Lecture 5

Synchronizing by Approximate Agreement

In the previous lecture, we've seen how to achieve a skew of $\mathcal{O}(d)$ in a system of n fully connected nodes with $f < n/3$ Byzantine faults. We've also seen that we can't do any better in terms of the number of faults that can be tolerated. So let's ask our usual question: Is this skew bound (asymptotically) optimal or can we do better? Already in a fault-free system, we know that we can't beat $\Omega(u + (\vartheta - 1)d)$. But can this bound be attained in the presence of faults?

5.1 Approximate Agreement

The answer is provided by leveraging techniques for the task of *approximate agreement.* For this problem, we assume the (convenient) abstraction of a synchronously operating system.

Definition 5.1 (Synchronous Execution)**.** *A synchronous execution proceeds in* synchronous rounds. *At the start of the execution, each node receives an input (whose type depends on the task at hand). In each round,*

1. *nodes perform local computations,*
2. *send messages to their neighbors in the network graph,*
3. *receive the messages of their neighbors, and (optionally)*
4. *may compute an output value and terminate (i.e., stop executing the other steps in future rounds).*

Note that a synchronous execution of a deterministic algorithm is fully determined by the input values and the (arbitrary) messages sent by faulty nodes. The key performance measures are the round complexity — the number of rounds until all nodes terminated — and the maximum size of messages (sent by correct nodes).

This model provides a very clean abstraction for describing the tool we would like to use.

Definition 5.2 (Approximate Agreement)**.** *Each node $v \in V$ is given an input value $r_v \in \mathbb{R}$. Given a constant $\varepsilon > 0$, the task is to generate output values $o_v \in \mathbb{R}$ so that*

agreement: $\max_{v,w \in V_g}\{o_v - o_w\} \leq \varepsilon$,

validity: $\forall v \in V_g \colon \min_{w \in V_g}\{r_w\} \leq o_w \leq \max_{w \in V_g}\{r_w\}$, *and*

termination: *each $v \in V_g$ determines is output o_v and terminates within a finite number of rounds.*

Remarks:

- The synchronous model is a highly useful abstraction in distributed computing. With known upper bounds $\mathcal{L}$ on local skew, λ on logical clock rates, and d on message delays, it is straightforward to simulate. Assuming that $\max_{v \in V_g}\{L_v(0)\} = L$, nodes send their messages for round $r \in \mathbb{N}$ at the time t when $L_v(t) = L + (r-1)\lambda(d + \mathcal{L})$. Thus, all messages for round r are received before the ones for round $r+1$ need to be sent.

- If the round number is not to be sent along with the message or for some other reason it's important that messages for round $r+1$ must not arrive anywhere before round r is complete at all nodes, one may add an additional $\lambda\mathcal{S}$ at the beginning of the round before messages are sent. We will use this in our algorithm!

- Recall that d accounts for local computations, not only the time messages are in transit. Thus, involved calculations affect the time the simulation takes via d!

- Lower bounds on the progress of logical clocks are needed for guaranteeing progress. The better the lower bound, the earlier the simulation completes (i.e., all nodes terminate).

- Without faults, synchronizers provide elegant solutions that work even if d is unknown. However, synchronizers wait for proof that all other nodes finished their current round before proceeding. Even a single crash fault (a node not sending any messages any more) would halt the entire system!

- Once we solved approximate agreement in this abstract model, we will employ it to agree on when the nodes should generate clock pulses, i.e., solve the pulse synchronization problem with it.

- The simulation of the synchronous algorithm and maintaining a small skew will go hand in hand!

Solving Approximate Agreement

Definition 5.3 (Diameters of Vectors)**.** *Denote by $\vec{r}$ the $|V_g|$-dimensional vector of correct nodes' inputs, i.e., $(\vec{r})_v = r_v$ for $v \in V_g$. Denote by $r^{(k)}$, $k \in \{1, \ldots, |V_g|\}$, the k^{th} entry when ordering the entries of $\vec{r}$ ascendingly.*

Algorithm 5.1: Approximate agreement step at node $v \in V_g$ (with synchronous message exchange).

```
1 // node $v$ is given input value $r_v$;
2 broadcast $r_v$ to all nodes (including self);
3 receive $\hat{r}_{wv}$ from each node $w$ ($\hat{r}_{wv} := r_v$ if no message with correct type
    of content from $w$ received);
4 $S_v \leftarrow \{\hat{r}_{wv} \,|\, w \in V\}$;
5 $o_v \leftarrow \frac{S_v^{(f+1)} + S_v^{(n-f)}}{2}$;
6 return $o_v$;
```

