > 0$ and we choose to satisfy this inequality without slack, by setting

$$R^- := \frac{(2\vartheta^2 + \vartheta)\sigma_h + (\vartheta^3 + \vartheta^2)\mathcal{S}(M) + \vartheta^2 d}{1 + \vartheta - \vartheta^2}. \tag{13.12}$$

Next, observe that

$$\begin{aligned}
\mathcal{S} &< \sigma_h + R^+ && \text{(13.2)}\\
&= R^- + (\vartheta+1)\mathcal{S}(M) + 2\sigma_h && \text{(13.11)}\\
&= O(\mathcal{S}(M) + \sigma_h) && \text{(13.12)}\\
&= O(\mathcal{S}(M) + d) && \sigma_h = O(d)\\
&= O\left(\delta + (\vartheta-1)T + d + \left(\vartheta - 1 + \frac{1}{2^{M-1}}\right)\mathcal{S}\right) && \text{def. of } \mathcal{S}(M)\\
&= O\left(d + \left(\vartheta - 1 + \frac{1}{2^{M-1}}\right)\mathcal{S}\right). && \text{def. of } \delta \text{ and } T,\ u < d
\end{aligned}$$

Using that $\vartheta - 1$ is a sufficiently small constant and assuming that M is at least a large enough constant, we can ensure that $\vartheta - 1 + 1/2^{M-1} < \varepsilon$ for any constant $\varepsilon > 0$.

Fixing $M = k\lceil \log(1 + d/u)\rceil \geq k$ for a sufficiently large constant $k \in \mathbb{N}_{>0}$, we can thus conclude that $\mathcal{S} = O(d)$. In particular,

$$R^+ + \sigma_h < \mathcal{S} = O(d),$$

implying by the assumptions of the lemma that (13.6) is satisfied. Moreover, $T = O(\mathcal{S} + d) = O(d)$ and

$$\begin{aligned}
\mathcal{S}(M) &= \frac{\mathcal{S}}{2^{M-1}} + O(\delta + (\vartheta-1)(T+\mathcal{S})) && \text{def. of } \mathcal{S}(M)\\
&= O(u + (\vartheta-1)(T+\mathcal{S}+d)) && \text{choice of } M,\ \text{def. of } \delta\\
&= O(u + (\vartheta-1)d). && T + \mathcal{S} = O(d)
\end{aligned}$$

To complete the proof, we need to show that we can choose $b = O(d \log d/u)$ so that the remaining inequalities (13.5), (13.8), and (13.7) are satisfied. Note that due to the established bounds, the r.h.s. of (13.5) is $O(d)$, implying that any choice of $b \geq B$ that is at least a sufficiently large constant guarantees (13.5). The inequalities (13.8) and (13.7) are now of the form

$$\begin{aligned}
B_2 &\leq (M-1)\frac{T - (\vartheta-1)\mathcal{S}}{\vartheta} + O(d) = (M-1)(\vartheta\mathcal{S} + d) + O(d)\\
B_3 &> (M-1)(T + 3\mathcal{S}) + O(d) = (M-1)((\vartheta^2 + \vartheta + 4)\mathcal{S} + \vartheta d) + O(d).
\end{aligned}$$

Therefore, if $\alpha > 6\vartheta$, choosing $b = O(MT) = O(d \log d/u)$ maximal such that (13.8) holds entails that

$$B_3 \geq \alpha B_2 - O(d) = (M-1)((\vartheta^2 + \vartheta + 4)\mathcal{S} + \vartheta d) + (M - O(1))d.$$

Since $M \geq k$ and we chose k sufficiently large, this implies (13.7). □

Bibliography

[1] Hopkins, A. L., T. B. Smith, and J. H. Lala. 1978. Ftmp – a highly reliable fault-tolerant multiprocess for aircraft. *Proceedings of the IEEE* 66 (10): 1221–1239. doi:10.1109/PROC.1978.11113.

[2] Kopetz, H. 2003. Fault containment and error detection in the time-triggered architecture. In *The sixth international symposium on autonomous decentralized systems, 2003. isads 2003.*, 139–146. doi:10.1109/ISADS.2003.1193942.

[3] Pease, M., R. Shostak, and L. Lamport. 1980. Reaching agreement in the presence of faults. *J. ACM* 27 (2): 228–234. doi:10.1145/322186.322188. http://doi.acm.org/10.1145/322186.322188.

[4] Srikanth, T. K., and Sam Toueg. 1987. Optimal clock synchronization. *J. ACM* 34 (3): 626–645. doi:10.1145/28869.28876. https://doi.org/10.1145/28869.28876.