Optimal Synchronization of ABD Networks

Ephraim Korach, Gerard Tel, Shmuel Zaks

RUU–CS–88–23
May 1988

Optimal Synchronization of ABD Networks

Ephraim Korach, Gerard Tel, Shmuel Zaks

Technical Report RUU-CS-88-[illegible]
May 1988

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

Optimal Synchronization of ABD Networks

Ephraim Korach†, Gerard Tel‡, Shmuel Zaks†

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

Abstract: We present a simple and efficient synchronizer for ABD networks, asynchronous networks with bounded delay message delivery. The algorithms improve on an earlier algorithm [Ch87]. Using a mathematical model for this type of synchronizer we achieve optimality results for ABD synchronizers.

1 Introduction

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 cycles. The actions of a process in cycle $(i+1)$ depend on its state after cycle i and the messages sent to it in cycle i. Note that it is therefore necessary that all messages that are sent to some process in cycle i are received before the process starts its computation of cycle $(i+1)$. We can think of the system as if there is a global clock, giving pulses at regular intervals. Computation takes place at clock pulses, and a message, 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 synchronous model is stronger than the asynchronous model. Consequently, distributed algorithms for synchronous networks are more efficient than algorithms for asynchronous networks. On the other hand, asynchronous networks are easier to build and have a potentially higher performance. Therefore simulation algorithms have been designed to simulate synchronous algorithms on asynchronous networks. These simulation algorithms are called *synchronizers* [Aw85]. The simplest of these mechanism ensures that exactly one message is sent over each link in the network in every cycle. If the simulated algorithm sends more messages over some link in some cycle, these messages must be packed in one larger logical message. If the simulated algorithm sends no messages over some link in some cycle, a special "empty message" must be sent. As a result of this policy, every process must receive exactly one

This work was partially carried out when the second author visited the Technion in December, 1987.

† Address of author: Department of Computer Science, The Technion, Haifa 32000, ISRAEL.

‡ The work of the this author is supported by the Foundation for Computer Science (SION) of the Netherlands Organization for Scientific Research (NWO).

message from every neighbor after every cycle. The next cycle is simulated when the message of the current cycle has been received from all neighbors.

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 edges and diameter D. A simple synchronous algorithm uses E messages and time D. When the simple synchronizer is used to simulate this algorithm, $E \cdot D$ 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 [Vi85]. These algorithms use a number of messages linear in the number of edges, but take exponential time. The simple synchronizer would increase the message complexity to exponential.

Recently Chou *et al.* [Ch87] introduced a new 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. 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.

Formally, if σ is the global time of the sending of a message, and τ is the global time of its receipt, then

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

In our analysis we will always refer to a global clock time, but this global clock 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 START messages is required to make every process starts its local clock at approximately the same time. After this initialization phase a processor will use its clock to decide when the next cycle of the simulated algorithm is executed. The following two requirements must be satisfied:

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

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

Requirement R1 is obvious because p's actions in cycle $(i+1)$ depend on q's message. Failure to meet requirement R2 may lead to incorrect simulation as was noted by Lakshmanan and Thulasiraman [TL87]. If the two requirements are satisfied each step of the synchronous algorithm is simulated correctly.

To compare the speed of synchronizers we introduce the concept of *cycle time*. The cycle time of a synchronizer is the time it takes to simulate one cycle of the synchronous algorithm. When the simple synchronizer, described earlier, is used on an ABD Network, it realizes a cycle time of 1. No mechanism can have a smaller cycle 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.* [Ch87] presented two synchronizers. Their first synchronizer has a cycle time of 2. To meet requirement R2, one bit is added to every message of the simulated algorithm. This extra bit is avoided in the second synchronizer, but this is paid for with a cycle time of 3. In this paper we present a synchronizer with a cycle time of 2, without the extra bit. This clearly improves on the results of [Ch87].

It remained an open question, whether a cycle time of 2 is optimal for ABD synchronizers. We develop a mathematical model and answer this question in the affirmative as well as in the negative. Depending on the topology of the network, some networks can be synchronized with a cycle 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 [Ch87].

This paper is organized as follows. In section 2 we present our ABD synchronizer. In section 3 we develop our mathematical model. In section 4 we show that some networks can be synchronized with a cycle time smaller than 2. In section 5 we show that for some networks 2 is optimal. In sections 6 and 7 we present two synchronizers for the case where clocks suffer drift. In section 8 we consider directed networks. In section 9 we offer final comments and suggestions for further research.