The diameter $\|\vec{r}\|$ *of* $\vec{r}$ *is the difference between the maximum and minimum components of* $\vec{r}$*. Formally,*

$$\|\vec{r}\| := r^{(|V_g|)} - r^{(1)} = \max_{r \in V_g}\{r_v\} - \min_{v \in V_g}\{r_v\}.$$

We will use the same notation for other values, e.g. $\vec{o}$*,* $o^{(k)}$*,* $\|\vec{o}\|$*, etc.*

For simplicity, we assume that $|V_g| = n - f$ in the following; all statements can be adapted by replacing $n - f$ with $|V_g|$ where appropriate. As usual, we require that $3f < n$.

Intuitively, Algorithm 5.1 discards the smallest and largest f values each to ensure that values from faulty nodes cannot cause outputs to lie outside the range spanned by the correct nodes' values. Afterwards, o_v is determined as the midpoint of the interval spanned by the remaining values. Since $f < n/3$, i.e., $n - f \geq 2f + 1$, the median of correct nodes' values is part of all intervals computed by correct nodes. From this, it is easy to see that $\|\vec{o}\| \leq \|\vec{r}\|/2$. We now prove these properties.

Lemma 5.4.

$$\forall v \in V_g \colon r^{(1)} \leq o_v \leq r^{(n-f)}\,.$$

Proof. As there are at most f faulty nodes, for $v \in V_g$ we have that

$$S_v^{(f+1)} \geq \min_{w \in V_g}\{\hat{r}_{wv}\} = r^{(1)}\,.$$

Analogously, $S_v^{(n-f)} \leq r^{(n-f)}$. We conclude that

$$r^{(1)} \leq S_v^{(f+1)} \leq \frac{S_v^{(f+1)} + S_v^{(n-f)}}{2} = o_v \leq S_v^{(n-f)} \leq r^{(n-f)}\,. \qquad \square$$

Lemma 5.5. $\|\vec{o}\| \leq \|\vec{r}\|/2$.

Proof. Since $f < n/3$, we have that $n - f \geq 2f + 1$. Hence, for all $v \in V_g$,

$$r^{(1)} \leq S_v^{(f+1)} \leq r^{(f+1)} \leq S_v^{(2f+1)} \leq S_v^{(n-f)} \leq r^{(n-f)}\,.$$

For any $v, w \in V_g$, it follows that

$$\begin{aligned} o_v - o_w &= \frac{S_v^{(f+1)} - S_w^{(f+1)} + S_v^{(n-f)} - S_w^{(n-f)}}{2} \\ &\leq \frac{r^{(f+1)} - r^{(1)} + r^{(n-f)} - r^{(f+1)}}{2} = \frac{r^{(n-f)} - r^{(1)}}{2} \\ &= \frac{\|\vec{r}\|}{2}\,. \end{aligned}$$

As $v, w \in V_g$ were arbitrary, this yields $\|\vec{o}\| \leq \|\vec{r}\|/2$. □

Applying this approach inductively yields a straightforward algorithm provided an upper bound $R \geq r^{(|V_g|)} - r^{(1)}$ is known.

Theorem 5.6 (Approximate Agreement)**.** *Applying Algorithm 5.1 iteratively (using the output of one step as input to the next) for $\lceil \log(R/\varepsilon) \rceil$ steps solves approximate agreement.*

Proof. Applying Lemma 5.5 inductively shows agreement. Applying Lemma 5.4 inductively shows validity. By construction, all nodes terminate after $\lceil \log(R/\varepsilon) \rceil$ synchronous rounds. □

Modifications for the Pulse Synchronization Problem

In our setting, we will not be able to guarantee exact communication of clock values. Accordingly, we slightly modify the communication model. More specifically, at certain times, nodes will need estimates of each other's logical clock values. Node v will use its estimate of w's clock value as approximation of the "input" r_w of $w \in V$. Thus, instead of receiving $\hat{r}_{wv} = r_w$ from $w \in V$, v will receive

$$r_w - \delta < \hat{r}_{wv} \leq r_w\,.$$

As shifting the values $\hat{r}_{wv}$ in Algorithm 5.1 by less than δ will affect the outputs by less than δ, we obtain the following corollary to Lemmas 5.4 and 5.5. See Figure 5.1 for a visualization.

