11 Low-degree Clock Distribution Networks

Chapter Contents

Learning Goals

The fault-tolerant synchronization algorithms for fully connected networks discussed in the previous two chapters are are well suited for small networks. However, for large scale networks, the communication infrastructure of $\Theta(n^2)$ links required by these algorithms becomes highly impractical.

In this chapter, we seek to adapt and generalize the introduced techniques to achieve Byzantine fault-tolerance in large networks at low node degrees. A possible scheme is to let a small number of fully connected root nodes generate clock signals with a fault-tolerant clock synchronization algorithms such as the Lynch-Welch algorithm (cf. **??**) and then distribute the clock signals over a low-degree distribution network to the remaining nodes. We will discuss such solutions in this chapter.

11.1 Overview

MF: fill in later

11.2 Fault-tolerant clock signals for large number of nodes

Assume we are facing the problem of synchronizing a large number of nodes n despite some of them being faulty. If not stated otherwise, we will assume Byzantine faults, i.e., worst-case behavior of faulty nodes. The problem of generating clock transitions at all correct nodes in synchrony then is accurately described by the pulse synchronization task, cf. Definition 9.5. Following the notation of Chapter 9, we write V_g for the set of correct nodes and $p_{v,r}$ for the time node $v \in V_g$ generates pulse $r \geq 1$, i.e., generates its r-th (rising) clock transition. We set the latter to ∞ if v never generates round r.

11.2.1 The challenge

As in previous chapters, we assume that the system behaves according to the TMP model, i.e., communication is by message passing with known upper and lower bounds on end-to-end delays, nodes have hardware clocks running at rates from $[1, \vartheta]$, and nodes perform computations by deterministic finite state machines. However, we remark that most of the nodes will not make use of the full power of these capabilities, which is line with our goal to be resource-efficient. The above applies to the set of correct nodes $V_g \subseteq V$, while faulty nodes again exhibit Byzantine, i.e., worst-case behavior.

Recall that the goal is to guarantee that our circuit generates local clock signals that fulfill the following constraints within all feasible executions:

Definition 9.5. *[Pulse Synchronization] In* pulse synchronization*, for each $i \in \mathbb{N}$, every (correct) node $v \in V_g$ generates* pulse *i exactly once. Let $p_{v,i}$ denote the time when v generates the i-th pulse. We require that there are $\mathcal{S}, P_{\min}, P_{\max} \in \mathbb{R}_{>0}$ satisfying*

1. $\sup_{i\in\mathbb{N}, v,w\in V_g}\{|p_{v,i} - p_{w,i}|\} = \mathcal{S}$ *(skew).*
2. $\inf_{i\in\mathbb{N}}\{\min_{v\in V_g}\{p_{v,i+1}\} - \max_{v\in V_g}\{p_{v,i}\}\} \geq P_{\min}$ *(minimum period).*
3. $\sup_{i\in\mathbb{N}}\{\max_{v\in V_g}\{p_{v,i+1}\} - \min_{v\in V_g}\{p_{v,i}\}\} \leq P_{\max}$ *(maximum period).*

What is different from the previous two chapters is that we do not assume a fully connected network. That is, the network graph $G = (V, E)$ need not be complete, and $v \in V_g$ can send messages to $w \in V_g$ only if $(v, w) \in E$.

Note that this means that the condition $f < n/3$ is still necessary (cf. Theorem 9.13) to solve pulse synchronization, but not sufficient: If f faults are placed in a worst-case fashion, we also need that each node has degree at least $2f + 1$ (cf. Theorem 9.2). As we seek low-degree solutions, this means we can tolerate only a number of faults that is proportional to the (minimum) node degree.

11.2.2 Fault distribution

Instead of accepting that with constant node degrees we can tolerate only a constant number of faults, we will make an additional assumption: Faults are not *distributed* in a worst-case fashion. The proof of Theorem 9.2 was based on separating a node from the rest of the network by a majority of faulty neighbors. However, the argument does not apply when the additional faults are elsewhere in the network. Hence, we will consider settings where *locally,* i.e., in the neighborhood of nodes, only a small number of nodes are faulty, but allow for many faults in the network as a whole. Care has to be taken in how this changes the model coverage!

