Clock Synchronization and the Power of Broadcasting*

Joseph Y. Halpern
IBM Almaden Research Center, Dept. K53/802
San Jose, CA 95120
halpern@almaden.ibm.com

Ichiro Suzuki†
Department of Electrical Engineering and Computer Science
University of Wisconsin – Milwaukee
P.O. Box 784, Milwaukee, WI 53201
suzuki@cvax.cs.uwm.edu

October 2, 1996

Abstract: We investigate the power of a broadcast mechanism in a distributed network. We do so by considering the problem of synchronizing clocks in an error-free network, under the assumption that there is no upper bound on message transmission time, but that broadcast messages are guaranteed to be received within an interval of size ϵ, for some fixed constant ϵ. This is intended to be an idealization of what happens in multiple access networks, such as the Ethernet. We then consider tradeoffs between the type and number of broadcasts, and the tightness of synchronization. Our results include (1) matching upper and lower bounds of $(1+\frac{1}{K})\epsilon$ on the precision of clock synchronization attainable for $n \geq 3$ processes using K $(n-1)$-casts, $3 \leq K \leq n$, (2) matching upper and lower bounds of $(1+\frac{1}{n})\epsilon$ on the precision of clock synchronization attainable for $n \geq 3$ processes using an arbitrary number of $(n-1)$-casts, and (3) matching upper and lower bounds of $(1+\frac{n-2}{n})\epsilon$ on the precision attainable using 2-casting.

*This paper is essentially identical to one that appears in *Distributed Computing* **5**:2, 1991, pp. 73–83.
†This author was supported in part by the National Science Foundation under grant CCR-9004346.

1 Introduction

Broadcasting is a basic primitive that is available in many computer networks (such as the Ethernet [14]). In some cases, the network provides the user with the ability to send an ℓ-cast, that is, a broadcast message to some subset of size ℓ. In this paper, we investigate the power of broadcasting—both by comparing broadcasting to point-to-point message transmission and by comparing ℓ-casting to k-casting, for $\ell \neq k$—in the context of the problem of synchronizing clocks in a fully-connected, error-free network. (We could, of course, identify a point-to-point message with a broadcast to one process, but we find it useful to distinguish broadcasts from point-to-point messages.)

Even in an Ethernet-style network, where all messages can be viewed as broadcast messages, an ℓ-cast can be viewed as being cheaper than an k-cast for $\ell < k$, since if a process is not among the recipients of a broadcast, it must do significantly less processing of the message. In particular, in large networks, we can view a broadcast to a large subset of processes as being considerably more expensive than a point-to-point message. As Gray says [7], "Practitioners want architectures which scale to arbitrary size networks. As a consequence they limit attention to *multicast*: broadcast to a "small" group of processes [3]. Multicast scales to large networks ... [whereas broadcast does not, because of its cost]." Given these observations, it becomes important to understand the tradeoffs between the use of broadcast messages and the ability to achieve certain goals in distributed systems.

We have taken clock synchronization as a paradigm problem to study in this context because it has been so well studied, both in the presence of faults (see, for example, [1, 5, 9, 11, 13, 15]) and without faults [8, 10, 12, 16]. The basic problem we consider is the same as that considered in [8, 12]: we assume that each process has a clock that runs at the rate of real time, and the problem is to synchronize these clocks as tightly as possible. Of course, in practice, clocks drift apart. However, the drift is typically small, and by ignoring it here, we can examine the impact of broadcasting more carefully. In all the papers cited above with the exception of [1, 16], it is assumed that processes can communicate only by sending point-to-point messages. Moreover, in all of these papers except [16], it is assumed that there is a (commonly) known upper bound on the message transmission time. Here we drop the assumption that there is an upper bound on the message transmission time. Without further assumptions, it is easy to show that clock synchronization is impossible; i.e., there is no upper bound on how tightly we can synchronize clocks. (We provide a formal proof of this fact in the next section.) So, following [16], we assume that the network has a broadcast primitive, in addition to a facility providing point-to-point message transmission. Although there is no upper bound on how long it will take for a broadcast message to arrive, we do assume that all processes receive a broadcast message within a (real time) window of ϵ, for some (presumably small) constant ϵ. This is meant to be an idealized version of what actually happens in an Ethernet (when probabilistic considerations are ignored). Processors can try to broadcast, but if there is a conflict (when two or more processes try to broadcast at the same time), there must be a retry. There is no upper bound on the number of retry attempts. Although with high probability a broadcast will succeed after very few attempts, we cannot place an upper bound on message transmission time. On the other hand, if a broadcast does succeed, then all the intended recipients receive it within a small window of time.