2 The Synchronizer

We first describe the initialization phase, explain when cycles are simulated, and show that R1 and R2 are satisfied. Throughout this paper we assume the network to be bidirectional. This means that for every link pq there is a link qp in the reverse direction also. We refer to a message, sent in cycle i, as a *cycle-i message*.

In the initialization phase a START message is sent 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. We give the program for an arbitrary process p. Initially the value of $started_p$ is $false$.

```
procedure INIT:
    (* Executed spontaneously (if $\neg started_p$)
        or upon receipt of the first START message *)
    begin $CLOCK_p := 0$ ; $started_p := true$ ;
        send START to every neighbor
    end

upon receipt of START from neighbor $q$ do
    begin if not $started_p$ then INIT ;
        $\delta_{pq} := CLOCK_p$
    end
```

The receiving times of START messages is stored, thus our algorithm uses more internal storage than the algorithms in [Ch87]. By w_p we denote the global time at which p executes *INIT*, and by $CLOCK_p^{(t)}$ we denote p's clock reading at global time t. At time w_p, $CLOCK_p$ is set to 0, and we assume clocks run accurately. We adopt the following Clock Axiom:

$$CLOCK_p^{(t)} = t - w_p \tag{CA}$$

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 q sends to p. We have $\sigma = w_q$ and $w_p \le \tau$ by virtue of the algorithm, and $\sigma \le \tau < \sigma+1$ by the Bounded Delay assumption BD. It follows that $w_p < w_q + 1$.

At time τ, δ_{pq} is set to $CLOCK_p^{(\tau)} = \tau - w_p$. It follows that

$$0 \le \delta_{pq}, \;\; w_q - w_p \le \delta_{pq} < w_q - w_p + 1. \tag{IP}$$

We refer to this equation as IP because it is the result of the Initialization Phase. Because we have a link 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 cycles 1, 2, 3,... The synchronizer simulates cycle i at local time $2i$:

```
when $CLOCK_p = 2i$ do
    execute cycle $i$, using the cycle-$(i-1)$ messages stored so far
```

Theorem 2.1: Cycle-$(i-1)$ messages arrive before the simulation of cycle i.

Proof: Assume q sends p a message in cycle $(i-1)$, and let the global time of sending and receipt of this message be σ and τ, respectively. By virtue of the algorithm $CLOCK_q^{(\sigma)} = 2(i-1)$. We now have

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

so p simulates cycle i later than τ. □

Theorem 2.2: The cycle number of a message can be determined using its local time of receipt and information from the initialization phase.

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

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

On the other hand,

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

It follows that $i = \left\lfloor \frac{CLOCK_p^{(\tau)} - \delta_{pq} + 1}{2} \right\rfloor$. □

Hence we adopt the following policy for storing messages from the basic algorithm:

```
upon receipt of a message M from $q$ do
    begin $i := \lfloor \frac{CLOCK_p^{(\tau)} - \delta_{pq} + 1}{2} \rfloor$;
            store M as a cycle-$i$ message
    end
```

This last routine completes the description of our synchronizer. Requirements R1 and R2 are satisfied by theorems 2.1 and 2.2. The cycle time of the synchronizer is 2, and, by theorem 2.2, no extra bits are necessary to determine the cycle number of a message.

It is well possible for a process to receive a cycle-i message before it has simulated cycle i itself. It is possible to receive a cycle-(i+1) message from one neighbor earlier than a cycle-i message from another neighbor. But every message is stored for the correct cycle, and every message is received in time.

3 A Mathematical Model

In the previous section we gave a synchronizer with a cycle time of 2, requiring no extra information to be sent in the messages of the simulated algorithm. As we will see, this is a strong result. In this section we will develop a mathematical model for ABD synchronizers to improve on this result or prove its optimality. We concentrate on requirement R1.

Theorem 3.1: In a synchronizer with cycle time smaller than 2, p can not determine the cycle number of a message from q based on $CLOCK_p$ and δ_{pq}.

