Optimal Synchronization of ABD Networks

Gerard Tel   Ephraim Korach   Shmuel Zaks

RUU-CS-88-23
May 1988

Utrecht University
Department of Computer Science
Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : + 31 - 30 - 531454

Optimal Synchronization of ABD Networks

Gerard Tel   Ephraim Korach   Shmuel Zaks

Technical Report RUU-CS-88-23
May 1988

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

ISSN: 0924–3275

Optimal Synchronization of ABD Networks

Gerard Tel[*] Ephraim Korach[†] Shmuel Zaks[‡]

Abstract

We present in this paper a simple and efficient synchronizer algorithm for Asynchronous Bounded Delay Networks. In these networks each processor has a local clock, and the message delay is bounded by a known constant. The algorithm improves on an earlier synchronizer for this network model, presented by Chou *et al.*[CCGZ90]. Moreover, using a mathematical model for this type of synchronizer, we show that the round time of the new synchronizer is optimal.

1 A Synchronizer Algorithm

Two models of computation have been used for the development of distributed algorithms: the *synchronous* and the *asynchronous* model. In the synchronous model the execution of an algorithm operates in discrete steps called *rounds*. The actions of a process in round $(i+1)$ depend on the state of the process after round i and the messages sent to it in round i. Note that it is therefore necessary that all messages that are sent to some process in round i are received before the process starts its computation of round $(i+1)$. We can think of the system as if there were a global clock, giving pulses at regular intervals. Computation takes place at clock pulses, and a message that is sent at one pulse is guaranteed to be received before the next pulse. In the asynchronous model it is assumed that there are no clocks and message delivery time is not bounded *a priori*.

[*]The work of this author was supported by the ESPRIT III Basic Research Actions Program of the EC under contract no. 7141 (project ALCOM II). Author's address: Department of Computer Science, University of Utrecht, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands. **Email:** `gerard@cs.ruu.nl`.

[†]Author's address: Department of Industrial Engineering, Ben Gurion University of the Negev, P.O. Box 653, Beer Sheva 84105, Israel. **Email:** `korach@bgumail.bgu.ac.il`.

[‡]Author's address: Department of Computer Science, The Technion, Technion City, Haifa 32000, Israel. **Email:** `zaks@cs.technion.ac.il`.

The synchronous model is stronger than the asynchronous model. Consequently, distributed algorithms for synchronous networks are more efficient and easier to design than algorithms for asynchronous networks. Therefore simulation algorithms have been designed to simulate synchronous computations on asynchronous networks. These simulation algorithms are called *synchronizers* [Awe85]. The simplest of these mechanisms ensures that exactly one message is sent over each link of the network in every round. If the simulated algorithm sends more messages over some link in some round, these messages must be packed into one larger (logical) message. If the simulated algorithm sends no messages over some link in some round, a special "empty message" must be sent. As a result of this policy, every process must receive exactly one message from every neighbor after every round. The next round is simulated when the message of the current round has been received from every neighbor.

The addition of the empty messages makes the synchronizer inefficient for computations that are "sparse" in time. The message complexity of the simulated algorithm equals its time complexity multiplied by the number of edges in the network. Consider, for example, the construction of a breadth first search tree in a network with E bidirectional edges and diameter D. A simple synchronous algorithm uses $2E$ messages and time D. When the simple synchronizer is used to simulate this algorithm, $2ED$ messages are sent in time $O(D)$. The situation is even worse for the simulation of some recent election and spanning tree algorithms by Vitanyi [Vit85]. These algorithms use a number of messages linear in the number of edges, but they are exponential in time. The simple synchronizer would increase the message complexity to exponential.

1.1 Asynchronous Bounded Delay Networks

Chou *et al.* [CCGZ90] proposed a network model, referred to as *Asynchronous Bounded Delay networks* (ABD networks). This model is weaker than the synchronous model, but stronger than the asynchronous model. It is assumed that processes have local clocks. These clocks run at the same speed, but they are not synchronized. That is, they need not show the same value at one instant. Furthermore a fixed bound on message delivery time is assumed. We choose our unit of time equal to this bound and assume henceforth that message delay is bounded by 1. Except in Section 1.3, we assume the network to be bidirectional.

Formally, communication satisfies the *Bounded Delay* axiom (BD); if σ is the global time of the sending of a message, and τ is the global time of its receipt, then

$$\sigma \leq \tau < \sigma + 1. \qquad \text{BD}$$

In our analysis we will always refer to a global time, but this global time is of course invisible to the processes. We assume that local clocks show a real-valued time, and that time for local processing is 0. These assumptions are justified because the granularity of the clock tick and the time for internal processing are usually very small compared to message delay time.

In ABD networks a synchronizer can work without the empty messages. An initial exchange of $\langle$ **start** $\rangle$ messages is required to make every process start its local clock at approximately the same time. After this initialization phase a processor will use its clock to decide when the next round of the simulated algorithm is executed. The following two requirements must be satisfied:

R1. If process q sends a message to its neighbor p in round i, this message must be received before p simulates round $(i+1)$; and

R2. if process p receives a message it must be possible for p to determine to what round this message belongs.

Requirement R1 must obviously be satisfied because p's actions in round $(i+1)$ depend on q's message. Failure to meet requirement R2 may lead to incorrect simulation because a message may then be processed as if belonging to a different round. If the two requirements are satisfied the synchronous computation is simulated correctly.

To compare the speed of synchronizers we introduce the concept of *round time*. The round time of a synchronizer is the time it takes to simulate one round of the synchronous algorithm. When the simple synchronizer, described earlier, is used on an ABD network, it realizes a round time of 1. No mechanism can have a smaller round time, because the simulated algorithm may be sending messages all the time, and these messages can take time up to 1 to arrive. Thus, the simple synchronizer is time-optimal, but it uses a lot of messages. From now on we only consider the message-efficient type of synchronizer for ABD networks.

Chou *et al.* [CCGZ90] presented two synchronizers. The first synchronizer has a round time of 2. To meet requirement R2, one bit is added to every basic message (i.e., every message of the simulated algorithm) as described in Theorem 2.2. This extra bit is avoided in the second synchronizer, but this is paid for with a round time of 3. In this article we present a synchronizer with a round time of 2 that works without the extra bit. This clearly improves on the results of [CCGZ90].

It remained an open question, whether a round time of 2 is optimal for the message-efficient ABD synchronizers under consideration (those that exchange control messages only during initialization). We develop a mathematical model and answer this question with "yes" for some cases and with "no" for others. Depending on their topology, some networks can be synchronized with a round time smaller than 2, while for others 2 is optimal.