In [16], the problem of clock synchronization in this model was considered under the extra assumption that a process can broadcast to everyone but itself. An upper bound of $(1+\frac{1}{n})\epsilon$ on the precision of synchronization was proved, as well as an almost matching lower bound of $(1+\frac{1}{n(n+2)})\epsilon$. Here we extend and generalize the results of [16], investigating in more detail the tradeoffs between the type and number of broadcasts allowed and the precision of synchronization attainable. Among other results, we show that if precisely K $(n-1)$-casts are allowed, then we can attain a synchronization of $(1+\frac{1}{K})\epsilon$ for $3 \leq K \leq n$; if $K=2$, then we can only attain a synchronization of 2ϵ. We prove matching lower bounds in all these cases. Moreover, we show that even with an arbitrary number of $(n-1)$-casts, we can only synchronize to within $(1+\frac{1}{n})\epsilon$; that is, we can get optimal synchronization with n $(n-1)$-casts. This result is explained by our observation that what really matters is not how many $(n-1)$-casts are used, but the number of distinct subsets of processes that are recipients of $(n-1)$-casts. Our lower bound of $(1+\frac{1}{K})\epsilon$ holds no matter how many $(n-1)$-casts are used (and no matter how much point-to-point communication is used) as long as there are no more than K distinct subsets of processes that receive broadcast messages. Since there are n distinct subsets of size $(n-1)$, our lower bound of $(1+\frac{1}{n})\epsilon$ holds with arbitrary $(n-1)$-casts. This proves that the upper bound of [16] is in fact optimal.

We then turn our attention to ℓ-casting. In the case of n-casting, where a broadcast reaches all processes in the system including the sender, we can prove a tight bound of ϵ on the precision of synchronization attainable. For $2 \leq \ell \leq n-1$, we provide an algorithm using ℓ-casting that synchronizes to within $(1+\frac{n-\ell}{n})\epsilon$. We conjecture that this is optimal, although we have only been able to prove this for the case $\ell = 2$ and $\ell = n-1$. For $2 < \ell < n-1$, we do have a nontrivial lower bound of $\max(\frac{n-1}{n}\frac{\ell}{\ell-1}\epsilon, (1+\frac{1}{\ell+1})\epsilon)$.

The rest of this paper is organized as follows. In the next section we give the necessary definitions and show that broadcasts are necessary to achieve synchronization in networks where there is no upper bound on message transmission time. We consider $(n-1)$-casts in Section 3, proving tight bounds on the synchronization achievable with K $(n-1)$-casts. In Section 4 we consider ℓ-casting. We conclude with some open questions and discussion in Section 5.

2 Preliminaries

As we mentioned above, the framework we use is essentially that of [8, 12]. Consider a network with $n \geq 2$ processes, say $P_1, P_2, \ldots, P_n$. Process P_i has a *physical clock* C_i which is a real-valued function of real time. We assume that the physical clocks run at the rate of real time and they cannot be reset by the processes; that is, $C_i(t) = C_i(0) + t$ at all times t. The processes have no access to the real time. Process P_i has a local variable A_i (for *adjustment*) which provides the difference between the logical and physical clocks of P_i. That is, the *logical clock* $L_i(t)$ of process P_i at time t is given by $L_i(t) = C_i(t) + A_i(t)$, where $A_i(t)$ is the value of A_i at t.

We assume that at some fixed real time (say, in response to some external signal) all the processes wish to synchronize their clocks as tightly as possible. They do so by sending messages, both point-to-point messages from one process to another process and broadcasts from some process to a subset consisting of at least 2 processes. There is no upper bound on the time it takes a message to arrive from the time that it is sent; however, broadcasts arrive to all their intended recipients within a window of size ϵ; i.e., if a broadcast is sent at time t, there is some $t' \geq t$ such that all recipients receive the broadcast within the interval $[t', t'+\epsilon]$. We shall be interested in the size of the subset of recipients; an *ℓ-cast* is a broadcast from a process to a set of ℓ processes.

Following [8], we assume that a clock synchronization algorithm is a deterministic algorithm in which the state transition and the action of sending messages of process P_i at time t is determined only by the value of $C_i(t)$ and the history of P_i at t. Here, the *history* of P_i at t is the sequence consisting of tuples of the forms $\langle P_j, m, T, y\rangle$ for every message P_i has sent or received before t, where $\langle P_j, m, T, y\rangle$ represents that message m was either sent ($y = \mathbf{sent}$) or received ($y = \mathbf{received}$) to or from P_j when the value of C_i was T.[1]

An algorithm is said to *synchronize* the logical clocks to within γ if the algorithm eventually terminates, and when it terminates at time t, $|L_i(t) - L_j(t)| \leq \gamma$ holds for all $i \neq j$. Although we do not require that the algorithm enforce monotonicity (i.e., that $L_i(t') \geq L_i(t)$ if $t' \geq t$), since this makes our lower bounds stronger, in fact, all our algorithms do enforce monotonicity.

An execution of an algorithm can be viewed as a function s such that for all $1 \leq i \leq n$ and times t, $s(i,t)$ is the history of P_i at t during the execution. This observation leads us to the following definitions. A function s such that $s(i,t)$ is a pair consisting of a history of P_i and a clock reading T for all $1 \leq i \leq n$ and times t is called a *scenario.* A scenario s is *legal* with respect to a given algorithm if there exists an execution of the algorithm such that for all $1 \leq i \leq n$ and times t, the history in $s(i,t)$ is the history of P_i at time t and the clock reading T in $s(i,t)$ is $C_i(t)$.

Let s_1 be a scenario of a given algorithm. We say that scenario s_2 is obtained from s_1 by *shifting* P_i by ρ_i if $s_2(i,t) = s_1(i, t-\rho_i)$ for $1 \leq i \leq n$. Intuitively, state transitions and sending and receiving of messages of P_i occur uniformly ρ_i later in s_2 than in s_1. Using $C_{i,s}(t)$ (resp. $L_{i,s}(t)$) to denote the physical (resp. logical) clock reading of process P_i in scenario s, note that we have $C_{i,s_2}(t) = C_{i,s_1}(t) - \rho_i$. Since the clock adjustment made by a process in an execution of the algorithm depends only on the reading of its physical clock and its message history, it is easy to see that the following lemma holds.