As an example, consider a network of (uniform) in-degree 3 and the fault model allowing that for each node, at most one of its in-neighbors is faulty.

E11.1 Consider a network of size $n = 300$ and assume that each node fails with independent probability $p = 0.01$. What is the probability that at least 2 nodes fail? What is the probability that at least $n/3 = 100$ nodes fail?

E11.2 In the same setting, assume that each node has in-degree 3. Bound the probability that no node has more than 1 faulty in-neighbor from above. Can you get a similarly strong bound if the probability that a node becomes faulty can depend on whether other faults in the network fail?

E11.3 Compare the results and discuss the consequences in light of Theorem 9.2.

11.2.3 Fault containment regions

In the above exercise, two important assumptions are made: (i) that it is clear what belongs to a node including network links, and (ii) that all nodes fail independently from each other; recall that we sought to capture this in Definition 9.3.

In fact, both (i) and (ii) are related, and (i) can always be chosen in a way to make property (ii) trivially true: group all circuitry into a single node. Strictly speaking, this is the *only* way to guarantee that nodes fail independently, since it is impossible to exclude that some catrastrophic event affects multiple nodes at the same time. However, in practice we only need that dependent failures are sufficiently unlikely, i.e., failures happen *almost* independently.

In the following, we assume that nodes are fault containment regions, but stress again that this requires careful design to make sure that the assumption of (almost) independent failures is valid. Mind that depending on how fault-containment regions are chosen, the system's guaranteed properties may change significantly.

In the following sections, we present two solutions. Both follow the approach of generating the clock signal using a few fully connected nodes and propagating it by means of a low-degree network. This separates the concern of clock generation from the easier task of *distributing* the clock signal, which results in simpler and more modular solutions.

11.3 Solution: Replicated Trees

We start with a straightforward, but naive solution, whose primary purpose is to highlight a pitfall the second one avoids. Suppose a target parameter f for the number of faults. Consider a system with $n_R = 3f + 1$ *root nodes,* up to f of which are Byzantine. We assume that the correct root nodes generate clock signals that are synchronized by a pulse synchronization algorithm resilient to f Byzantine faults, e.g. Algorithm 5.

The distribution network, consisting of the $n_T = n - n_R \in (2f + 1)\mathbb{N}_{>0}$ remaining nodes and their incident links, is a $(2f + 1)$-fold replicated clock tree. Each of these trees (see **??** for how to design such a tree) is driven by a different root node, leaving f root nodes without a corresponding tree. Starting from a clock tree with $n_T/(2f + 1)$ non-root nodes, we replicate each non-root node $2f$ additional times and denote for an original node v the resulting nodes by v_i, $i \in \{1, \dots, 2f + 1\}$.

Each of the clock trees now propagates the pulses of its root, which occur close to each other, because the root nodes execute a pulse synchronization algorithm. Denote by $p_{v_i,k}$ for $i \in \{1, \dots, 2f + 1\}$ and $k \in \mathbb{N}_{>0}$ the time when v_i forwards the k-th pulse. The (local) computational logic uses the pulse signals $p_{v_i,k}$ by considering for each k the median of the $2f + 1$ pulse times as the time when the k-th clock pulse arrives. That is, denoting by $P_{v,k} := \{p_{v_i,k} \mid i \in \{1, \dots, 2f + 1\}\}$ and by $P^{(j)}_{v,k}$ the j-th value when sorting $P_{v,k}$ in ascending order, the local logic will interpret $p_{v,k} := P^{(f+1)}_{v,k}$ as the k-th pulse of the fault-tolerant clock signal driving it.

Note that this implies that the replicas v_i of v need to be physically fairly close to each other, and the clock trees should be laid out such that these nodes receive the pulses from their roots at (almost) the same time. Yet, we need to separate both them and their incident links well enough for considering them separate fault containment regions!

Figure 11.1 shows a high-level circuit for the replicated trees solution for the case $f = 1$.