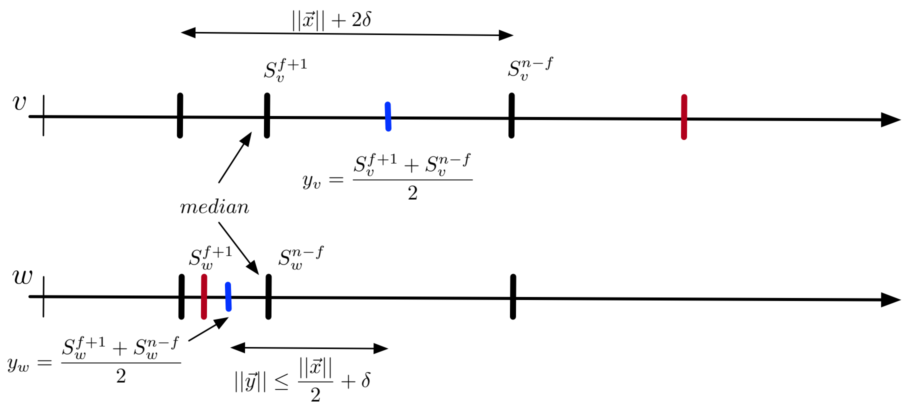
Corollary 5.7. *With the above modification to the communication model, Algorithm 5.1 guarantees*

(i) $\forall v \in V_g \colon r^{(1)} - \delta < o_v \leq r^{(n-f)}$ *and*

(ii) $\|\vec{o}\| \leq \|\vec{r}\|/2 + \delta$.

Remarks:

- Now all we need to do is to gather estimates, use Algorithm 5.1 to determine adjustments to the logical clocks, and iterate.
- Trivia: When I suggested to Danny Dolev that one could make use of approximate agreement as the basis for a clock synchronization algorithm, he told me that this was precisely the motivation for introducing the problem and pointed me towards the paper implementing this approach. He and his co-authors were merely about three decades and a brilliant abstraction ahead of me!



$||\vec{x}|| + 2\delta$
S_v^{f+1}
S_v^{n-f}
v
$y_v = \frac{S_v^{f+1} + S_v^{n-f}}{2}$
median
S_w^{f+1}
S_w^{n-f}
w
$y_w = \frac{S_w^{f+1} + S_w^{n-f}}{2}$
$||\vec{y}|| \leq \frac{||\vec{x}||}{2} + \delta$

Figure 5.1: An execution of Algorithm 5.1 at nodes v and w of a system consisting of $n = 4$ nodes. There is a single faulty node and its values are indicated in red. Note that the ranges spanned by the values received from non-faulty nodes are *almost* identical; the difference originates in the perturbations of up to δ.

5.2 A Variant of the Lynch-Welch Algorithm

The algorithm is now constructed as follows. Assuming some bound $H \geq \max_{v \in V_g}\{H_v(0)\}$ on the skew at initialization, nodes generate their first pulse at local time H. This marks the (local) start of the first round. Then they wait until they can be sure that all nodes have generated their pulse. At the respective hardware time, they transmit an empty message — no content is needed, as the local time when the message is sent is hardwired into the algorithm. Then

Algorithm 5.2: Lynch-Welch pulse synchronization algorithm, code for node $v \in V_g$. $\mathcal{S}$ denotes a (to-be-determined) upper bound on $\|\vec{p}_r\|$ for each $r \in \mathbb{N}$ and T is the nominal round duration.

```
1  // $H_w(0) \in [0, \mathcal{S})$ for all $w \in V$
2  set $L_v(0) := H_v(0)$
3  increase $L_v$ at rate $h_v$ at all times
4  generate pulse 1 at the time $p_{v,1}$ with $L_v(p_{v,1}) = \mathcal{S}$;
5  **foreach** *round* $r \in \mathbb{N}$ **do**
6      wait until local time $(r-1)T + (\vartheta + 1)\mathcal{S}$;// all nodes are in round $r$
7      broadcast empty message to all nodes (including self);
8      wait until time $\tau_{v,r}$ when $L_v(\tau_{v,r}) = (r-1)T + (\vartheta^2 + \vartheta + 1)\mathcal{S} + \vartheta d$;
         // correct nodes' messages arrived
9      **for** *each node* $w \in V$ **do**
10         // abbreviate $p_r := \max_{w \in V_g}\{p_{w,r}\}$ (unknown to the node!)
11         compute $\Delta_w^v \in (L_v(p_r) - L_w(p_r) - \delta, L_v(p_r) - L_w(p_r)]$
12     $S_v \leftarrow \{\Delta_w^v \mid w \in V\}$ (as multiset, i.e., values may repeat)
13     $L_v(\tau_{v,r}) \leftarrow L_v(\tau_{v,r}) + \left(S_v^{(f+1)} + S_v^{(n-f)}\right)/2$
14     generate pulse $r+1$ at the time $p_{v,r+1}$ with $L_v(p_{v,r+1}) = \mathcal{S} + rT$
```