Lemma 1 *Let s_1 and s_2 be scenarios which are legal with respect to a given algorithm, where s_2 is obtained from s_1 by shifting P_i by ρ_i for $1 \leq i \leq n$. Let t be any time after the execution of the algorithm terminates at every process in both s_1 and s_2. Then $L_{i,s_2}(t) = L_{i,s_1}(t) - \rho_i$.*

Proof Scenarios s_1 and s_2 are indistinguishable in the sense that each process has identical histories in s_1 and s_2 when its physical clock has the same value. Thus for each $1 \leq i \leq n$, the values of A_i computed by P_i are the same in both scenarios. Since $C_{i,s_2}(t) = C_{i,s_1}(t-\rho_i)$, it follows that $L_{i,s_2}(t) = L_{i,s_1}(t-\rho_i) = L_{i,s_1}(t) - \rho_i$. (Since P_i has terminated in s_2 by time t, it must have also terminated in s_1 by time $t - \rho_i$, and hence it does not adjust its logical clock in s_1 in the interval $[t-\rho_i, t]$.) $\square$

We conclude this section by proving, as claimed in the introduction, that we really need to use broadcast messages in order to synchronize clocks at all in our framework, given that we have no upper bound on message transmission time. In fact, we show something even

[1] We assume that processes are up throughout the running of the algorithm. We could assume instead that processes either wake up in response to some external signal or as a result of receiving a message. We have ignored the issue of processes waking up for ease of exposition; all our results hold with essentially no change in the more complicated model.

stronger, namely, that every process must receive a broadcast message in order to guarantee synchronization.

Theorem 1 *The logical clocks of $n \geq 2$ processes cannot be synchronized by an algorithm $\mathcal{A}$ such that in each execution of $\mathcal{A}$, some process does not receive a broadcast message.*

Proof Suppose that there exists an algorithm $\mathcal{A}$ which synchronizes the logical clocks to within γ, and in each execution of $\mathcal{A}$, some process does not receive a broadcast message. We derive a contradiction, using the by-now standard "many scenarios" technique [5, 6, 8, 12].

Choose $D > \gamma$. Let s_1 be a scenario that corresponds to an execution of $\mathcal{A}$ in which the transmission time of every message (both broadcast and point-to-point) is $2D$. Suppose without loss of generality that P_1 does not receive any broadcast messages in s_1. Let s_2 be the scenario obtained from s_1 by shifting P_1 by $2D$ and all other processes by zero. Scenario s_2 is legal, since (1) all messages are received in nonnegative time, and (2) broadcast messages are received simultaneously. Specifically, the message transmission time in s_2 is zero from P_1 to $P_2, \ldots, P_n$, $4D$ from $P_2, \ldots, P_n$ to P_1, and $2D$ for all other messages. Figure 1 shows a broadcast from P_1 to P_2, P_3, P_4 and some point-to-point messages in scenarios s_1 and s_2, where $n = 4$. The execution of each process is represented by a horizontal line. Vertical lines are drawn at intervals of D. Message transmissions are indicated by an arrow.

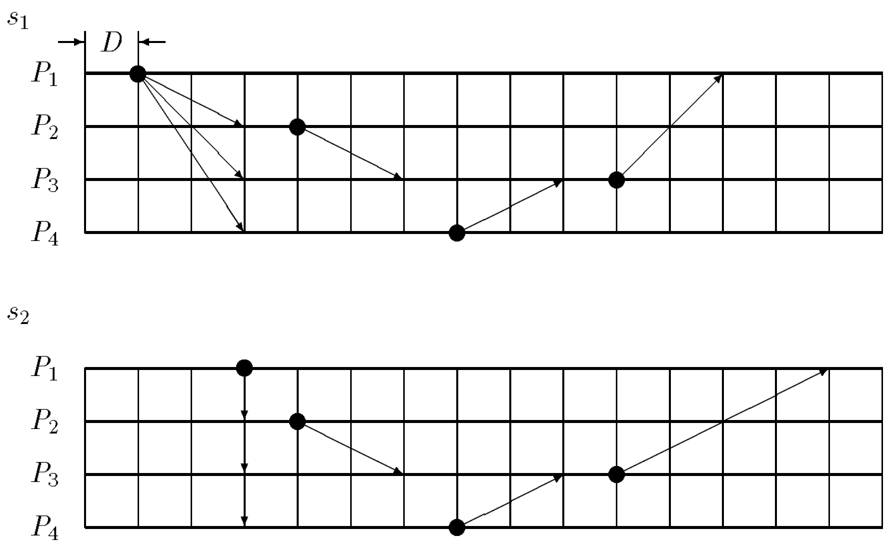

Figure 1: Scenarios s_1 and s_2 in the proof of Theorem 1.