We also consider the case where clocks do not run at exactly the same speed, but instead suffer from drift. Again we improve on the results of [CCGZ90].

```
var $started_p$  : boolean   init false ;
    $\delta_{pq}$       : real      for each neighbor $q$ ;

procedure $INIT$: (* Executed only if not $started_p$, either spontaneously
                    or upon receipt of the first ⟨ start ⟩ message *)
    begin $C_p := 0$ ; $started_p := true$ ;
          send ⟨ start ⟩ to every neighbor
    end

upon receipt of ⟨ start ⟩ from neighbor $q$
    do begin if not $started_p$ then $INIT$ ;
             $\delta_{pq} := C_p$
       end

upon receipt of a message $M$ from $q$
    do begin $i := \lfloor \frac{C_p - \delta_{pq} + 1}{2} \rfloor$ ;
             store $M$ as a round-$i$ message
       end

when $C_p = 2i$
    do Execute round $i$ of the simulated algorithm, using all
       round-$(i-1)$ messages so far.
```

Algorithm 1: The Synchronizer for Process p.

1.2 The Synchronizer

We first describe the initialization phase, explain when rounds are simulated, and show that R1 and R2 are satisfied. We refer to a message, sent in round i, as a *round-i message.*

In the initialization phase a ⟨ **start** ⟩ message is sent in both directions over every link in the network. Every process resets its local clock to 0 at the moment it sends ⟨ **start** ⟩ messages to all of its neighbors. This is done exactly once in every process. Each process can start its clock and send the messages spontaneously, but must do so at the latest upon receipt of the first ⟨ **start** ⟩ message. Algorithm 1 gives the program for an arbitrary process p.

Because the receiving times of ⟨ **start** ⟩ messages are stored, our algorithm uses more internal storage than the algorithms in [CCGZ90]. By w_p we denote the global time at which p executes $INIT$, and by $C_p^{(t)}$ we denote p's clock reading at global time t. At time w_p, C_p is set to 0, and we assume clocks run accurately. Formally, the clocks satisfy the following *Clock Axiom.*

$$C_p^{(t)} \ = \ t \ - \ w_p. \qquad \text{CA}$$

Let p and q be arbitrary neighbors in the network and σ and τ the global time of sending and receipt of the ⟨ **start** ⟩ message that q sends to p. We have $\sigma = w_q$

and $w_p \leq \tau$ by virtue of the algorithm, and $\sigma \leq \tau < \sigma + 1$ by the Bounded Delay assumption BD. It follows that $w_p \leq w_q + 1$.

At time τ, δ_{pq} is set to $C_p^{(\tau)} = \tau - w_p$. It follows that, as a result of the *Initialization Phase*, the δ's and w's satisfy

$$0 \leq \delta_{pq}, \quad w_q - w_p \leq \delta_{pq} < w_q - w_p + 1. \qquad \text{IP}$$

Because a $\langle\,$**start**$\,\rangle$ message is sent from p to q also, the same holds with p and q interchanged. It follows that $|w_p - w_q| < 1$ and $\delta_{pq} < 2$.

The simulated algorithm operates in rounds 1, 2, 3,... The synchronizer simulates round i at local time $2i$ as described in Algorithm 1. It will now be shown that Algorithm 1 satisfies requirement R1.

Theorem 1.1 *Round-i messages arrive before the simulation of round* $(i+1)$.

Proof. Assume q sends p a message in round m, and let the global times of sending and receipt of this message be σ and τ, respectively. By virtue of the algorithm $C_q^{(\sigma)} = 2i$. We now have

$$\begin{aligned} C_p^{(\tau)} &= \tau - w_p && \text{(CA)} \\ &< \sigma + 1 - w_p && \text{(BD)} \\ &= w_q + 2i + 1 - w_p && \text{(CA)} \\ &< 2(i+1), && \text{(IP)} \end{aligned}$$

so p simulates round $(i+1)$ later than τ. □

Next it will be shown that the algorithm fulfills requirement R2.

Theorem 1.2 *The round number of a message is correctly determined from its local time of receipt and information from the initialization phase.*

Proof. Assume q sends p a round-i message, and let the global times of sending and receipt of this message be σ and τ, respectively. We now have

$$\begin{aligned} C_p^{(\tau)} - \delta_{pq} &= (\tau - w_p) - \delta_{pq} && \text{(CA)} \\ &> (\sigma - w_p) - (w_q - w_p + 1) && \text{(BD, IP)} \\ &= (w_q + 2i - w_p) - (w_q - w_p + 1) && \text{(CA)} \\ &= 2i - 1. \end{aligned}$$

On the other hand,

$$\begin{aligned} C_p^{(\tau)} - \delta_{pq} &= (\tau - w_p) - \delta_{pq} && \text{(CA)} \\ &< (\sigma + 1 - w_p) - (w_q - w_p) && \text{(BD, IP)} \\ &= (w_q + 2i + 1 - w_p) - (w_q - w_p) && \text{(CA)} \\ &= 2i + 1. \end{aligned}$$

It follows that $i = \lfloor \frac{C_p^{(\tau)} - \delta_{pq} + 1}{2} \rfloor$, hence the messages received in Algorithm 1 are stored under the correct round number. $\square$

Concluding, Algorithm 1 satisfies requirement R1 by Theorem 1.1 and requirement R2 (without using additional information in messages) by Theorem 1.2. The round time of Algorithm 1 is 2.

It is well possible for a process to receive a round-i message before it has itself simulated round i. It is also possible to receive a round-$(i+1)$ message from one neighbor earlier than a round-i message from another neighbor. The data structure in which the messages are stored must provide sufficient flexibility to do so. In the most unfavorable situation, all neighbors of p simulate round $(i+1)$ earlier than p, which may force p to simultaneously buffer all messages sent to p in two consecutive rounds.

1.3 Unidirectional Networks

We now drop the assumption that the network is bidirectional and show how Algorithm 1 can be adapted to this more general situation. In directed networks, the existence of an edge qp does not imply that a $\langle$ **start** $\rangle$ message is sent from p to q.

The modified algorithm uses the same initialization phase as Algorithm 1. After this initialization we have (for every edge qp) $w_p < w_q + 1$ and we find

$$0 \leq \delta_{pq}, \quad w_q - w_p \leq \delta_{pq} < w_q - w_p + 1 \qquad \text{IP}$$

as in Section 1.2. We do not necessarily have $w_q < w_p + 1$, but instead we have $w_q < w_p + d(p, q)$, where $d(p, q)$ denotes the distance from p to q. Define d_m as $(\max_{qp \in E}\ d(p, q))$, and assume d_m to be known by all processes. A process now simulates round i at local time $(d_m + 1) \cdot i$. (The value $d_m + 1$ is known as the *girth* of the network.) It is easily seen that all messages now arrive in time. Because $d_m \geq 1$, the round time of this synchronizer is at least 2, so there is no need for an extra bit in messages. As in Theorem 1.2 it can be shown that

$$(d_m + 1)i - 1 < C_p^{(\tau)} - \delta_{pq} < (d_m + 1)i + 1$$

if τ is the time of receipt of a round-i message from q. Consequently, it suffices to change "2" into "$d_m + 1$" in the routines to receive a basic message and to simulate the next round.