nodes wait until the local time when all such messages from correct nodes are certainly received and compute their estimates of the relative clock differences to other nodes. Finally, they apply Algorithm 5.1 to compute an adjustment to the (local) starting time of the next round. This ensures bounded skew for the next pulse and thus also the starting times of the next round. From there, the process is iterated.

Algorithm 5.1 is phrased in a parametrized fashion suitable for the analysis. This means that we assume a skew bound of $\mathcal{S}$ to hold on initialization, an error bound δ on the logical clock estimates nodes compute of each other, and a nominal round duration of T. We then determine valid choices for these parameters from the analysis, where we need to determine δ depending on how the estimates are computed.

"Rounds" of the algorithm simulate the synchronous operation assumed in the approximate agreement problem, where each iteration of the loop simulates one synchronous round. For this to work as intended, two requirements need to be met in each round:

(i) Messages sent by correct nodes are received at all correct nodes after starting the round and before they compute their clock adjustment, i.e., during $[p_{v,r}, \tau_{v,r}]$.

(ii) T is large enough to ensure that the clock adjustment makes no logical clock "jump" past $L_v(p_{v,r+1}) = \mathcal{S} + rT$, skipping a pulse.

If these properties are satisfied in round r, we will say that *round r is executed correctly.* We will show that this holds for all $r \in \mathbb{N}$ inductively, where the induction hypothesis is that $\|\vec{p}_r\| \leq \mathcal{S}$; this simulatenously shows that the algorithm has a small skew! For $r = 1$, this is immediate from our assumption on the initial hardware clock values.

Lemma 5.8. *Suppose that $T/\vartheta \geq (\vartheta^2+\vartheta+1)\mathcal{S}+\vartheta d$ and $\mathcal{S} \geq 2(\delta+(1-1/\vartheta)T)$. Moreover, assume that for $r \in \mathbb{N}$ it holds that all prior rounds have been executed correctly, and that $\|\vec{p}_r\| \leq \mathcal{S}$. Then*

(i) round r is executed correctly,

(ii) $\|\vec{p}_{r+1}\| \leq \mathcal{S}$, and

(iii) $T/\vartheta - \mathcal{S} \leq p_{r+1} - p_r \leq T + \delta$.

Proof. By assumption, no messages sent by correct nodes in rounds $r' < r$ are received in round r. Consider the message $v \in V$ sends after entering round r. It is sent no earlier than time $p_{v,r} + \mathcal{S} \geq \max_{w \in V_g}\{p_{w,r}\}$, as $\|\vec{p}_r\| \leq \mathcal{S}$ by assumption. It is received before time

$$p_{v,r} + \vartheta\mathcal{S} + d \leq \min_{w \in V_g}\{p_{w,r}\} + (\vartheta+1)\mathcal{S} + d\,.$$

As $\tau_{w,r} \geq p_{w,r} + (\vartheta+1)\mathcal{S} + d$ for all $w \in V_g$, this shows part (i) of correct execution of round r.

Concerning part (ii), we apply statement (i) of Corollary 5.7, showing that the logical clock of $v \in V_g$ cannot be set to a larger value than

$$\begin{aligned}
L_v(\tau_{v,r}) &\leq L_v(p_r) + \int_{p_r}^{\tau_{v,r}} h_v(t)\,dt - \min_{w\in V_g}\{\Delta_w^v\}\\
&\leq L_v(p_r) + \int_{p_r}^{\tau_{v,r}} h_v(t)\,dt + \max_{w\in V_g}\{L_w(p_r)\} - L_v(p_r)\\
&\leq \max_{w\in V_g}\{L_w(p_r)\} + \vartheta(\tau_{v,r} - p_r)\\
&\leq \max_{w\in V_g}\{L_w(p_{w,r}) + \vartheta(\tau_{v,r} - p_{w,r})\}\\
&= (r-1)T + \mathcal{S} + \vartheta\left(\tau_{v,r} - \min_{w\in V_g}\{p_{w,r}\}\right).
\end{aligned}$$