Let t be a real time such that the algorithm terminates at every process in both s_1 and s_2 by time t. Let $L_{1,s_1}(t) = T_1$ and $L_{2,s_1}(t) = T_2$. By Lemma 1, $L_{1,s_2}(t) = T_1 - 2D$ and $L_{2,s_2}(t) = T_2$. By the assumption on γ, we have

$$T_1 \leq T_2 + \gamma$$

and

$$T_2 \leq T_1 - 2D + \gamma.$$

By adding the two inequalities we obtain

$$\gamma \geq D.$$

This is a contradiction. □

3 Optimal clock synchronization using $(n-1)$-casting

In this section we focus on optimal clock synchronization using $(n-1)$-casting. We assume $n \geq 3$ (since if $n = 2$, then $(n-1)$-casting amounts to point-to-point communication, so that clock synchronization is impossible by Theorem 1). Informally, this is meant to correspond to the case where a process can broadcast to all processes other than itself (although, in fact, our arguments hold even in the case where a process can include itself among the recipients of its broadcast, as long as the list of recipients has no more than $n-1$ processes). Since we assume a broadcast is expensive, we focus here on the tradeoff between the number of broadcasts and the precision of synchronization.

We know from Theorem 1 that we require at least two $(n-1)$-casts in order to achieve synchronization. If we are allowed no more than two $(n-1)$-casts, then we can synchronize processes to within 2ϵ:

Theorem 2 *The logical clocks of $n \geq 3$ processes can be synchronized to within 2ϵ using two $(n-1)$-casts and one point-to-point message.*

Proof Since the algorithm is simple, we only describe it informally. We let P_1 and P_2 broadcast to $P_2, P_3, \ldots, P_n$ and $P_1, P_3, P_4, \ldots, P_n$, respectively. (The broadcast of P_2 can actually be a 2-cast to P_1 and P_3.) Processes $P_2, P_3, \ldots, P_n$ set their logical clocks to zero upon receipt of the broadcast of P_1 (that is, P_i sets $A_i = -C_i$). Then the logical clocks of $P_2, \ldots, P_n$ are synchronized to within ϵ. P_3 then sends a message to P_1 carrying the difference D between the times it received P_1's broadcast and P_2's broadcast. If P_1 received P_2's broadcast and P_3's message at physical times T and T', respectively, then P_1 sets its logical clock to $T' - T + D$ when it receives P_3's message (that is, P_1 sets $A_1 = D - T$). It is easy to show that the logical clocks of P_1 and P_2 are now synchronized to within ϵ, and hence the logical clocks of all processes are synchronized to within 2ϵ. □

It turns out that the one point-to-point message used in our algorithm is necessary. We return to this issue at the end of the section.

The upper bound of Theorem 2 is tight. The logical clocks of $n \geq 3$ processes cannot be synchronized any more closely than 2ϵ by using two $(n-1)$-casts, no matter how much additional point-to-point communication we use. The lower bound is an easy corollary to the following result, which shows that the lower bound holds even if we allow an arbitrary number of $(n-1)$-casts, as long as they cannot be sent to more than two distinct subsets of processes of size $n-1$.

Theorem 3 *The logical clocks of $n \geq 3$ processes cannot be synchronized any more closely than 2ϵ by an algorithm $\mathcal{A}$ which uses $(n-1)$-casting and point-to-point communication, if, in each execution of $\mathcal{A}$, $(n-1)$-casts cannot be sent to more than two distinct subsets of processes of size $n-1$.*

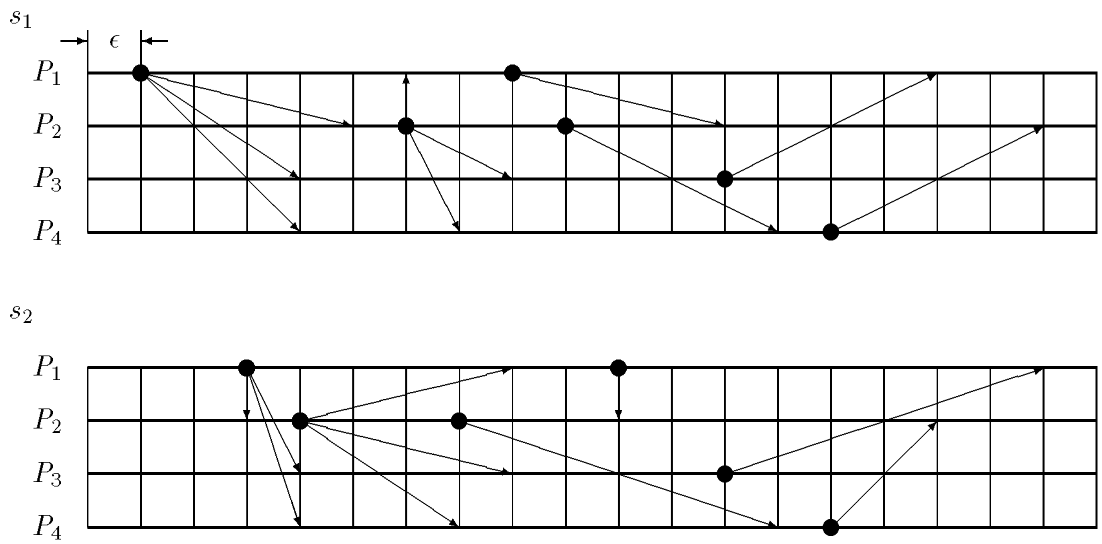

Figure 2: Scenarios s_1 and s_2 in the proof of Theorem 3.