2 Optimality of the Synchronizer

Algorithm 1 has a round time of 2 and uses no extra bit in basic messages, which is a fairly strong result. Yet the question arises whether faster synchronizers exist, i.e., synchronizers with a round time smaller than 2. The results in this section concern synchronizers that use the same initialization phase as Algorithm 1. To determine the time of simulation of a round, we allow that all information gathered in the initialization phase, i.e., the (local) time of receipt of $\langle$ **start** $\rangle$ messages, can be used. We do not allow the use of other information, such as the receipt time of basic messages.

It will turn out that if the round time of a synchronizer is smaller than 2, one bit of extra information in basic messages is necessary (Theorem 2.1) and sufficient (Theorem 2.2) to satisfy requirement R2. If a round time of 2 is acceptable, no additional bit is necessary as is demonstrated by Algorithm 1. Therefore, we concentrate on requirement R1. We develop a mathematical model for the type of synchronizer under consideration, and give a more precise definition of the round time. We shall arrive at the following conclusions.

1. There exist networks (notably, complete networks and stars) that can be synchronized with a round time smaller than 2.

2. There exist networks (notably, rings and cubes) that cannot be synchronized with a round time smaller than 2.

Therefore, for some classes of networks, and for the case that the network topology is unknown to the processes, Algorithm 1 is optimal.

We excluded the use of receipt times of basic messages. The low message complexity of our synchronizer is attractive mainly in message sparse computations, as argued in the introduction of Section 1. In these computations the basic messages will be of little use. A more sophisticated algorithm could derive from these arrival times more accurate information about the differences between the clock readings than given in equation IP, and in some executions arrive at a round time close to 1. On the other hand, if, in a certain execution, all messages over one channel have the same transmission delay, the receipt time of basic messages gives no extra information at all. Thus the "worst case" behavior of this approach is no better than that of not using the receipt time of basic messages.

2.1 A Mathematical Model

In this section we will develop a mathematical model for ABD synchronizers to either improve on the round time of Algorithm 1 or prove its optimality. First the requirement R2 will be dealt with in the following two theorems.

Theorem 2.1 *In a synchronizer with round time smaller than 2, C_p and δ_{pq} are insufficient for p to determine the round number of a message from q.*

Proof. Assume q simulates round i at local time T and round $(i+1)$ at local time $T + 2 - \Delta$, for some $\Delta > 0$.

Now consider first the situation that $w_q = w_p + 1 - \frac{1}{2}\Delta$, q's $\langle$ **start** $\rangle$ message to p has delay 0, and q sends a round-i message that suffers a delay of $1 - \frac{1}{2}\Delta$. Because

$\Delta > 0$, this delay satisfies BD. In this situation, $\delta_{pq} = 1 - \frac{1}{2}\Delta$ and at the moment of receipt of the message by p, $C_p - \delta_{pq} = T + 1 - \frac{1}{2}\Delta$.

Second, consider the situation that $w_q = w_p$, q's $\langle\, \textbf{start}\, \rangle$ message suffers a delay of $1 - \frac{1}{2}\Delta$, and q sends a round-$(i+1)$ message that has delay 0. Again $\delta_{pq} = 1 - \frac{1}{2}\Delta$ and at the moment of receipt of the message by p, $C_p - \delta_{pq} = T + 1 - \frac{1}{2}\Delta$.

Consequently, when $\delta_{pq} = 1 - \frac{1}{2}\Delta$ and a message is received at local time $T + \delta_{pq} + 1 - \frac{1}{2}\Delta$, it is not possible to determine the round number of this message from δ_{pq} and the local clock time. □

We remark that this result cannot be circumvented by making T dependent of δ_{qp}, because in both situations it is possible to choose the delay of p's $\langle\, \textbf{start}\, \rangle$ message such that $\delta_{qp} = 0$. A consequence is that if the round time of a synchronizer is smaller than 2 it is necessary to send extra information in messages; one bit suffices for this purpose.

Theorem 2.2 *If a synchronizer satisfies R1, one bit of extra information per message suffices to satisfy R2 also.*

Proof. Suppose p receives a message from q between the simulation of rounds j and $(j+1)$, and p must determine the round number i of this message. By R1 and because p has simulated round j already, $i \geq j$. Because it is possible that p will send a message to q in round $j+1$ and by R1, q simulates round $(j+2)$ later than p simulates round $(j+1)$, and $i \leq j+1$ follows; hence $j \leq i \leq j+1$. Let *par* be the parity of i and assume q included *par* in the message. Now p can compute i using

$$\textbf{if } par = par(j) \textbf{ then } i := j \textbf{ else } i := j+1.$$

So it suffices to include the parity of round numbers in messages of the simulated algorithm. □

Observe that p determines the round numbers without using δ_{pq}. Alg. 1 and the two theorems justify our claim that no additional information is necessary if the round time is at least 2, and one bit suffices if the round time is smaller than 2.

We shall, from now on, concentrate on the problem to fulfil requirement R1. Let $G = (V, E)$ be an undirected graph. A synchronizer, using only information gathered in the initialization phase, is modeled by a synchronizer function, whose arguments are the round number and this information.

Definition 2.3 *A* synchronizer function *F for G is a collection of functions*

$$F_p : \mathbb{N} \times [0,2)^d \to \mathbb{R} \qquad \textit{for each } p \in V,$$

where d is the degree of p in G.

The interpretation of a synchronizer function F is as follows. If p has received $\langle\, \textbf{start}\, \rangle$ messages of its neighbors $q_1, \ldots, q_d$ at local times $\delta_{pq_1}, \ldots, \delta_{pq_d}$, then p simulates round i at local time $F_p(i, \delta_{pq_1}, \ldots, \delta_{pq_d})$. Henceforth we write $\vec{\delta_p}$ for $\delta_{pq_1}, \ldots, \delta_{pq_d}$.

Definition 2.4 *A scenario* for G is a $(|V| + |E|)$-tuple $[w_p : p \in V\,;\ \delta_{pq} : qp \in E]$ such that w_p, $\delta_{pq} \in \mathbb{R}$, and for all $qp \in E$

$$\max(0,\ w_q - w_p) \leq \delta_{pq} < w_q - w_p + 1. \qquad \text{SA}$$

Theorem 2.5 *An ABD synchronizer satisfies requirement R1 if and only if its synchronizer function F satisfies, for every scenario S, every link qp, and every i:*

$$F_p(i+1,\ \vec{\delta_p}) + w_p - F_q(i,\ \vec{\delta_q}) - w_q - 1 \geq 0. \qquad \text{CC}$$

Proof. The rationale behind the proof is that scenarios correspond exactly to possible executions of the initialization phase of the synchronizer, and the correctness criterion CC corresponds to the requirement that a round-i message from q to p arrives in time.

Suppose F satisfies CC (for every S, qp, and i). Consider a message sent from q to p in round i. This message is sent at global time $w_q + F_q(i,\ \vec{\delta_q})$ and hence, by BD, it is received before $w_q + F_q(i,\ \vec{\delta_q}) + 1$. Process p simulates round $i+1$ at global time $w_p + F_p(i+1, \vec{\delta_p})$. From Section 1.2, notably, equation IP, we know that all w and δ obtained during the initialization phase satisfy SA, i.e., $[w_p : p \in V\,;\ \delta_{pq} : qp \in E]$ is a legal scenario. But then, by CC,

$$w_p + F_p(i+1,\ \vec{\delta_p}) \geq w_q + F_q(i,\ \vec{\delta_q}) + 1,$$

which shows that the message arrives in time.