Proof: Assume q simulates cycle i at local time T and cycle (i+1) at local time $T+2-\Delta$, for some $\Delta > 0$. On the one hand, it is possible that $w_q = w_p + 1 - \frac{1}{2}\Delta$, q's START message to p has delay 0, and q sends a cycle-i message that suffers a delay of $1-\frac{1}{2}\Delta$. Because $\Delta > 0$ this delay satisfies BD. We have $\delta_{pq} = 1 - \frac{1}{2}\Delta$ and at the moment of receipt of the message by p $CLOCK_p - \delta_{pq} = T + 1 - \frac{1}{2}\Delta$. On the other hand, it is possible that $w_q = w_p$, q's START message suffers a delay of $1-\frac{1}{2}\Delta$, and q sends a cycle-(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 $CLOCK_p - \delta_{pq} = T+1-\frac{1}{2}\Delta$. Thus when $\delta_{pq} = 1-\frac{1}{2}\Delta$ and a message is received at local time $T + \delta_{pq} + 1 - \frac{1}{2}$ it is not possible to determine whether this is a cycle-i or a cycle-(i+1) message, even if p knows T. (We can give p's START message a delay such that δ_{qp} is the same in the two executions above. Thus, it is no use to make T dependent of δ_{qp}.) □

Therefore we feel that if the cycle time of a synchronizer is smaller than 2 it is necessary to send extra information in messages. One bit suffices for this purpose.

Theorem 3.2: If a synchronizer satisfies R1, one bit information per message suffices to satisfy R2 also.

Proof: Suppose p receives a message from q between the simulation of cycle j and $(j+1)$. Now p must determine the cycle number i of this message. By R1 and the fact that p has simulated cycle j already, $i \geq j$. Again by R1, q simulates cycle $(j+2)$ later than p simulates cycle $(j+1)$, 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 **if** $par = par(j)$ **then** $i := j$ **else** $i := j+1$ **fi**. So it suffices to include the parity of cycle numbers in messages of the simulated algorithm. □

Note that p does not need δ_{pq} to determine cycle numbers. Let $G = (V,E)$ be an undirected graph.

Definition 3.3: A *synchronizer function* F for G is a collection of functions $F_p, p \in V$, $F_p : \mathbb{N} \times [0,2)^d \rightarrow \mathbb{R}$, where d is the degree of p in G.

The interpretation of a synchronizer function F is as follows. If p has received START messages of its neighbors $q_1,..., q_d$ at local times $\delta_{pq_1},..., \delta_{pq_d}$ then p simulates cycle i at local time $F_p(i, \delta_{pq_1},..., \delta_{pq_d})$. Henceforth we write $\vec{\delta}_p$ for $\delta_{pq_1},..., \delta_{pq_d}$.

Definition 3.4: 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 3.5: An ABD synchronizer satisfies requirement R1 if and only if its underlying 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: Suppose F satisfies CC. Consider a message, sent from q to p in cycle 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 cycle $i+1$ at global time $w_p + F_p(i+1, \vec{\delta}_p)$. From section 2 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, $w_p + F_p(i+1, \vec{\delta}_p) \geq w_q + F_q(i, \vec{\delta}_q) + 1$, and 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, some i. Construct the following execution of the synchronizer. Process p awakes spontaneously at time w_p, the START message over edge qp arrives at global time $w_p + \delta_{pq}$. By SA, all START messages satisfy the BD axiom and each process executes INIT no later than at the receipt of the first START message. Let q send a message to p in the i^{th} cycle of the simulated algorithm. Because CC does not hold, the message may arrive too late. □

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 say that the function is correct.

Definition 3.6: For a synchronizer function F, the *cycle time* of F is

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

We conclude this section by showing the correctness of the synchronizer of section 2 in this model.

Theorem 3.7: The synchronizer function F, defined by $F_p(i, \vec{\delta}_p) = 2i$, satisfies CC.

Proof: For all S, qp, i we have $F_p(i+1, \vec{\delta}_p) + w_p - F_q(i, \vec{\delta}_q) - w_q - 1 = 2i + 2 + w_p - 2i - w_q - 1 = w_p - w_q + 1 \geq 0$ by SA. □

This function clearly has a cycle time of 2.

4 Fast Synchronization

In this section we show that some networks can be synchronized with a cycle time smaller than 2. First we consider the network K_2 consisting of two processors p and q, and two edges pq and qp.

Theorem 4.1: There exists a correct synchronizer function for K_2 with a cycle 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:

$$F_p(i+1, \vec{\delta}_p) + w_p - F_q(i, \vec{\delta}_q) - w_q - 1$$

$$= 1\frac{1}{2}(i+1) + \frac{1}{2}\delta_{pq} + w_p - 1\frac{1}{2}i - \frac{1}{2}\delta_{qp} - w_q - 1$$

$$\geq 1\frac{1}{2} + \frac{1}{2}(w_q - w_p) + w_p - \frac{1}{2}(w_p - w_q + 1) - w_q - 1 \qquad \text{(SA)}$$

$= 0.$

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

Theorem 4.2: A cycle 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 $F_p(i+1, \frac{1}{2}) \geq F_q(i, \frac{1}{2}) + 1\frac{1}{2}$ 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 $\lim_{i \to \infty} \frac{F_q(i, \frac{1}{2})}{i} \geq \frac{3}{2}$. □

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

Theorem 4.3: There exists a synchronizer function for the $\mathbf{K}_n$ with a cycle time of $2 - \frac{1}{n}$.