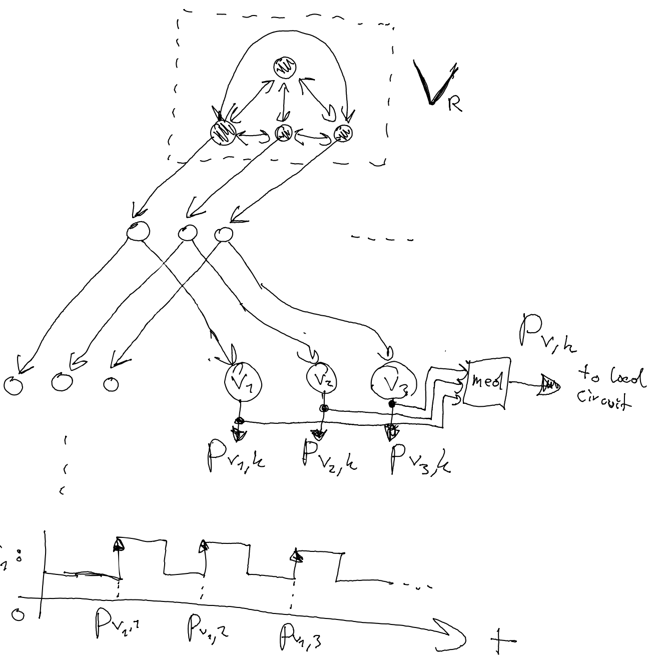

Figure 11.1
Replicated tree solution for $f = 1$. Three $(2f + 1)$ among the synchronized root nodes each drive a clock tree. The local computational logic at nodes v_1, v_2, v_3 uses the median of the 3 $(2f + 1)$ pulse times produced by v_1, v_2, v_3.

In the following, we neglect the delays of local computational logic, including the (quite involved!) logic that would be needed to determine $p_{v,k}$ locally. We now state a theorem implying that up to f Byzantine faults cannot significantly deteriorate the timing guarantees compared to a fault-free system without redundancy. Since our main focus is the resilience to faults, which readily follows from the majority of the redundant clock trees being completely without fault, we only sketch the proof.

Theorem 11.1. *Suppose that there are at most f Byzantine faulty nodes and assume that the (minimum and maximum) delays in the replicated trees are identical to those of the original, non-replicated tree. Denote by $\mathcal{S}$ the skew of the pulse synchronization algorithm executed by the root nodes. Fix any correct root node. Then all timing guarantees that the original tree driven by this root node would satisfy without faults hold for the redundant system with respect to the pulse times $p_{v,k}$, albeit weakened by an additive $\mathcal{S}$.*

Proof sketch. Since there are at most f faulty nodes, the pulse synchronization algorithm guarantees some bounded skew $\mathcal{S}$. The bound on the number of faults also implies that out of the $2f + 1$ trees, at least $f + 1$ contain no faulty node (including their roots). Because we assumed that each tree has the same characteristics as the original one, the relative timing guarantees between any pair of nodes v_i and w_j in these trees would be the same as between v and w in the original tree; as the roots instead pulse at most $\mathcal{S}$ time apart, the pulsing times of v_i and w_j might be an additional $\mathcal{S}$ apart. Finally, the fact that $p_{v,k} = P_{v,k}^{(f+1)}$ and $p_{w,k} = P_{w,k}^{(f+1)}$ ensures that the up to f values in these sets from trees with faulty nodes that violate the resulting time bounds are discarded. □

Note that, compared to the fault-free setting, there are three disadvantages in terms of timing, only one of which is explicit in the theorem:

- The skew of up to $\mathcal{S}$ between the redundant roots adds to the overall skew of the clock distribution scheme.
- The pulse synchronization algorithm is likely to provide a less stable time reference than a single clock source, in the sense that the period bounds of the pulse synchronization algorithm are weaker.
- The redundant trees will not have the exact same timing properties, and the additional circuitry to determine the median signal at computational logic might further affect timing.

On the other hand,

- the system can withstand up to f failures and
- using the median signal discards outliers, thereby reducing the effects of process variations and similar contributions to skew.

E11.4 Is the above skew bound tight? If yes, sketch the execution where the skew is indeed attained. Try to use as few faults as possible.

E11.5 How did we choose the fault-containment regions? Consider choosing an entire redundant clock tree as fault containment region. How does this affect the results? Does it matter whether we can ensure independence of faults within a single tree?

E11.6 Suppose that the probability that one of the redundant clock trees contains at least one faulty node is q. What is the probability that there are at most f trees with a faulty node? How should you choose f for a given q when trying to make sure that the clocking system fails with probability at most 1%? Is this always possible?