Suppose CC is not satisfied for some scenario $S = [w_p : p \in V\,;\ \delta_{pq} : qp \in E]$, some edge qp, and some i, i.e.,

$$w_p + F_p(i+1,\ \vec{\delta_p}) < w_q + F_q(i,\ \vec{\delta_q}) + 1.$$

Construct the following execution of the synchronizer. Process p awakes spontaneously at time w_p, the $\langle$ **start** $\rangle$ message over edge qp arrives at global time $w_p + \delta_{pq}$. By SA, all $\langle$ **start** $\rangle$ messages satisfy the BD axiom in this execution, and each process executes INIT no later than at the receipt of the first $\langle$ **start** $\rangle$ message. Process q may send to p a basic message in round i, i.e., at global time $w_q + F_q(i,\ \vec{\delta_q})$. The message delay can be arbitrarily close to 1, so tis message may arrive *later* than at global time $w_p + F_p(i+1,\ \vec{\delta_p})$ (but still *before* $w_q + F_q(i,\ \vec{\delta_q}) + 1$). Thus the message arrives too late, violating R1. $\square$

Thus correct synchronizers correspond with synchronizer functions satisfying CC for all S, all qp, and all i. In the sequel, when we say a function satisfies CC we mean that this is the case for all S, all qp, and all i. We also simply say that the function is correct in this case. We can now give a precise definition of the round time of a synchronizer.

Definition 2.6 *For a synchronizer function F, the* round time *of F is*

$$\alpha(F) = \max_{p \in G} \sup_{\vec{\delta_p}} \lim_{i \to \infty} \frac{F_p(i,\, \vec{\delta_p})}{i}.$$

The correctness of Algorithm 1 can be easily demostrated in this model.

Theorem 2.7 *The synchronizer function F with $F_p(i,\, \vec{\delta_p}) = 2i$ is correct.*

Proof. For all S, qp, i we have

$$\begin{aligned} F_p(i+1,\, \vec{\delta_p}) &+ w_p - F_q(i,\, \vec{\delta_q}) - w_q - 1 \\ &= 2i + 2 + w_p - 2i - w_q - 1 && \text{(Def. } F\text{)} \\ &= w_p - w_q + 1 \\ &\geq 0. && \text{(SA)} \end{aligned}$$

□

This function clearly has a round time of 2.

2.2 Fast Synchronization

In this section we show that some networks can be synchronized with a round time smaller than 2. First we consider the network $\mathbf{K}_2$ consisting of two processors p and q, connected by a bidirectional edge pq.

Theorem 2.8 *There exists a synchronizer function for $\mathbf{K}_2$, satisfying CC, with a round time of $1\frac{1}{2}$.*

Proof. Take $F_p(i,\, \delta_{pq}) = 1\frac{1}{2}i + \frac{1}{2}\delta_{pq}$ and $F_q(i,\, \delta_{qp}) = 1\frac{1}{2}i + \frac{1}{2}\delta_{qp}$. For all S, i,

$$\begin{aligned} F_p(i+1,\, \vec{\delta_p}) &+ w_p - F_q(i,\, \vec{\delta_q}) - w_q - 1 \\ &= 1\tfrac{1}{2}(i+1) + \tfrac{1}{2}\delta_{pq} + w_p - 1\tfrac{1}{2}i - \tfrac{1}{2}\delta_{qp} - w_q - 1 && \text{(Def. } F\text{)} \\ &> 1\tfrac{1}{2} + \tfrac{1}{2}(w_q - w_p) + w_p - \tfrac{1}{2}(w_p - w_q + 1) - w_q - 1 && \text{(SA)} \\ &= 0 \end{aligned}$$

The proof for the reverse direction is similar by symmetry. □

Theorem 2.9 *A round time of $1\frac{1}{2}$ is optimal for $\mathbf{K}_2$.*

Proof. Let F satisfy CC. For any $\rho \in (0, \frac{1}{2})$, let S_ρ be the scenario where $w_q = w_p + \frac{1}{2} - \rho$, $\delta_{pq} = \delta_{qp} = \frac{1}{2}$. F satisfies CC for S_ρ, so

$$F_p(i+1,\, \frac{1}{2}) \geq F_q(i,\, \frac{1}{2}) + 1\frac{1}{2} - \rho.$$

This holds for all $\rho > 0$, and thus

$$F_p(i+1,\,\frac{1}{2}) \geq F_q(i,\,\frac{1}{2}) + 1\frac{1}{2}$$

follows. Repeat this argument with p and q interchanged and find $F_q(i+2,\,\frac{1}{2}) \geq F_q(i,\,\frac{1}{2}) + 3$. It follows that $\lim\limits_{i\to\infty} \frac{F_q(i,\frac{1}{2})}{i} \geq \frac{3}{2}$. □

Complete networks. We generalize the results for $\mathbf{K}_2$ in two ways. Let $\mathbf{K}_n$ be the complete network with n nodes, i.e., $V = \{1,\,\ldots,\,n\}$ and $E = \{(p,q) : p \neq q\}$.

Theorem 2.10 *There exists a synchronizer function for* $\mathbf{K}_n$*,satisfying CC, with a round time of* $2 - \frac{1}{n}$.

Proof. Take $F_p(i, \delta_{p1},\,\ldots,\,\delta_{p,n-1}) = (2-\frac{1}{n})i + \frac{1}{n}(\delta_{p1} + \ldots + \delta_{p,n-1})$. We prove CC on edge 21.

$$\begin{array}{rll}
F_1(i+1, & \delta_{12},\,\delta_{13},\ldots) + w_1 - F_2(i,\delta_{21},\,\delta_{23},\ldots) - w_2 - 1 & \\
= & (2-\frac{1}{n})(i+1) + \frac{1}{n}(\delta_{12}+\delta_{13}+\ldots) + w_1 & \text{(Def. } F\text{)} \\
 & \quad - (2-\frac{1}{n})i - \frac{1}{n}(\delta_{21}+\delta_{23}+\ldots) - w_2 - 1 & \\
> & (2-\frac{1}{n}) + \frac{1}{n}((w_2-w_1)+(w_3-w_1)+\ldots) + w_1 & \text{(SA)} \\
 & \quad - \frac{1}{n}((w_1-w_2+1)+(w_3-w_2+1)+\ldots) - w_2 - 1 & \\
= & 0 &
\end{array}$$

The proof for the other edges is similar. □

Theorem 2.11 *A round time of* $(2-\frac{1}{n})$ *is optimal for the* $\mathbf{K}_n$.