Proof: Take $F_p(i, \delta_1,..,\delta_{n-1}) = (2 - \frac{1}{n})i + \frac{1}{n}(\delta_1 + ... + \delta_{n-1})$. We prove CC on edge 21:

$$F_1(i+1, \delta_{12}, \delta_{13},..) + w_1 - F_2(i, \delta_{21}, \delta_{23},..) - w_2 - 1$$

$$= (2 - \frac{1}{n})(i+1) + \frac{1}{n}(\delta_{12} + \delta_{13} + ...) + w_1 - (2 - \frac{1}{n})i - \frac{1}{n}(\delta_{21} + \delta_{23} + ...) - w_2 - 1$$

$$\geq (2 - \frac{1}{n}) + \frac{1}{n}((w_2 - w_1) + (w_3 - w_1) + ...) + w_1 \qquad \text{(SA)}$$

$$- \frac{1}{n}((w_1 - w_2 + 1) + (w_3 - w_2 + 1) + ...) - w_2 - 1$$

$$= 0.$$

The proof for the other edges is similar. □

Theorem 4.4: A cycle time of $(2 - \frac{1}{n})$ is optimal for the $\mathbf{K}_n$.

Proof: (Assume the δ arguments of F_p are listed in the order $\delta_{p,p+1}, \delta_{p,p+2},..$) Let F satisfy CC. For $\rho \in (0, \frac{1}{n})$, let S_ρ be the scenario where

$$w_p = \frac{p}{n} \qquad \text{for } p < n$$

$$w_n = 1 - \rho$$

$$\delta_{1p} = \frac{p-1}{n}$$

$$\delta_{np} = \frac{p}{n}$$

$$\delta_{qp} = \max(0, \frac{p-q}{n}) \quad \text{for } 1 < q < n.$$

This tuple satisfies SA and proves that

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

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

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

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

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

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

The star network S_n has nodes p, q_1, ... q_{n-1} and edges pq_1, ... pq_{n-1} and q_1p, ... $q_{n-1}p$. Note that $K_2 = S_2$. We generalize the above results for S_2 to S_n.

Theorem 4.5: There exists a synchronizer function for S_n with a cycle time of $1\frac{1}{2}$.

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

$$F_p(i+1, \vec{\delta}_p) + w_p - F_{q_j}(i, \delta_{q_jp}) - w_{q_j} - 1$$

$$= 1\frac{1}{2}(i+1) + \frac{1}{2} + w_p - 1\frac{1}{2}i - \delta_{q_jp} - w_{q_j} - 1$$

$$\geq 1\frac{1}{2} + \frac{1}{2} + w_p - (w_p - w_{q_j} + 1) - w_{q_j} - 1 \qquad \text{(SA)}$$

$$= 0$$

and

$$F_{q_j}(i+1, \delta_{q_jp}) + w_{q_j} - F_p(i, \vec{\delta}_p) - w_p - 1$$

$$= 1\frac{1}{2}(i+1)+\delta_{q_jp}+w_{q_j}-1\frac{1}{2}i-\frac{1}{2}-w_p-1$$

$$\geq 1\frac{1}{2}+(w_p-w_{q_j})+w_{q_j}-\frac{1}{2}-w_p-1 \qquad \text{(SA)}$$

$$= 0$$

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

Theorem 4.6: A cycle time of $1\frac{1}{2}$ is optimal for $\mathbf{S}_n$.

Proof: Modify the proof of theorem 4.2 for any of the edges of $\mathbf{S}_n$. □

We have seen that when the cycle time is smaller than 2 an extra bit in messages is necessary. The value of δ_{pq} is not needed for determining the cycle 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 cycle is simulated. Thus, all synchronizers in this section can be implemented in $O(1)$ internal storage.

5 Lower Bound Results

In this section we show that a cycle time of 2 is optimal for rings of size 4 and larger. Theorem 5.5 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 5.1: For a synchronizer function F for G, $A \in Aut(G)$, $F \bullet A$ is the synchronizer function H defined by

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

(Eventually the elements of $\vec{\delta}_p$ are reordered according to A.)