E11.7 How would q change with system size? How large can the probability of an individual node to fail be for the above scheme to be of use?

These exercises show that there is limited gain in this approach. The main issue is that each fault may bring down a large part of a clock tree. While a more careful analysis could yield better results—for instance, we could exploit that for each branch of the original tree, up to f redundant trees could be faulty, but these do not need the *same* trees in different branches—we can do much better when applying the idea of relying on *local* redundancy when propagating the clock signal.

11.4 Solution: Interlinked Trees

Theorem 11.1 has two major shortcomings:

1. It requires that the *total* number of faults is at most f, although Theorem 9.2 only imposes a *local* limit on the faults in each neighborhood. Accordingly, we can still hope that the overall possible number of faults that can be sustained is much larger, so long as the local constraint is satisfied.
2. The computational logic is clocked by the median pulses from several clock trees, which would require fairly complex circuitry.

An improvement that addresses both issues at the same time is to integrate the selection of the median signal into the tree nodes, see Figure 11.2. First, this means that we now only require that for each node v in the original tree, fewer than half of its replicas are faulty, which is much less stringent than requiring half of the *trees* to contain no faulty node. Second, clocking computational logic by only one replica of a node now means that it will receive a correct and synchronized clock signal so long as the supplying node is faulty (and the overall constraints on faults are satisfied). This provides the flexibility to decide for each part of the computational logic how much redundancy is needed—and if redundancy is required, we can replicate the computational modules and clock each replica from a different replica of the corresponding tree node.

As this intertwines the clock signal propagation in the redundant clock trees, we refer to this scheme as *interlinked trees* in the following.

Again, for simplicity we assume that the timing properties of the original clock tree are preserved. That is, if (v, w) is an edge in the original tree with end-to-end delay from $[d-u, d]$, the same applies for (v_i, w_j) for $i, j \in \{1, \ldots, 2f+1\}$. As before, set $P_{v,k} := \{p_{v_i,k} \mid i \in \{1, \ldots, 2f+1\}\}$, denote by $P_{v,k}^{(j)}$ the j-th value when sorting $P_{v,k}$ in ascending order, and set $p_{v,k} := P_{v,k}^{(f+1)}$

Theorem 11.2. *Suppose that there are at most f Byzantine faulty root nodes and for each node v in the original tree, at most f of its replicas are faulty. Moreover, assume that the (minimum and maximum) delays in the replicated trees are identical to those of the original, non-replicated tree. Denote by $\mathcal{S}$ the*

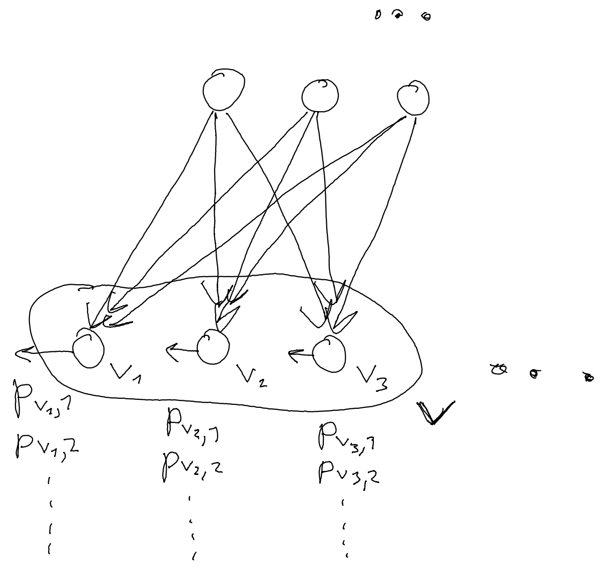

Figure 11.2
Interlinked trees solution for $f = 1$. The solution differs from the replicated trees solution in that each stage comprising of the $2f+1 = 3$ replica nodes receives the output clock signals from all replicas of the parent node. This extends to the first stage, in which each node receives signals from $2f + 1 = 3$ root nodes. A single such hop in the (replicated) clock distribution tree is shown.

skew of the pulse synchronization algorithm executed by the root nodes. Fix any correct root node. Then all timing guarantees that the original tree driven by this root node would satisfy without faults hold for the redundant system with respect to the times $p_{v,k}$, albeit weakened by an additive $\mathcal{S}$.