Proof Fix an algorithm $\mathcal{A}$ which synchronizes the logical clocks to within γ such that in each execution of $\mathcal{A}$, no more than two distinct subsets of size $n-1$ are recipients of $(n-1)$-casts. Without loss of generality let s_1 be a scenario of $\mathcal{A}$ in which

1. each of $P_3, \ldots, P_n$ receives all $(n-1)$-casts,

2. the transmission time of every point-to-point message is 4ϵ,

3. the transmission time of the message to P_i in every $(n-1)$-cast to $P_2, \ldots, P_n$ is 4ϵ if $i = 2$ and 3ϵ otherwise, and

4. the transmission time of the message to P_i in every $(n-1)$-cast to $P_1, P_3, \ldots, P_n$ is ϵ if $i = 1$ and 2ϵ otherwise.

Clearly s_1 is legal. Let s_2 be the scenario obtained from s_1 by shifting P_1 by 2ϵ, P_2 by -2ϵ, and all other processes by zero. Scenario s_2 is legal, since (1) all messages are received in nonnegative time and (2) broadcast messages are received within an interval of size ϵ. See Figure 2.

Let t be any time after the algorithm terminates at every process in both s_1 and s_2. Let $L_{1,s_1}(t) = T_1$ and $L_{2,s_1}(t) = T_2$. By Lemma 1, $L_{1,s_2}(t) = T_1 - 2\epsilon$ and $L_{2,s_2}(t) = T_2 + 2\epsilon$. Then by the assumption on γ we have

$$T_1 \leq T_2 + \gamma$$

and

$$T_2 + 2\epsilon \leq T_1 - 2\epsilon + \gamma.$$

By adding the two inequalities we obtain

$$\gamma \geq 2\epsilon.$$

□

Corollary 1 *The logical clocks of $n \geq 3$ processes cannot be synchronized any more closely than 2ϵ by using only two $(n-1)$-casts and additional point-to-point communication.*

We now turn our attention to the case where we allow K $(n-1)$-casts, for $3 \leq K \leq n$. Again, we can prove tight bounds in this case. We start with the upper bound. Our algorithm for the upper bound is a generalization of that given in [16] for the case $K = n$.

Theorem 4 *The logical clocks of $n \geq 3$ processes can be synchronized to within $(1 + \frac{1}{K})\epsilon$ by using K $(n-1)$-casts for $3 \leq K \leq n$.*

Proof We describe an algorithm which uses an averaging process much in the same spirit of the algorithm in [12]. The concept of "view" introduced below is essential in describing the algorithm.

Let S be an $(n-1)$-element subset of $\{1, 2, \ldots, n\}$. Suppose that a process $(n-1)$-casts to the processes P_i such that $i \in S$, and let v_i be the value of physical clock C_i at the moment the broadcast is received by P_i. Then the function V from S to real numbers such that $V(i) = v_i$ for $i \in S$ is called a *view*. The set S is the domain of the view. Since, by assumption, messages broadcast by a process are received within an interval of ϵ, the following lemma is immediate.

Lemma 2 *Let V be a view with domain S. There exist some t and a function $\alpha : S \rightarrow [0, \epsilon]$ such that $V(i) = C_i(t) + \alpha(i)$ for each $i \in S$.*

The algorithm consists of three phases. It uses K $(n-1)$-casts in Phase 1 and $n-1$ point-to-point messages in each of Phases 2 and 3.

Phase 1 For $1 \leq h \leq K$, let $S_h = \{1, 2, \ldots, n\} - \{h\}$. For each $1 \leq h \leq K$, choose an arbitrary process and let it $(n-1)$-cast to the processes P_i such that $i \in S_h$.

Phase 2 Each process sends one fixed process, say P_1, a message containing the times (on its physical clock) that each of the broadcast messages arrived. From this information P_1 computes views $V_1, V_2, \ldots, V_K$ with domain $S_1, S_2, \ldots, S_K$, respectively, and then computes $A_1, A_2, \ldots, A_n$ as follows. For $1 \leq i \leq K$,

$$A_i = \frac{1}{K} \sum_{1 \leq k \leq K,\, k \neq i} D_{k,i}$$

where for $1 \leq k \leq K$ such that $k \neq i$,

$$D_{k,i} = \frac{1}{K-2} \sum_{1 \leq h \leq K,\, k,i \in S_h} (V_h(k) - V_h(i)).$$

For $K + 1 \leq i \leq n$,

$$A_i = \frac{1}{K} \sum_{1 \leq k \leq K} D_{k,i}$$

where for $1 \leq k \leq K$,

$$D_{k,i} = \frac{1}{K-1} \sum_{1 \leq h \leq K,\, k \in S_h} (V_h(k) - V_h(i)).$$

Phase 3 For each $2 \leq i \leq n$, P_1 sends P_i a message containing A_i. (Observe that we can replace these $(n-1)$ point-to-point messages by one $(n-1)$-cast.)

For $k \neq i$, $D_{k,i}$ is the average of the differences between the physical clock readings of P_k and P_i observed in views V_h such that $k, i \in S_h$. The number of such views is $K-2$ if $1 \leq i \leq K$, and $K-1$ if $K+1 \leq i \leq n$. Conceptually, A_i is the average of $D_{k,i}$ over all $1 \leq k \leq K$, including $D_{i,i} = 0$ if $1 \leq i \leq K$, which is viewed as the difference between the physical clock readings of P_i and P_i.