Lemma 5.2: If F satisfies CC, so does $H = F \bullet A$, and $\alpha(H) = \alpha(F)$.

Proof: Fix a scenario S, edge qp, cycle 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 $S' = A \bullet S$ 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 5.3: For synchronizer functions F_1 and F_2, $\sigma_1, \sigma_2 \in I\!R$, $\sigma_1 F_1 + \sigma_2 F_2$ 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 5.4: 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 scenario, edge, i, we have

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

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 in the following sense with addition:

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

It follows that $\alpha(H) \leq \sigma_1 \alpha(F_1) + \sigma_2 \alpha(F_2)$. □

Theorem 5.5: 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 $A \in Aut(G)$, $H = H \bullet A$.

Proof: Let F be given. Take $k = |Aut(G)|$ and define $H = \sum_{B \in Aut(G)} \frac{1}{k}(F \bullet B)$. By lemma 5.2 and 5.4 H is again correct and $\alpha(H) \leq \alpha(F)$. Furthermore, for $A \in Aut(G)$, $H \bullet A = (\sum_{B \in Aut(G)} \frac{1}{k}(F \bullet B)) \bullet A = \sum_{B \in Aut(G)} \frac{1}{k}(F \bullet B \bullet A) = H$ because $Aut(G) \bullet A = Aut(G)$. □

The network $\mathbf{R}_n$ has n nodes $1,..,n$, and $2n$ edges $(p, p+1)$ and $(p, p-1)$, where indices are counted modulo n.

Theorem 5.6: A cycle time of 2 is optimal for $\mathbf{R}_4$.

Proof: Let F be a correct synchronizer function for $\mathbf{R}_4$. By theorem 5.5 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

$$\begin{array}{lll} w_1 = 0, & \delta_{12} = 1, & \delta_{14} = 0, \\ w_2 = 1 - \rho, & \delta_{23} = 1, & \delta_{21} = 0, \end{array}$$

$$w_3 = 1-\frac{1}{2}\rho,\quad \delta_{34} = 0,\qquad \delta_{32} = \frac{1}{2}\rho,$$

$$w_4 = 0,\qquad \delta_{41} = 0,\qquad \delta_{43} = 1.$$

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

Theorem 5.7: A cycle time of 2 is optimal for $\mathbf{R}_n$, $n>4$.

Proof: As the previous theorem. Extend scenario S_ρ as given to a scenario for $\mathbf{R}_n$ with

$$\delta_{1n} = 0,\qquad \delta_{45} = 0,$$

$$w_i = 0,\qquad \delta_{i,i-1} = 0,\qquad \delta_{i,i+1} = 0\qquad \text{for } i>4.\ \square$$

The 2^N cube $\mathbf{C}_N = (V,E)$ where $V = \{0,1\}^N$, and $E = \{(p,q)\in V^2{:}p$ and q differ in one bit$\}$.

Theorem 5.8: A cycle time of 2 is optimal for $\mathbf{C}_N$, $N \ge 2$.

Proof: Modify the proof of theorem 5.6 for any surface of the cube. □

Because $\mathbf{R}_3 = \mathbf{K}_3$ and $\mathbf{C}_1 = \mathbf{R}_2 = \mathbf{K}_2$, we have now determined the optimal cycle times for all rings, stars, complete networks, and cubes. If a synchronizer with a cycle 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 3.2. 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 the synchronizer in section 2. Then no extra bit is necessary, but the internal storage in a process equals its degree in the network.

6 Drifting Clocks

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

$$(1+\varepsilon)^{-1}(t - w_p) \le CLOCK_p^{(t)} \le (1-\varepsilon)^{-1}(t - w_p). \qquad \text{(CA-}\varepsilon\text{)}$$

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}.

In this section we will present an algorithm that resembles the algorithm in section 2. Cycle i is simulated at local time αi for some $\alpha > 2$. It will turn out, as in [Ch87], that after a finite number of cycles a new execution of the initialization phase is necessary. The initialization phase of this algorithm (and the algorithm in the next section) is the same as in section 2. As in section 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,

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

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

Theorem 6.1: All cycle-($i-1$) messages arrive in time for the simulation of cycle i if

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

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

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

Cycle i is simulated by p when $CLOCK_p = \alpha i$. The message is clearly in time if

$$(1-\varepsilon)^{-1}((1+\varepsilon)\alpha(i-1) + 2) \leq \alpha i,$$

and this is equivalent to I1. □

Theorem 6.2: The cycle number of a message can be determined using its local time of receipt and information from the initialization phase if

$$i \leq \frac{(1+\varepsilon)^2\alpha - 2(1+3\varepsilon)}{4\varepsilon\alpha}. \qquad \text{(I2)}$$