It follows that no node can reach logical clock value $rT + \mathcal{S}$ earlier than time $\min_{w\in V_g}\{p_{w,r}\} + T/\vartheta$. In particular, this is bounded from below by $p_r + T/\vartheta - \mathcal{S}$, showing the lower bound of the third claim of the lemma.

On the other hand, for all $v \in V_g$, we have that

$$\tau_{v,r} \leq p_{v,r} + (\vartheta^2 + \vartheta)\mathcal{S} + \vartheta d \leq \min_{w\in V_g}\{p_{w,r}) + (\vartheta^2 + \vartheta + 1)\mathcal{S} + \vartheta d\,,$$

where the second step uses that $\|p_{v,r}\| \leq \mathcal{S}$. As $T/\vartheta \geq (\vartheta^2 + \vartheta + 1)\mathcal{S} + \vartheta d$, this shows that round r is executed correctly. In particular, the times $p_{v,r+1}$, $v \in V_g$, are well-defined.

By statement (i) of Corollary 5.7, we have that, at time $\tau_{v,r}$, $v \in V_g$ cannot set its logical clock to a smaller value than

$$\begin{aligned}
L_v(\tau_{v,r}) &\geq L_v(p_r) + \int_{p_r}^{\tau_{v,r}} h_v(t)\,dt + \min_{w\in V_g}\{L_w(p_r)\} - L_v(p_r) - \delta\\
&\geq \int_{p_r}^{\tau_{v,r}} h_v(t)\,dt + \min_{w\in V_g}\{L_w(p_r)\} - \delta\\
&= \int_{p_r}^{\tau_{v,r}} h_v(t)\,dt + (r-1)T + \mathcal{S} - \delta\,.
\end{aligned}$$

As hardware clock rates are at least 1, this shows that $p_{r+1} \leq p_r + T + \delta$, i.e., the upper bound of the third claim of the lemma holds.

It remains to show the second claim, i.e., the bound on the skew. To simplify our reasoning, pretend that the clock adjustments from round r would take place at time p_r. Denote by $L'_v(p_r)$, $v \in V_g$, the respective modified logical clocks, which increase at the rate of the hardware clocks during round r and satisfy $L'_v(t) = L_v(t)$ at times $t \geq \tau_{v,r}$. By the above bound, we thus have that

$$L'_v(p_r) = L_v(\tau_{v,r}) - \int_{p_r}^{\tau_{v,r}} h_v(t)\,dt \geq (r-1)T + \mathcal{S} - \delta\,.$$

Next, note that $\|\vec{L}(p_r)\| \leq \vartheta\|\vec{p_r}\| \leq \vartheta\mathcal{S}$ by assumption. By statement (ii) of Corollary 5.7, this implies that $\|\vec{L}'(p_r)\| \leq \vartheta\mathcal{S}/2 + \delta$. Now let $v, w \in V_g$ maximize $p_{r+1,v} - p_{r+1,w}$. We have that $p_{r+1,v} - p_r \leq rT + \mathcal{S} - L'_v(p_r)$ and $p_{r+1,w} - p_r \geq (rT + \mathcal{S} - L'_w(p_r))/\vartheta$ due to the bounds on the hardware clock rates. Hence,

$$\begin{aligned} p_{r+1,v} - p_{r+1,w} &\leq L'_w(p_r) - L'_v(p_r) + \left(1 - \frac{1}{\vartheta}\right)(rT + \mathcal{S} - L'_w(p_r)) \\ &\leq \|\vec{L}'(p_r)\| + \left(1 - \frac{1}{\vartheta}\right)(rT + \mathcal{S} - (L'_v(p_r) + \|\vec{L}'(p_r)\|)) \\ &\leq \|\vec{L}'(p_r)\| + \left(1 - \frac{1}{\vartheta}\right)(T + \delta - \|\vec{L}'(p_r)\|) \\ &\leq \frac{\vartheta\mathcal{S}}{2} + \delta + \left(1 - \frac{1}{\vartheta}\right)\left(T - \frac{\vartheta\mathcal{S}}{2}\right) \\ &= \frac{\mathcal{S}}{2} + \delta + \left(1 - \frac{1}{\vartheta}\right)T\,. \end{aligned}$$

This being bounded by $\mathcal{S}$ is equivalent to $\mathcal{S} \geq 2(\delta + (1 - 1/\vartheta)T)$. □

Before we can prove our main theorem, we need to get a hold on δ. This is a straightforward calculation.