Proof. (Assume the arguments of F_p are listed in the order $\delta_{p,p+1}, \delta_{p,p+2}, \ldots$) Let F be a synchronizer function satisfying CC. For $\rho \in (0,\,\frac{1}{n})$, let S_ρ be the scenario where $w_p = \frac{p}{n}$ for $p < n$, $w_n = 1-\rho$, $\delta_{1p} = \frac{p-1}{n}$, $\delta_{np} = \frac{p}{n}$, and $\delta_{qp} = \max(0,\,\frac{p-q}{n})$ for $1 < q < n$. This tuple satisfies SA and, using CC, it follows that

$$F_1(i+1,\,\frac{1}{n},\,\frac{2}{n},\,\ldots) \geq F_n(i,\,\frac{1}{n},\,\frac{2}{n},\,\ldots) + 1 + \frac{n-1}{n} - \rho.$$

This holds for all $\rho > 0$, and thus

$$F_1(i+1,\,\frac{1}{n},\,\frac{2}{n},\,\ldots) \geq F_n(i,\,\frac{1}{n},\,\frac{2}{n},\,\ldots) + 1 + \frac{n-1}{n}$$

follows. Repeat this argument n times, with a cyclic shift of process names, and find

$$F_n(i+n,\,\frac{1}{n},\,\frac{2}{n},\,\ldots) \geq F_n(i,\,\frac{1}{n},\,\frac{2}{n},\,\ldots) + 2n - 1.$$

It follows that $\alpha(F) \geq 2 - \frac{1}{n}$. □

Star Networks. The star network $\mathbf{S}_n$ consists of n nodes p, $q_1, \ldots, q_{n-1}$ and $n-1$ edges $pq_1, \ldots, pq_{n-1}$. Note that $\mathbf{K}_2 = \mathbf{S}_2$. We generalize the results for $\mathbf{K}_2$ to $\mathbf{S}_n$.

Theorem 2.12 *There exists a synchronizer function for $\mathbf{S}_n$ with round time $1\frac{1}{2}$.*

Proof. Take $F_p(i, \vec{\delta_p}) = 1\frac{1}{2}i + \frac{1}{2}$ and $F_{q_j}(i, \delta_{q_jp}) = 1\frac{1}{2}i + \delta_{q_jp}$. Now we have

$$\begin{aligned} F_p(i+1, \vec{\delta_p}) &+ w_p - F_{q_j}(i, \delta_{q_jp}) - w_{q_j} - 1 \\ &= 1\tfrac{1}{2}(i+1) + \tfrac{1}{2} + w_p - 1\tfrac{1}{2}i - \delta_{q_jp} - w_{q_j} - 1 && \text{(Def. } F\text{)} \\ &> 1\tfrac{1}{2} + \tfrac{1}{2} + w_p - (w_p - w_{q_j} + 1) - w_{q_j} - 1 && \text{(SA)} \\ &= 0. \end{aligned}$$

and

$$\begin{aligned} F_{q_j}(i+1, \delta_{q_jp}) &+ w_{q_j} - F_p(i, \vec{\delta_p}) - w_p - 1 \\ &= 1\tfrac{1}{2}(i+1) + \delta_{q_jp} + w_{q_j} - 1\tfrac{1}{2}i - \tfrac{1}{2} - w_p - 1 && \text{(Def. } F\text{)} \\ &\geq 1\tfrac{1}{2} + (w_p - w_{q_j}) + w_{q_j} - \tfrac{1}{2} - w_p - 1 && \text{(SA)} \\ &= 0. \end{aligned}$$

for all S, j, and i, hence F satisfies CC. □

Theorem 2.13 *A round time of $1\frac{1}{2}$ is optimal for $\mathbf{S}_n$.*

Proof. Apply the proof of Theorem 2.9 to any of the edges of $\mathbf{S}_n$. □

We have seen that when the round time is smaller than 2 an extra bit in messages is necessary. The value of δ_{pq} is not needed for determining the round number of a message in this case. In all synchronizers in this section only the sum of δ_{pq} is needed in a process to determine when a next round is simulated. Thus, all synchronizers in this section can be implemented in $O(1)$ internal storage (excluding the space needed for temporary storage of messages of the simulated algorithm).

2.3 Lower Bound Results

In this section we show that a round time of 2 is optimal for rings of size 4 and larger. Theorem 2.18 facilitates the proof. It says that we may assume that a synchronizer function for a ring is identical in each process, and symmetric in its two δ-arguments. Recall that an *automorphism* of G is an isomorphism of G onto itself and $Aut(G)$ is the group of automorphisms of G.

Definition 2.14 *For a synchronizer function F for G, $A \in Aut(G)$, $F \circ A$ is the synchronizer function H defined by*

$$H_p(i, \vec{\delta_p}) = F_{A(p)}(i, \vec{\delta_p}).$$

(The elements of $\vec{\delta_p}$ are reordered according to A.)

Lemma 2.15 *If F satisfies CC, so does $H = F \circ A$, and $\alpha(H) = \alpha(F)$.*

Proof. Fix a scenario S, edge qp, round number i. By definition we have

$$H_p(i+1,\, \vec{\delta_p}) + w_p - H_q(i,\, \vec{\delta_q}) - w_q - 1 = F_{A(p)}(i+1,\, \vec{\delta_p}) + w_p - F_{A(q)}(i,\, \vec{\delta_q}) - w_q - 1.$$

Now consider the scenario S' where $w'_p = w_{A^{-1}(p)}$ and $\delta'_{pq} = \delta_{A^{-1}(p)\,A^{-1}(q)}$. F satisfies CC for this scenario on edge $A(p)A(q)$, i.e.,

$$F_{A(p)}(i+1,\, \vec{\delta'}_{A(p)}) + w'_{A(p)} - F_{A(q)}(i,\, \vec{\delta'}_{A(q)}) - w'_{A(q)-1} \geq 0.$$

But then

$$H_p(i+1,\, \vec{\delta_p}) + w_p - H_q(i,\, \vec{\delta_q}) - w_q - 1 \geq 0.$$

The second part of the lemma is trivial. □

Definition 2.16 *For synchronizer functions F_1 and F_2, $\sigma_1,\, \sigma_2 \in \mathbb{R}$, $\sigma_1 F_1 + \sigma_2 F_2$ is the synchronizer function H defined by*

$$H_p(i,\, \vec{\delta_p}) = \sigma_1 F_{1,p}(i,\, \vec{\delta_p}) + \sigma_2 F_{2,p}(i,\, \vec{\delta_p}).$$

Lemma 2.17 *If F_1 and F_2 satisfy CC, $\sigma_1,\, \sigma_2 \geq 0$, $\sigma_1 + \sigma_2 = 1$, then $H = \sigma_1 F_1 + \sigma_2 F_2$ satisfies CC and $\alpha(H) \leq \max(\alpha(F_1),\, \alpha(F_2))$.*

Proof. For every S, qp, i, we have

$$\begin{aligned} H_p(i+1,\, \vec{\delta_p}) &+ w_p - H_q(i,\, \vec{\delta_q}) - w_q - 1 \\ &= \sigma_1(F_{1,p}(i+1,\, \vec{\delta_p}) + w_p - F_{1,q}(i,\, \vec{\delta_q}) - w_q - 1) + \\ &\qquad \sigma_2(F_{2,p}(i+1,\, \vec{\delta_p}) + w_p - F_{2,q}(i,\, \vec{\delta_q}) - w_q - 1) \\ &\geq 0 + 0 \end{aligned}$$

because F_1 and F_2 satisfy CC and $\sigma_1,\, \sigma_2 \geq 0$. Furthermore, max, sup, and lim commute with multiplication by a constant and distribute over addition in the following sense:

$$\max(T_1 + T_2) \leq \max(T_1) + \max(T_2).$$

(And similar for sup and lim.) It follows that $\alpha(H) \leq \sigma_1 \alpha(F_1) + \sigma_2 \alpha(F_2)$. □

It will now be shown that for each synchronizer function F, it is possible to construct a "symmetric" function that is at least as good as F in terms of round time.