By Lemma 2, for each view V_h there exist t_h and a function $\alpha_h : S_h \to [0, \epsilon]$ such that $V_h(i) = C_i(t_h) + \alpha_h(i)$ for each $i \in S_h$. We now prove that the algorithm does indeed synchronize to within the required precision, using a sequence of lemmas. In the lemmas that follow, we fix an execution of the algorithm and take t to be a time after the algorithm has terminated in that execution.

Lemma 3 *For* $1 \leq i \leq K$,

$$L_i(t) = \frac{1}{K} \sum_{1 \leq k \leq K} C_k(t) + \frac{1}{K(K-2)} \sum_{1 \leq k \leq K,\, k \neq i} \ \sum_{1 \leq h \leq K,\, k,i \in S_h} (\alpha_h(k) - \alpha_h(i)).$$

Proof Since $C_k(t_h) - C_i(t_h) = C_k(t) - C_i(t)$, for $1 \leq k \leq K$ such that $k \neq i$ we have

$$\begin{aligned} D_{k,i} &= \frac{1}{K-2} \sum_{1 \leq h \leq K,\, k,i \in S_h} (V_h(k) - V_h(i)) \\ &= \frac{1}{K-2} \sum_{1 \leq h \leq K,\, k,i \in S_h} ((C_k(t_h) + \alpha_h(k)) - (C_i(t_h) + \alpha_h(i))) \\ &= C_k(t) - C_i(t) + \frac{1}{K-2} \sum_{1 \leq h \leq K,\, k,i \in S_h} (\alpha_h(k) - \alpha_h(i)). \end{aligned}$$

Thus

$$\begin{aligned} L_i(t) &= C_i(t) + A_i(t) \\ &= C_i(t) + \frac{1}{K} \sum_{1 \leq k \leq K,\, k \neq i} D_{k,i} \\ &= C_i(t) + \frac{1}{K} \sum_{1 \leq k \leq K,\, k \neq i} (C_k(t) - C_i(t)) \\ &\quad + \frac{1}{K(K-2)} \sum_{1 \leq k \leq K,\, k \neq i} \ \sum_{1 \leq h \leq K,\, k,i \in S_h} (\alpha_h(k) - \alpha_h(i)) \\ &= \frac{1}{K} \sum_{1 \leq k \leq K} C_k(t) + \frac{1}{K(K-2)} \sum_{1 \leq k \leq K,\, k \neq i} \ \sum_{1 \leq h \leq K,\, k,i \in S_h} (\alpha_h(k) - \alpha_h(i)). \end{aligned}$$

□

Lemma 4 *For $K+1 \leq i \leq n$,*

$$L_i(t) = \frac{1}{K} \sum_{1 \leq k \leq K} C_k(t) + \frac{1}{K(K-1)} \sum_{1 \leq k \leq K} \sum_{1 \leq h \leq K,\, k \in S_h} (\alpha_h(k) - \alpha_h(i)).$$

Proof The proof is essentially identical to that of the previous lemma. We now have a factor of $(K-1)$ in the denominator of the second term rather than $(K-2)$; this difference is due to the different definitions of $D_{k,i}$ depending on whether $1 \leq i \leq K$ or $K+1 \leq i \leq n$. We leave details to the reader. □

Lemma 5 *For any $1 \leq i, j \leq K$ such that $i \neq j$, $|L_i(t) - L_j(t)| \leq (1 + \frac{1}{K})\epsilon$.*

Proof By Lemma 3 and elementary manipulation of the double summation,

$$\begin{aligned} L_i(t) &= \frac{1}{K} \sum_{1 \leq k \leq K} C_k(t) + \frac{1}{K(K-2)} \left(\sum_{1 \leq h \leq K,\, i,j \in S_h} (\alpha_h(j) - \alpha_h(i)) \right. \\ &\quad \left. + \sum_{1 \leq k \leq K,\, k \neq i,j} \left(\sum_{1 \leq h \leq K,\, k,i,j \in S_h} \alpha_h(k) + \alpha_j(k) - \sum_{1 \leq h \leq K,\, k,i \in S_h} \alpha_h(i) \right) \right). \end{aligned}$$

Similarly,

$$\begin{aligned} L_j(t) &= \frac{1}{K} \sum_{1 \leq k \leq K} C_k(t) + \frac{1}{K(K-2)} \left(\sum_{1 \leq h \leq K,\, i,j \in S_h} (\alpha_h(i) - \alpha_h(j)) \right. \\ &\quad \left. + \sum_{1 \leq k \leq K,\, k \neq i,j} \left(\sum_{1 \leq h \leq K,\, k,i,j \in S_h} \alpha_h(k) + \alpha_i(k) - \sum_{1 \leq h \leq K,\, k,j \in S_h} \alpha_h(j) \right) \right). \end{aligned}$$

Taking into account that the two expressions given above have common terms, we obtain

$$L_i(t) - L_j(t) = \frac{1}{K(K-2)}(X - Y)$$

where

$$X = 2 \sum_{1 \leq h \leq K,\, i,j \in S_h} \alpha_h(j) + \sum_{1 \leq k \leq K,\, k \neq i,j} \left(\alpha_j(k) + \sum_{1 \leq h \leq K,\, k,j \in S_h} \alpha_h(j) \right)$$

and

$$Y = 2 \sum_{1 \leq h \leq K,\, i,j \in S_h} \alpha_h(i) + \sum_{1 \leq k \leq K,\, k \neq i,j} \left(\alpha_i(k) + \sum_{1 \leq h \leq K,\, k,i \in S_h} \alpha_h(i) \right).$$