Proof: Let σ, τ be as above, then

$$\begin{aligned} CLOCK_p^{(\tau)} - \delta_{pq} &\leq (1-\varepsilon)^{-1}(\tau - w_p) - (1+\varepsilon)^{-1}(w_q - w_p) && \text{(CA-}\varepsilon\text{, IP-}\varepsilon\text{)}\\ &< (1-\varepsilon)^{-1}(\sigma + 1 - w_p) - (1+\varepsilon)^{-1}(w_q - w_p) && \text{(BD)}\\ &\leq (1-\varepsilon)^{-1}((1-\varepsilon)\alpha(i-1) + w_q + 1 - w_p) - (1+\varepsilon)^{-1}(w_q - w_p) && \text{(CA-}\varepsilon\text{)} \end{aligned}$$

$$= \frac{1+\varepsilon}{1-\varepsilon}\alpha(i-1)+\frac{2\varepsilon}{1-\varepsilon^2}(w_q-w_p)+\frac{1}{1-\varepsilon}$$

$$\leq \frac{1+\varepsilon}{1-\varepsilon}\alpha(i-1)+\frac{2\varepsilon}{1-\varepsilon^2}+\frac{1}{1-\varepsilon} \qquad \text{(IP-}\varepsilon\text{)}$$

$$= \frac{1+\varepsilon}{1-\varepsilon}\alpha(i-1)+\frac{1+3\varepsilon}{1-\varepsilon^2}.$$

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

$$CLOCK_p^{(\tau)} - \delta_{pq} > (1+\varepsilon)^{-1}(\tau - w_p) - (1-\varepsilon)^{-1}(w_q - w_p + 1) \qquad \text{(CA-}\varepsilon\text{, IP-}\varepsilon\text{)}$$

$$\geq (1+\varepsilon)^{-1}(\sigma - w_p) - (1-\varepsilon)^{-1}(w_q - w_p + 1) \qquad \text{(BD)}$$

$$\geq (1+\varepsilon)^{-1}((1-\varepsilon)\alpha i + w_q - w_p) - (1-\varepsilon)^{-1}(w_q - w_p + 1) \qquad \text{(CA-}\varepsilon\text{)}$$

$$= \frac{1-\varepsilon}{1+\varepsilon}\alpha i - \frac{2\varepsilon}{1-\varepsilon^2}(w_q-w_p) - \frac{1}{1-\varepsilon}$$

$$> \frac{1-\varepsilon}{1+\varepsilon}\alpha i - \frac{2\varepsilon}{1-\varepsilon^2} - \frac{1}{1-\varepsilon} \qquad \text{(IP-}\varepsilon\text{)}$$

$$= \frac{1-\varepsilon}{1+\varepsilon}\alpha i - \frac{1+3\varepsilon}{1-\varepsilon^2}.$$

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

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

and this is equivalent to I2. □

The reader may verify that $\frac{(1+\varepsilon)^2\alpha - 2(1+3\varepsilon)}{4\varepsilon\alpha} \leq \frac{(1+\varepsilon)\alpha - 2}{2\varepsilon\alpha}$ (use $\varepsilon \leq 1$ and $\alpha \geq 2$), hence I2 implies I1. With a fixed α, we can either simulate the number of cycles given by I1, and use an extra bit for recognizing messages, or simulate a (smaller) number of cycles given by I2 and use no extra bit. After this number of cycles the initialization phase must be executed again to simulate more cycles. 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 [Ch87], we include an example computation. Assume the timers may miss a tenth of a second a day, which makes $\varepsilon = \frac{1}{864000}$, and set $\alpha = 7$. Using I1 we find that 308571 cycles can be simulated before reinitialization is necessary if an extra bit is used in messages. Using I2 we find that 154286 cycles can be simulated before reinitialization is necessary if no extra bit is used. The algorithm of [Ch87] simulates 142045 cycles when $\alpha = 8$, using no extra bit.

7 Faster Simulation

In this section we develop a faster algorithm to synchronize ABD Networks with drifting timers. No reinitialization will be necessary at all during simulation. The initialization phase is again the same as in section 2. We postulate that cycle i is simulated at local time $f(i)$, and do not assume that f is a linear function.