Lemma 5.9. *Suppose round r is executed correctly and $v \in V_g$ receives the message from $w \in V_g$ for this round at time t. Then setting*

$$\Delta_w^v := L_v(t) - (r-1)T - (\vartheta^2 + 1)\mathcal{S} - \vartheta d$$

yields an estimate satisfying $\delta \leq u + (\vartheta - 1)d + 2(\vartheta^2 - \vartheta)\mathcal{S}$.

Proof. Denote by t the time when v receives the message from w and by t_s the time when it was sent. We have that

$$L_v(t) - L_w(t_s) \in (L_v(t_s) - L_w(t_s) + d - u, L_v(t_s) - L_w(t_s) + \vartheta d)\,.$$

Moreover,

$$|L_v(t_s) - L_v(p_r) - (L_w(t_s) - L_w(p_r))| \leq (\vartheta - 1)(t_s - p_r) \leq \left(\vartheta^2 - \vartheta\right)\mathcal{S}\,.$$

We conclude that

$$\begin{aligned} &L_v(t) - L_w(t_s) \in \\ &(L_v(p_r) - L_w(p_r) + d - u - (\vartheta^2 - \vartheta)\mathcal{S}, L_v(p_r) - L_w(p_r) + \vartheta d + (\vartheta^2 - \vartheta)\mathcal{S})\,. \end{aligned}$$

As $L_w(t_s) = (r-1)T + (\vartheta + 1)\mathcal{S}$ by the design of the algorithm, the claim of the lemma follows. □

Theorem 5.10. *Assume that $3 + 4\vartheta - 4\vartheta^2 - 2\vartheta^3 > 0$ and that estimates are computed according to Lemma 5.9. For any choice of*

$$T \geq \frac{6\vartheta^4(u + d)}{3 + 4\vartheta - 4\vartheta^2 - 2\vartheta^3} \in \mathcal{O}(d)\,,$$

set

$$\mathcal{S} := \frac{2(u + (\vartheta - 1)d + (1 - 1/\vartheta)T)}{1 + 4\vartheta - 4\vartheta^2} \in \mathcal{O}\left(u + \left(1 - \frac{1}{\vartheta}\right)T\right)\,.$$

If $\max_{v \in V}\{H_v(0)\} \leq \mathcal{S}$, then Algorithm 5 solves pulse synchronization with skew $\mathcal{S}$, $P_{\min} \geq T/\vartheta - \mathcal{S}$, and $P_{\max} \leq T + 2\mathcal{S}$.

Proof. Set $\delta := u + (\vartheta - 1)d + 2(\vartheta^2 - \vartheta)\mathcal{S}$ in accordance with Lemma 5.9. Thus,

$$\mathcal{S} = 2\left(u + (\vartheta - 1)d + 2(\vartheta^2 - \vartheta)\mathcal{S} + \left(1 - \frac{1}{\vartheta}\right)T\right) = 2\left(\delta + \left(1 - \frac{1}{\vartheta}\right)T\right).$$

Moreover,

$$T \geq \frac{6\vartheta^4(u + d) + 2(\vartheta^3 - 1)T}{3 + 4\vartheta - 4\vartheta^2 - 2\vartheta^3 + 2(\vartheta^3 - 1)} > \frac{6\vartheta^3(u + (\vartheta - 1)d) + 2(\vartheta^3 - 1)T}{1 + 4\vartheta - 4\vartheta^2} + \vartheta^2 d,$$

i.e.,

$$\frac{T}{\vartheta} > (\vartheta^2 + \vartheta + 1) \cdot \frac{2(u + (\vartheta - 1)d) + 2(1 - 1/\vartheta)T}{1 + 4\vartheta - 4\vartheta^2} + \vartheta d = (\vartheta^2 + \vartheta + 1)\mathcal{S} + \vartheta d.$$