Since $0 \leq \alpha_h(k) \leq \epsilon$ for any $1 \leq h \leq K$ and $k \in S_h$,

$$0 \leq X, Y \leq (2(K-2) + (K-2)(1 + (K-2)))\epsilon = (K-2)(K+1)\epsilon.$$

Therefore we have

$$\begin{aligned} |L_i(t) - L_j(t)| &\leq \frac{1}{K(K-2)}(K-2)(K+1)\epsilon \\ &= (1 + \frac{1}{K})\epsilon. \end{aligned}$$

□

Lemma 6 *For any $1 \leq i \leq K$ and $K+1 \leq j \leq n$, $|L_i(t) - L_j(t)| \leq (1 + \frac{1}{K})\epsilon$.*

Proof The proof is essentially the same as that of Lemma 5, using Lemma 4 instead of Lemma 3; we leave details to the reader. □

Lemma 7 *For all i, j such that $K+1 \leq i, j \leq n$ and $i \neq j$, we have $|L_i(t) - L_j(t)| \leq \epsilon$.*

Proof By Lemma 4,

$$L_i(t) - L_j(t) = \frac{1}{K(K-1)} \sum_{1 \leq k \leq K} \sum_{1 \leq h \leq K,\, k \in S_h} (\alpha_h(j) - \alpha_h(i)).$$

Since $0 \leq \alpha_h(k) \leq \epsilon$ for any $1 \leq h \leq K$ and $k \in S_h$,

$$|L_i(t) - L_j(t)| \leq \frac{1}{K(K-1)} K(K-1)\epsilon = \epsilon.$$

□

This completes the proof of Theorem 4. □

We now prove a matching lower bound. Again, it turns out that the key issue is not how many $(n-1)$-casts are actually sent, but the number of distinct subsets of processes of size $n-1$ that receive $(n-1)$-casts.

Theorem 5 *The logical clocks of $n \geq 3$ processes cannot be synchronized any more closely than $(1 + \frac{1}{K})\epsilon$ by using $(n-1)$-casts and additional point-to-point communication, if $(n-1)$-casts cannot be sent to more than K distinct subsets of processes of size $n-1$, for $3 \leq K \leq n$.*

Proof The proof again uses the many-scenario technique, although we have to work a bit harder to construct the scenarios in this case. Fix an algorithm which synchronizes the logical clocks to within γ without sending $(n-1)$-casts to more than K distinct subsets of processes of size $n-1$. Without loss of generality let s_0 be a scenario in which

1. each of $P_{K+1}, \ldots, P_n$ receives all $(n-1)$-casts,
2. the transmission time of every point-to-point message is $(1 + \frac{1}{K})\epsilon$, and
3. the transmission time of broadcast messages is as given below.

Think of processes $P_1, \ldots, P_K$ as being on a ring, and define the "distance" from P_i to P_j $(1 \leq i, j \leq K)$ on the ring by:

$$d(i,j) = \begin{cases} j - i & \text{if } i \leq j \\ K + j - i & \text{if } i > j \end{cases}$$

For each $1 \leq i \leq K$, let S_i be the set of all processes other than P_i. Then in s_0, for each $1 \leq i \leq K$, the transmission time $\tau_{S_i,j}$ of the message to P_j $(j \neq i)$ in every $(n-1)$-cast to the processes in S_i from any process is:

$$\tau_{S_i,j} = \begin{cases} (2 - \frac{d(i,j)}{K})\epsilon & \text{if } 1 \leq j \leq K \text{ and } j \neq i \\ 2\epsilon & \text{if } K+1 \leq j \leq n \end{cases}$$