Theorem 2.18 *For any correct synchronizer function F there is a correct synchronizer function H such that*

(i) *$\alpha(H) \leq \alpha(F)$; and*
(ii) *for all $A \in Aut(G)$, $H = H \circ A$.*

Proof. Let F be given. Take $k = |Aut(G)|$ and define $H = \sum_{B \in Aut(G)} \frac{1}{k}(F \circ B)$. By Lemmas 2.15 and 2.17, H is again correct and $\alpha(H) \leq \alpha(F)$. Furthermore, for $A \in Aut(G)$,

$$\begin{aligned} H \circ A &= (\textstyle\sum_{B \in Aut(G)} \frac{1}{k}(F \circ B)) \circ A \\ &= \textstyle\sum_{B \in Aut(G)} \frac{1}{k}(F \circ B \circ \ A) \\ &= H, \end{aligned}$$

because $Aut(G) \circ A = Aut(G)$. □

Lower Bounds for the Ring and Hypercube. The network $\mathbf{R}_n$ has n nodes $1, ..., n$, and n edges $(p,\, p+1)$, where indices are counted modulo n. Note that bidirectional rings are considered here.

Theorem 2.19 *A round time of 2 is optimal for* $\mathbf{R}_4$.

Proof. Let F be a correct synchronizer function for $\mathbf{R}_4$. By Theorem 2.18 we may assume that each process p has the same local function $F_p = F$ and that this function is symmetric in its two δ-arguments. For $\rho \in (0,\, 1)$, let S_ρ be the scenario where

$$\begin{array}{lll} w_1 = 0, & \delta_{12} = 1, & \delta_{14} = 0, \\ w_2 = 1 - \rho, & \delta_{23} = 1, & \delta_{21} = 0, \\ w_3 = 1 - \frac{1}{2}\rho, & \delta_{34} = 0, & \delta_{32} = \frac{1}{2}\rho, \\ w_4 = 0, & \delta_{41} = 0, & \delta_{43} = 1. \end{array}$$

These values are according to SA, and because F satisfies CC for this scenario on edge 21 we have $F(i+1,\, 0,\, 1) \geq F(i,\, 0,\, 1) + 2 - \rho$. Again this holds for all $\rho > 0$, and $F(i+1,\, 0,\, 1) \geq F(i,\, 0,\, 1) + 2$ follows. Thus $\alpha(F) \geq 2$. □

Theorem 2.20 *A round time of 2 is optimal for* $\mathbf{R}_n$*, if* $n > 4$.

Proof. As the previous theorem. Extend scenario S_ρ as given in the proof of Theorem 2.19 to a scenario for $\mathbf{R}_n$ with

$$\begin{array}{llll} & \delta_{1n} = 0, & \delta_{45} = 0, & \\ w_i = 0, & \delta_{i,i-1} = 0, & \delta_{i,i+1} = 0 & \text{for } i > 4. \end{array}$$

□

The results for rings can be easily extended to Hypercubes, because each two–dimensional face of the Hypercube is a ring. The 2^N hypercube s the graph $\mathbf{C}_N = (V,\, E)$, where $V = \{0,\, 1\}^N$, and $E = \{(p, q) \in V^2 :\ p \text{ and } q \text{ differ in one bit}\}$.

Theorem 2.21 *A round time of 2 is optimal for* $\mathbf{C}_N$ *if* $N \geq 2$.

Proof. Modify the proof of Theorem 2.19 for any surface of the cube. □

Summary. Because $\mathbf{R}_3 = \mathbf{K}_3$ and $\mathbf{C}_1 = \mathbf{R}_2 = \mathbf{K}_2$, we have now determined the optimal round times for all rings, stars, complete networks, and cubes. If a synchronizer with a round time of 2 is used, there are two options to satisfy requirement R2. A bit can be added to messages as described in the proof of Theorem 2.2. In this case the δ_{pq} need not be stored during the simulation and the synchronizer can be implemented in $O(1)$ storage per process. The other option is to use Algorithm 1. Then no extra bit is necessary, but the internal storage in a process equals its degree in the network.

3 Drifting Clocks

Until now we have assumed that clocks run accurately. In this section we will develop synchronizers for the more realistic case where clocks may suffer a small, bounded drift. By an *ϵ-bounded drift* we mean that it takes a clock at least $(1-\epsilon)\Delta$ and at most $(1+\epsilon)\Delta$ global time to advance an amount Δ. In other words, we replace the clock axiom CA by CA-ϵ:

$$(1+\epsilon)^{-1}(t - w_p) \leq C_p^{(t)} \leq (1-\epsilon)^{-1}(t - w_p). \qquad \text{CA-}\epsilon$$

The constant ϵ is known from the specification of the underlying hardware clocks. Typically ϵ is very small, in the order of 10^{-5} or 10^{-6}. We adhere to the original bounded delay axiom BD.

3.1 A Linear Algorithm

In this section we will present an algorithm that resembles Algorithm 1. Round i is simulated at local time αi for some $\alpha > 2$. It will turn out, as in [CCGZ90], that after a finite number of rounds a new execution of the initialization phase is necessary. The initialization phase of this algorithm (and the algorithm in Section 3.2) is the same as for Algorithm 1. As in Section 1.2, we find that after initialization $w_p < w_q + 1$ if edge qp exists. For δ_{pq} we have again $\delta_{pq} \geq 0$ and, using CA-ϵ instead of CA,

$$0 \leq \delta_{pq}, \quad (1+\epsilon)^{-1}(w_q - w_p) \leq \delta_{pq} < (1-\epsilon)^{-1}(w_q - w_p + 1). \qquad \text{IP-}\epsilon$$

As mentioned above, round i is simulated at time αi.