Theorem 7.1: All cycle-$(i-1)$ messages will arrive in time if

$$f(i) \geq a_1 f(i-1) + b_1, \tag{J1}$$

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

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

$$\begin{aligned} CLOCK_p^{(\tau)} &\leq (1-\varepsilon)^{-1}(\tau - w_p) && \text{(CA-}\varepsilon\text{)} \\ &< (1-\varepsilon)^{-1}(\sigma + 1 - w_p) && \text{(BD)} \\ &\leq (1-\varepsilon)^{-1}((1+\varepsilon) f(i-1) + w_q + 1 - w_p) && \text{(CA-}\varepsilon\text{)} \\ &< (1-\varepsilon)^{-1}((1+\varepsilon) f(i-1) + 2). && \text{(IP-}\varepsilon\text{)} \\ &= a_1 f(i-1) + b_1, \end{aligned}$$

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

Theorem 7.2: The cycle 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 $a_2 = \left(\frac{1+\varepsilon}{1-\varepsilon}\right)^2$ and $b_2 = \frac{2+6\varepsilon}{(1-\varepsilon)^2}$.

Proof: Let σ, τ be as above, then

$$\begin{aligned} CLOCK_p^{(\tau)} - \delta_{pq} &\leq (1-\varepsilon)^{-1}(\tau - w_p) - (1+\varepsilon)^{-1}(w_q - w_p) && \text{(CA-}\varepsilon\text{, IP-}\varepsilon\text{)} \\ &< (1-\varepsilon)^{-1}(\sigma + 1 - w_p) - (1+\varepsilon)^{-1}(w_q - w_p) && \text{(BD)} \\ &\leq (1-\varepsilon)^{-1}((1-\varepsilon) f(i-1) + w_q + 1 - w_p) - (1+\varepsilon)^{-1}(w_q - w_p) && \text{(CA-}\varepsilon\text{)} \end{aligned}$$

$$= \frac{1+\varepsilon}{1-\varepsilon} f(i-1) + \frac{2\varepsilon}{1-\varepsilon^2}(w_q - w_p) + \frac{1}{1-\varepsilon}$$

$$\le \frac{1+\varepsilon}{1-\varepsilon} f(i-1) + \frac{2\varepsilon}{1-\varepsilon^2} + \frac{1}{1-\varepsilon} \qquad \text{(IP-}\varepsilon\text{)}$$

$$= \frac{1+\varepsilon}{1-\varepsilon} f(i-1) + \frac{1+3\varepsilon}{1-\varepsilon^2}.$$

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

$$CLOCK_p^{(\tau)} - \delta_{pq} > (1+\varepsilon)^{-1}(\tau - w_p) - (1-\varepsilon)^{-1}(w_q - w_p + 1) \qquad \text{(CA-}\varepsilon\text{, IP-}\varepsilon\text{)}$$

$$\le (1+\varepsilon)^{-1}(\sigma - w_p) - (1-\varepsilon)^{-1}(w_q - w_p + 1) \qquad \text{(BD)}$$

$$\ge (1+\varepsilon)^{-1}((1+\varepsilon) f(i) + w_q - w_p) - (1-\varepsilon)^{-1}(w_q - w_p + 1) \qquad \text{(CA-}\varepsilon\text{)}$$

$$= \frac{1-\varepsilon}{1+\varepsilon} f(i) - \frac{2\varepsilon}{1-\varepsilon^2}(w_q - w_p) - \frac{1}{1-\varepsilon}$$

$$> \frac{1-\varepsilon}{1+\varepsilon} f(i) - \frac{2\varepsilon}{1-\varepsilon^2} - \frac{1}{1-\varepsilon} \qquad \text{(IP-}\varepsilon\text{)}$$

$$= \frac{1-\varepsilon}{1+\varepsilon} f(i) - \frac{1+3\varepsilon}{1-\varepsilon^2}.$$

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

$$\frac{1-\varepsilon}{1+\varepsilon} f(i) - \frac{1+3\varepsilon}{1-\varepsilon^2} \ge \frac{1+\varepsilon}{1-\varepsilon} f(i-1) + \frac{1+3\varepsilon}{1-\varepsilon^2}$$

or $f(i) \ge a_2 f(i-1) + b_2$. □

Because $a_2 \ge a_1$ and $b_2 \ge b_1$, again J2 implies J1. We note that the function $f(i) = b\,\frac{a^i - 1}{a - 1}$ satisfies $f(i) = a\,f(i-1) + b$. 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 message, 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 cycle time. Yet, for all values for which they can be compared with the functions in section 6, 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 cycle. This cycle is simulated at time $7 \times 308571 = 2159997$. The synchronizer in this section simulates this cycle at local time $f_1(308571) = 900915$. With no extra bit, reinitialization was necessary after the 154286th cycle. This cycle is simulated at time $7 \times 154286 = 1080002$. The synchronizer in this section simulates this cycle at local time $f_2(154286) = 450461$.