The claim is now shown by a straightforward induction on the pulse number, where the hypothesis includes that all previous rounds have been executed correctly. The induction is anchored at the first pulse, which satisfies the skew bounds due to the assumed bound on the hardware clock values at time 0. The induction step is performed by invoking Lemma 5.8, where Lemma 5.9 shows that δ is indeed a bound on the quality of estimates. We obtain that $\mathcal{S}$ is a bound on the skew for all pulses and that $T/\vartheta - \mathcal{S} \leq p_{r+1} - p_r \leq T + \delta$ for each $r \in \mathbb{N}$. This implies that $P_{\min} \geq (T - \mathcal{S})/\vartheta$ and, using that $\max_{v \in V_g}\{p_{v,r}\} - \min_{v \in V_g}\{p_{v,r}\} \leq \mathcal{S}$ and $\delta < \mathcal{S}$, that $P_{\max} \leq T + 2\mathcal{S}$. $\square$

Remarks:

- The theorem requires that $3 + 4\vartheta - 4\vartheta^2 - 2\vartheta^3 > 0$, which is the case for $\vartheta \leq 1.09$. As ϑ approaches this threshold, the skew goes to ∞.

- Sending $(T, \vartheta) \to (\infty, 1)$, the ratio $P_{\max}/P_{\min} \in (1 + o(1))\vartheta$. However, when sending $T \to \infty$ while keeping ϑ fixed, the ratio converges to a constant $c \in 1 + \mathcal{O}(\vartheta - 1)$.

- If on initialization such a tight skew bound cannot be guaranteed, one can choose T accordingly larger.

- Alternatively, one can only initially use the larger T and keep reducing T alongside the decrease in (the worst-case bound on) the skew. You'll analyze this in the exercises.

- A known bound on the initial skew is necessary for executing the algorithm. You'll show this in the exercises as well.

- We haven't clarified how nodes compute their estimates of faulty nodes' clocks. What if these nodes send no or many messages during a round? The answer is simple: It doesn't matter. As the approximate agreement algorithm works regardless of what values faulty nodes provide, choosing *any* default value for nodes clearly not obeying the protocol will do.

Bibliographic Notes

Approximate agreement was introduced by Dolev et al. [DLP$^+$86], actually having the goal in mind to use it for clock synchronization. As shown by Fekete [Fek86], the rate of convergence provided by their algorithm is close to being asymptotically optimal, and it is asymptotically optimal if only one round of communication per iteration is performed. He also shows that faster convergence is possible if the (maximum) number of possible faults is smaller.

The clock synchronization protocol by Lynch and Welch [LL84] is able to exploit this, too, to achieve faster convergence and thus slightly smaller skews. The respective modification is straightforward and can also be applied to the variant presented in this lecture, which follows [KL16]. The main difference to [LL84] is, just as for the Srikanth-Toueg algorithm from the previous lecture, that no tick numbers are communicated by the algorithm. One can also adjust clock rates (as opposed to just correcting clock offsets), but this requires the additional assumption that hardware clock rates change slowly [KL16]. The Lynch-Welch algorithm has already found its way into practice: it's the synchronization mechanism underlying industrial systems used, e.g., in cars and planes [KB03, FAS09].

Bibliography

[DLP$^+$86] Danny Dolev, Nancy A. Lynch, Shlomit S. Pinter, Eugene W. Stark, and William E. Weihl. Reaching Approximate Agreement in the Presence of Faults. *J. ACM*, 33(3):499–516, 1986.

[FAS09] Matthias Függer, Eric Armengaud, and Andreas Steininger. Safely Stimulating the Clock Synchronization Algorithm in Time-Triggered Systems - a Combined Formal & Experimental Approach. *IEEE Trans. Industrial Informatics*, 5(2):132–146, 2009.

[Fek86] A. D. Fekete. Asymptotically Optimal Algorithms for Approximate Agreement. In *Proc. 5th Symposium on Principles of Distributed Computing (PODC)*, pages 73–87, 1986.

[KB03] Hermann Kopetz and G. Bauer. The Time-Triggered Architecture. *Proceedings of the IEEE*, 91(1):112–126, 2003.

[KL16] Pankaj Khanchandani and Christoph Lenzen. Self-stabilizing Byzantine Clock Synchronization with Optimal Precision. In *Proc. 18th Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 213–230, 2016.

[LL84] Jennifer Lundelius and Nancy Lynch. An Upper and Lower Bound for Clock Synchronization. *Information and Control*, 1984.