Theorem 3.1 *All round-i messages arrive in time for the simulation of round $(i+1)$ if*

$$(i+1) \leq \frac{(1+\epsilon)\alpha - 2}{2\epsilon\alpha}. \qquad I1$$

Proof. Again let σ, τ be the times of sending and receipt of a round-i message from q to p. By virtue of the algorithm we have $C_q^{(\sigma)} = \alpha i$. Thus

$$\begin{aligned} C_p^{(\tau)} &\le (1-\epsilon)^{-1}(\tau - w_p) && \text{(CA-}\epsilon\text{)} \\ &< (1-\epsilon)^{-1}(\sigma + 1 - w_p) && \text{(BD)} \\ &\le (1-\epsilon)^{-1}((1+\epsilon)\alpha i + w_q + 1 - w_p) && \text{(CA-}\epsilon\text{)} \\ &< (1-\epsilon)^{-1}((1+\epsilon)\alpha i + 2). && \text{(IP-}\epsilon\text{)} \end{aligned}$$

Round $(i+1)$ is simulated by p when $C_p = \alpha(i+1)$; clearly the message is in time if

$$(1-\epsilon)^{-1}((1+\epsilon)\alpha i + 2) \le \alpha(i+1),$$

and this is equivalent to I1. □

Theorem 3.2 *The round number of a message can be determined using its local time of receipt and information from the initialization phase if*

$$i \le \frac{(1+\epsilon)^2\alpha - 2(1+3\epsilon)}{4\epsilon\alpha}. \qquad \textit{I2}$$

Proof. Let σ, τ be the times of sending and receipt of a round-$(i-1)$ message; then

$$\begin{aligned} C_p^{(\tau)} - \delta_{pq} &\le (1-\epsilon)^{-1}(\tau - w_p) - (1+\epsilon)^{-1}(w_q - w_p) && \text{(CA-}\epsilon\text{, IP-}\epsilon\text{)} \\ &< (1-\epsilon)^{-1}(\sigma + 1 - w_p) - (1+\epsilon)^{-1}(w_q - w_p) && \text{(BD)} \\ &\le (1-\epsilon)^{-1}((1+\epsilon)\alpha(i-1) + w_q + 1 - w_p) && \text{(CA-}\epsilon\text{)} \\ &\qquad - (1+\epsilon)^{-1}(w_q - w_p) \\ &= \tfrac{1+\epsilon}{1-\epsilon}\alpha(i-1) + \tfrac{2\epsilon}{1-\epsilon^2}(w_q - w_p) + \tfrac{1}{1-\epsilon} \\ &\le \tfrac{1+\epsilon}{1-\epsilon}\alpha(i-1) + \tfrac{2\epsilon}{1-\epsilon^2} + \tfrac{1}{1-\epsilon} && \text{(IP-}\epsilon\text{)} \\ &= \tfrac{1+\epsilon}{1-\epsilon}\alpha(i-1) + \tfrac{1+3\epsilon}{1-\epsilon^2}. \end{aligned}$$

On the other hand, if σ, τ are the times of sending and receipt of a round-i message we have

$$\begin{aligned} C_p^{(\tau)} - \delta_{pq} &> (1+\epsilon)^{-1}(\tau - w_p) - (1-\epsilon)^{-1}(w_q - w_p + 1) && \text{(CA-}\epsilon\text{, IP-}\epsilon\text{)} \\ &\ge (1+\epsilon)^{-1}(\sigma - w_p) - (1-\epsilon)^{-1}(w_q - w_p + 1) && \text{(BD)} \\ &\ge (1+\epsilon)^{-1}((1-\epsilon)\alpha i + w_q - w_p) && \text{(CA-}\epsilon\text{)} \\ &\qquad - (1-\epsilon)^{-1}(w_q - w_p + 1) \\ &= \tfrac{1-\epsilon}{1+\epsilon}\alpha i - \tfrac{2\epsilon}{1-\epsilon^2}(w_q - w_p) - \tfrac{1}{1-\epsilon} \\ &> \tfrac{1-\epsilon}{1+\epsilon}\alpha i - \tfrac{2\epsilon}{1-\epsilon^2} - \tfrac{1}{1-\epsilon} && \text{(IP-}\epsilon\text{)} \\ &= \tfrac{1-\epsilon}{1+\epsilon}\alpha i - \tfrac{1+3\epsilon}{1-\epsilon^2}. \end{aligned}$$

So a process can distinguish a round-$(i-1)$ from a round-i message if

$$\frac{1+\epsilon}{1-\epsilon}\alpha(i-1)+\frac{1+3\epsilon}{1-\epsilon^2}\leq\frac{1-\epsilon}{1+\epsilon}\alpha i-\frac{1+3\epsilon}{1-\epsilon^2}$$

and this is equivalent to I2. □

The reader may verify that $\frac{(1+\epsilon)^2\alpha-2(1+3\epsilon)}{4\epsilon\alpha}\leq\frac{(1+\epsilon)\alpha-2}{2\epsilon\alpha}$ (use $\epsilon\leq 1$ and $\alpha\geq 2$), hence I2 implies I1. With a fixed α, we can either simulate the number of rounds given by I1, and use an extra bit for recognizing messages, or simulate the (smaller) number of rounds given by I2 and use no extra bit. After this number of rounds the initialization phase must be executed again to simulate more rounds. In the first case the synchronizer can be implemented in $O(1)$ storage per process, in the second case storage in a process equals its degree in the network.

To get a feeling of the values actually involved, and compare this algorithm with the algorithm in [CCGZ90], we include an example computation. Assume the timers may drift a tenth of a second a day, which makes $\epsilon=\frac{1}{864000}$, and set $\alpha=7$. Using I1 we find that 308571 rounds can be simulated before reinitialization is necessary if an extra bit is used in messages. Using I2 we find that 154286 rounds can be simulated before reinitialization is necessary if no extra bit is used. The algorithm of [CCGZ90] simulates 142045 rounds when $\alpha=8$, using no extra bit. (We compare with $\alpha=8$ because already without drift the algorithm of [CCGZ90] has a round time one higer than ours, and here we aim to compute the effect of drift.)

3.2 An Exponential Algorithm

In this section we develop a faster algorithm to synchronize ABD networks with drifting clocks. No reinitialization will be necessary at all during simulation. The initialization phase is again the same as for Algorithm 1. We postulate that round i is simulated at local time $f(i)$, but do not assume, as in Section 3.1, that f is a linear function.

Theorem 3.3 *All round-$(i-1)$ messages will arrive in time if*

$$f(i)\geq a_1 f(i-1)+b_1, \qquad J1$$

where $a_1=\frac{1+\epsilon}{1-\epsilon}$ and $b_1=\frac{2}{(1-\epsilon)}$.

Proof. Again let σ,τ be the times of sending and receipt of a round-$(i-1)$ message from q to p. By virtue of the algorithm we have $C_q^{(\sigma)}=f(i-1)$. Thus

$$\begin{array}{rlr}
C_p^{(\tau)} & \leq (1-\epsilon)^{-1}(\tau-w_p) & (\text{CA-}\epsilon)\\
& < (1-\epsilon)^{-1}(\sigma+1-w_p) & (\text{BD})\\
& \leq (1-\epsilon)^{-1}((1+\epsilon)f(i-1)+w_q+1-w_p) & (\text{CA-}\epsilon)\\
& < (1-\epsilon)^{-1}((1+\epsilon)f(i-1)+2) & (\text{IP-}\epsilon)\\
& = a_1 f(i-1)+b_1. &
\end{array}$$

So the message is clearly in time if $f(i) \geq a_1 f(i-1) + b_1$. $\square$

Theorem 3.4 *The round number of a message can be determined using its local time of receipt and information from the initialization phase if*

$$f(i) \geq a_2 f(i-1) + b_2, \tag{J2}$$

where $a_2 = \left(\frac{1+\epsilon}{1-\epsilon}\right)^2$ *and* $b_2 = \frac{2+6\epsilon}{(1-\epsilon)^2}$.