8 Unidirectional Networks

In this section we drop the assumption that the network is bidirectional. That is, the existance of an edge qp no longer implies the existance of an edge pq. For simplicity we assume that clocks run exact, i.e., we adopt the clock axiom CA as in section 2. The generalization of the results in this section can be done as in the previous two sections.

Again, the initialization phase is as in section 2. After this initialization we have (for every edge qp) $w_p < w_q + 1$ and we find

$$0 \le \delta_{pq}, \quad w_q - w_p \le \delta_{pq} < w_q - w_p + 1 \tag{IP}$$

as in section 2. We do not necessarily have $w_q < w_p + 1$, but instead we have $w_q < w_p + d(p,q)$. Here $d(p,q)$ denotes the distance from p to q. Define d_m as $\max\limits_{qp \in E} d(p,q)$ and assume d_m to be known by all processes. Further, a process simulates cycle i at local time $(d_m + 1)i$. It is easily seen that all messages now arrive in time. Because $d_m \ge 1$, the cycle time of this synchronizer is at least 2, so there is no need for an extra bit in messages. As in theorem 2.2 it can be shown that

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

if τ is the time of receipt of a cycle-i message from q.

9 Conclusions

Our results improve in several ways on previous ABD synchronizers. We presented several synchronizers. For the ideal case, where clocks do not drift, we presented a synchronizer with a cycle time of 2, which does not require an extra bit in messages. Internal storage needed in a node to implement this synchronizer equals the degree of the node on the network. As an alternative, the synchronizer can be implemented with $O(1)$ storage per process, but then an extra bit in messages is necessary. This is the first algorithm of [Ch87]. We presented several synchronizers with a cycle time smaller than 2, for networks with special topology (complete network and star). These synchronizers require an extra bit in messages, and can be implemented in $O(1)$ storage per process. For the case where timers do drift we presented two synchronizers. Both synchronizers work faster if an extra bit is included in messages. This faster mode, with an extra bit in messages, can again be implemented in $O(1)$ storage per process. Our second synchronizer for this case does not require that after a finite number of cycles the network is reinitialized.

In the ideal case (no clock drift) we proved the optimality of a cycle time of 2 for some networks. A topic for further research is to find optimal synchronizers for other networks than the topologies studied here. Another topic for further research is to study the clock drift case in more detail. In section 7 we modeled the synchronizer as a function $F: \mathbb{N} \rightarrow \mathbb{R}$. Here also $\vec{\delta}_p$ could be taken into account and a different function could be used in every process. We did not do this here because the gain of this approach in the ideal case (sections 3 to 5) was limited to special topologies only. We expect this to be the case when clocks drift also.

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 $|CLOCK_p^{(t)} - CLOCK_q^{(t)}| < \Delta$. It is easy to see that a process can now simulate cycle i at local time $(1+\Delta)i$, and R1 is satisfied. In [LL84] it is shown that clocks in the $\mathbf{K}_n$ can not be synchronized tighter than within $1 - \frac{1}{n}$. This corresponds with theorem 4.4.

10 References

[Aw85] Awerbuch, B., *Complexity of Network Synchronization*, Journal of the ACM 32 (1985) 804-823.

[Ch87] Chou, C.-T., I. Cidon, I.S. Gopal, S. Zaks, *Synchronizing Asynchronous Bounded Delay Networks*, in: J. van Leeuwen (ed.), Distributed Algorithms: 2nd International Workshop, Lecture Notes in Computer Science 312, Springer Verlag, 1988, pp. 212-218.

[LL84] Lundelius, J., N. Lynch, *An Upper and Lower Bound for Clock Synchronization*, Information and Control 62 (1984) 190-204.

[LT87] Lakshmanan, K.B., K. Thulasiraman, *On the Use of Synchronizers for Asynchronous Communication Networks*, in: J. van Leeuwen (ed.), Distributed Algorithms: 2nd International Workshop, Lecture Notes in Computer Science 312, Springer Verlag, 1988, pp. 257-277.

[Vi85] Vitanyi, P.M.B., *Time-Driven Algorithms for Distributed Control*, Tech. Rep. CS-R8510, CWI, Amsterdam, The Netherlands, 1985.