Proof sketch. The proof is very similar to the one of Theorem 11.1.

We prove the claim by induction on the nodes of the original tree, where we show that $p_{v_i,k}$ satisfy the claimed time bounds for all $v_i \in V_g$ and $k \in \mathbb{N}_{>0}$. Since at most f replicase of an original node v may be faulty, this readily implies the same for $p_{v,k}$. Since there are at most f faulty roots, the pulse synchronization algorithm guarantees some bounded skew $\mathcal{S}$. This anchors the induction at the root of the original tree.

For the induction step, consider a node w of the original tree with parent v for which we already established the claim. Since the claim already holds for all of the at least $f + 1$ correct replicas v_i of v and we assume that the replica edges have the same end-to-end delay as (v, w) in the original tree, we get for each correct replica w_i of w that it will discard the at most f values outside the feasible time interval for its pulse. Thus, the induction step succeeds. □

Compared to Theorem 11.1, there are a few downsides:

- Where before we increased the number of edges by (roughly) factor $2f + 1$, now this factor is $(2f + 1)^2$. Even for the (arguably) most interesting case of $f = 1$, this means a factor-9 increase rather than a factor of 3.
- Each node in the clock tree now needs to select the median signal out of the arriving ones. Even if pulses are clearly separated, this requires non-trivial logic, which will affect timing.

In return,

- the system can tolerate a much larger number of faults, so long as they are distributed well (see exercises below),
- the benefit of discarding outliers now applies in each stage of the tree, and
- the computational logic does not require the ability to select the median clock signal, but can be clocked directly from the output of a single node.

E11.8 Denote by q the (independent) probability with which an individual node v has more than f faulty replicas. If the system should still work correctly as a whole with probability 99%, how large can q be?

E11.9 Given n and f, estimate the probability with which individual nodes may independently fail such that the system as a whole works correctly.

E11.10 For large n, which of the two solutions is more robust to faults? Does this change if you choose the value of f for the first solution large, such that the number of links in both schemes is the same?

11.4.1 Synchronization between replicas of the same node

We have suggested above the possibility to clock computational logic from individual replicas, i.e., using $p_{v_i,k}$ for some fixed i as the k-th clock pulse rather than determining $p_{v,k}$ locally. If we are using replicas of the same logic block clocked by different v_i, it is likely that their outputs will ultimately be compared to weed out faulty values in one way or another. This means that it is of interest to bound the skew between $|p_{v_i,k} - p_{v_j,k}|$ for $i \neq j$.

E11.11 Recall that there can be up to f faulty replicas of each node. As function of $\mathcal{S}$ and the end-to-end delays on the path from the root to v in the original tree, how large can $|p_{v_i,k} - p_{v_j,k}|$ become in the worst case? Provide an execution where the maximum is attained.

This worst-case bound is only attained when there are many faults in consecutive stages, however. If there is just a single fault-free stage, immediately tight bounds on $|p_{v_i,k} - p_{v_j,k}|$ are (re-)established.

Lemma 11.3. *Suppose (v, w) is an edge in the original tree and assume its end-to-end delay is from $[d - u, d]$. If there is no faulty replica of v and a majority of them generates pulse k, then for any $w_i, w_j \in V_g$ we have that $|p_{v_i,k} - p_{v_j,k}| \leq u$.*

Proof. Since more than half of the replicas of v generate their k-th pulse, each correct replica of w generates their k-th pulse eventually. The reception times of the k-th pulses from replicas of v differ by at most u. Since all replicas of v are correct, this implies that the reception times of the median signals at replicas of w do not differ by more than u. □

We remark that if there are fewer than f, but non-zero faults in a stage, some weaker form of convergence can be shown. However, as we are mostly interested in the case of $f = 1$, the above lemma is sufficient.

E11.12 Fix $f = 1$. For given n, what is the probability p of independent node failures that can be sustained such that overall system failur occurs with probability at most 1%?

E11.13 Given the bound on p from the previous exercise, how likely is it that s consecutive stages contain a fault?

E11.14 Use the probability bound you just computed and a union bound to conclude that, with probability at least 99%, for no node v of the original tree, both its parent and its grandparent has a faulty replica.

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.