Proof. Let σ, τ be as above; then

$$\begin{aligned}
C_p^{(\tau)} - \delta_{pq} &\leq (1-\epsilon)^{-1}(\tau - w_p) - (1+\epsilon)^{-1}(w_q - w_p) && \text{(CA-}\epsilon\text{, IP-}\epsilon\text{)} \\
&< (1-\epsilon)^{-1}(\sigma + 1 - w_p) - (1+\epsilon)^{-1}(w_q - w_p) && \text{(BD)} \\
&\leq (1-\epsilon)^{-1}((1+\epsilon)f(i-1) + w_q + 1 - w_p) && \text{(CA-}\epsilon\text{)} \\
&\qquad - (1+\epsilon)^{-1}(w_q - w_p) \\
&= \tfrac{1+\epsilon}{1-\epsilon} f(i-1) + \tfrac{2\epsilon}{1-\epsilon^2}(w_q - w_p) + \tfrac{1}{1-\epsilon} \\
&\leq \tfrac{1+\epsilon}{1-\epsilon} f(i-1) + \tfrac{2\epsilon}{1-\epsilon^2} + \tfrac{1}{1-\epsilon} && \text{(IP-}\epsilon\text{)} \\
&= \tfrac{1+\epsilon}{1-\epsilon} f(i-1) + \tfrac{1+3\epsilon}{1-\epsilon^2}.
\end{aligned}$$

On the other hand, for a round-i message we have

$$\begin{aligned}
C_p^{(\tau)} - \delta_{pq} &> (1+\epsilon)^{-1}(\tau - w_p) - (1-\epsilon)^{-1}(w_q - w_p + 1) && \text{(CA-}\epsilon\text{, IP-}\epsilon\text{)} \\
&\geq (1+\epsilon)^{-1}(\sigma - w_p) - (1-\epsilon)^{-1}(w_q - w_p + 1) && \text{(BD)} \\
&\geq (1+\epsilon)^{-1}((1-\epsilon)f(i) + w_q - w_p) && \text{(CA-}\epsilon\text{)} \\
&\qquad - (1-\epsilon)^{-1}(w_q - w_p + 1) \\
&= \tfrac{1-\epsilon}{1+\epsilon} f(i) - \tfrac{2\epsilon}{1-\epsilon^2}(w_q - w_p) - \tfrac{1}{1-\epsilon} \\
&> \tfrac{1-\epsilon}{1+\epsilon} f(i) - \tfrac{2\epsilon}{1-\epsilon^2} - \tfrac{1}{1-\epsilon} && \text{(IP-}\epsilon\text{)} \\
&= \tfrac{1-\epsilon}{1+\epsilon} f(i) - \tfrac{1+3\epsilon}{1-\epsilon^2}.
\end{aligned}$$

So a process can distinguish a round-$(i-1)$ message from a round-i message if

$$\frac{1-\epsilon}{1+\epsilon} f(i) - \frac{1+3\epsilon}{1-\epsilon^2} \geq \frac{1+\epsilon}{1-\epsilon} f(i-1) + \frac{1+3\epsilon}{1-\epsilon^2},$$

or $f(i) \geq a_2 f(i-1) + b_2$. $\square$

Because $a_2 \geq a_1$ and $b_2 \geq b_1$, again J2 implies J1. We note that the function $f(i) = b\frac{a^i-1}{a-1}$ satisfies $f(i) = af(i-1) + b$. Hence, we can use the function

$$f_1(i) = b_1 \frac{a_1^i - 1}{a_1 - 1}$$

and use an extra bit for recognizing messages, or use

$$f_2(i) = b_2 \frac{a_2^i - 1}{a_2 - 1}$$

and no extra bit. These functions are exponential in i and thus have an unbounded round time. Yet, for all values for which they can be compared with the functions in Section 3.1, they perform better.

First consider the case where an extra bit is used in messages. In the previous section, using $\alpha = 7$, reinitialization was necessary after the 308571th round. This round is simulated at time $7 \times 308571 = 2159997$. The synchronizer in this section simulates this round at local time $f_1(308571) = 900915$. With no extra bit, reinitialization was necessary after the 154286th round. This round is simulated at time $7 \times 154286 = 1080002$. The synchronizer in this section simulates this round at local time $f_2(154286) = 450461$.

4 Conclusions

In this article we have studied a class of synchronizers for ABD networks. The synchronizers in this class use extra control messages only during the initialization phase. Our starting point was a simple synchronizer by Chou *et al.* [CCGZ90]. In Section 1.2 we improved on this synchronizer: in our version no extra bit in a message is necessary to determine its round number.

We studied the effect of three changes in the model on the synchronizer algorithm and its performance. In Section 1.3 we studied unidirectional networks. In Section 2 we made the (local) time of simulation of a round dependent on the (local) time of receipt of messages in the initialization phase. It was proved that this improves performance of the synchronizer in stars and complete networks, but not in rings and cubes. In Section 3 we studied the effect of drift of the local clocks and showed that a (slower) synchronization is still possible.

In our models we considered the effect of these three changes separately. Of course it is possible to make a (more complex) mathematical model including unidirectional networks, use of local receipt times, and drifting clocks at the same time. We conjecture that no new conclusions are found in this way.

After the initialization phase a process knows the start-time of a neighbor's clock to be within a certain interval of length 1. Using the receipt time of more messages the length of this "uncertainty interval" may be decreased. If this information is spread over the network and used in a proper way, the round time could be decreased also. We did not study synchronizers using this principle.

There is an intimate relation between the problem of synchronizing an ABD network and the problem of clock synchronization. Assume the clocks can be synchronized within Δ, i.e., at any moment t we have $|C_p^{(t)} - C_q^{(t)}| < \Delta$. It is easy to see that a process can now simulate round i at local time $(1 + \Delta)i$, and R1 and R2 are satisfied. In [LL84] it is shown that clocks in the $\mathbf{K}_n$ cannot be synchronized tighter than within $1 - \frac{1}{n}$. This corresponds with our results in Section 2.2.

References

[Awe85] Awerbuch, B. Complexity of network synchronization. *J. ACM* **32** (1985), 804–823.

[CCGZ90] Chou, C. T., Cidon, I., Gopal, I. S., and Zaks, S. Synchronizing asynchronous bounded delay networks. *IEEE Trans. Commun.* **38**, 2 (1990), 144–147.

[LL84] Lundelius, J., and Lynch, N. A. An upper and lower bound for clock synchronization. *Information and Control* **62** (1984), 190–204.

[Vit85] Vitányi, P. M. B. Time-driven algorithms for distributed control. Tech. Rep. CS–R8510, Centre for Mathematics and Computer Science, Amsterdam, 1985.

Contents