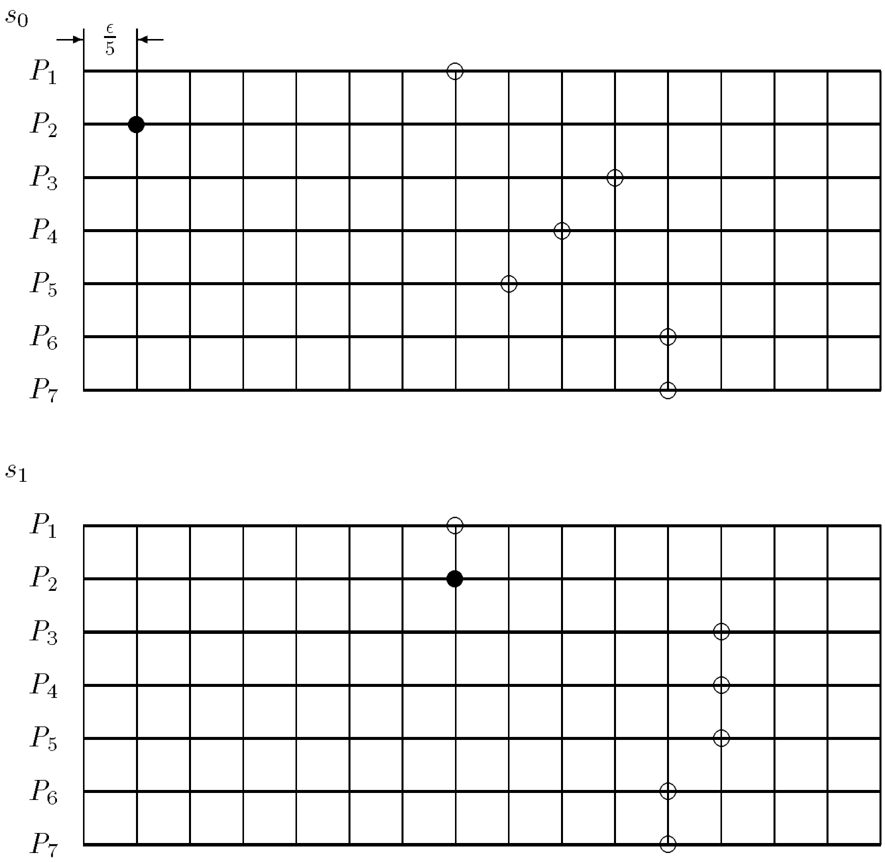

Figure 3: A broadcast of P_2 to $P_1, P_3, \ldots, P_7$ in s_0 and s_1 in the proof of Theorem 5, where $n = 7$ and $K = 5$.

Note that $\tau_{S_i,j} \geq (1 + \frac{1}{K})\epsilon$ for any $i \neq j$. Clearly s_0 is legal.

Let s_1 be the scenario obtained from s_0 by shifting each P_j by $\rho_{1,j}$, where:

$$\rho_{1,j} = \begin{cases} 0 & \text{if } j = 1 \\ (1 + \frac{1}{K})\epsilon & \text{if } j = 2 \\ \frac{j-1}{K}\epsilon & \text{if } 3 \leq j \leq K \\ 0 & \text{if } K + 1 \leq j \leq n \end{cases}$$

Figure 3 shows a broadcast of P_2 to $P_1, P_3, \ldots, P_7$ in s_0 and s_1, where $n = 7$ and $K = 5$. The messages are sent at '$\bullet$' and received at '$\circ$'.

Now we show that s_1 is legal. First, note that $|\rho_{1,j} - \rho_{1,k}| \leq (1 + \frac{1}{K})\epsilon$ for any $1 \leq j, k \leq n$. This, together with the fact that the transmission time of every message in s_0 is at least $(1 + \frac{1}{K})\epsilon$, ensures that all messages have nonnegative transmission time in s_1. Next, suppose that for some $1 \leq i \leq K$, a process $(n-1)$-casts to the processes in S_i at time t in s_0. By the definition of s_0 and the construction of s_1, the messages are received by P_j $(j \neq i)$ at

$t + \tau_{S_i,j} + \rho_{1,j}$ in s_1, where by elementary analysis we have the following. For $i = 1$:

$$\tau_{S_1,j} + \rho_{1,j} = \begin{cases} 3\epsilon & \text{if } j = 2 \\ 2\epsilon & \text{if } 3 \le j \le n \end{cases}$$

For $i = 2$:

$$\tau_{S_2,j} + \rho_{1,j} = \begin{cases} (1 + \frac{1}{K})\epsilon & \text{if } j = 1 \\ (2 + \frac{1}{K})\epsilon & \text{if } 3 \le j \le K \\ 2\epsilon & \text{if } K + 1 \le j \le n \end{cases}$$

For $3 \le i \le n$:

$$\tau_{S_i,j} + \rho_{1,j} = \begin{cases} (1 + \frac{i-1}{K})\epsilon & \text{if } j = 1 \\ (2 + \frac{i-1}{K})\epsilon & \text{if } j = 2 \\ (1 + \frac{i-1}{K})\epsilon & \text{if } 3 \le j \le i - 1 \\ (2 + \frac{i-1}{K})\epsilon & \text{if } i + 1 \le j \le K \\ 2\epsilon & \text{if } K + 1 \le j \le n \end{cases}$$

Thus the messages are received within an interval of ϵ. Therefore s_1 is legal.

We construct scenarios $s_2, \ldots, s_K$ in a completely symmetrical manner; for $2 \le m \le K$, s_m is the scenario obtained from s_0 by shifting each P_j by $\rho_{m,j}$, where:

$$\rho_{m,j} = \begin{cases} \frac{d(m,j)}{K}\epsilon & \text{if } 1 \le j \le m \\ (1 + \frac{1}{K})\epsilon & \text{if } j = m + 1 \\ \frac{d(m,j)}{K}\epsilon & \text{if } m + 2 \le j \le K \\ 0 & \text{if } K + 1 \le j \le n \end{cases}$$

We omit the proof of the legality of $s_2, \ldots, s_K$, since the argument is similar to that for s_1.

Let t be any time after the algorithm terminates at every process in $s_0, s_1, \ldots, s_K$. Let $L_{i,s_0}(t) = T_i$ for $1 \le i \le K$. By Lemma 1, for $1 \le m \le K - 1$ we have $L_{m,s_m}(t) = T_m$ and $L_{m+1,s_m}(t) = T_{m+1} - (1 + \frac{1}{K})\epsilon$, while $L_{K,s_K}(t) = T_K$ and $L_{1,s_K} = T_1 - (1 + \frac{1}{K})\epsilon$. By the assumption on γ, we have

$$T_m \le T_{m+1} - (1 + \frac{1}{K})\epsilon + \gamma$$

for $1 \le m \le K - 1$ and

$$T_K \le T_1 - (1 + \frac{1}{K})\epsilon + \gamma.$$

By adding the K inequalities we obtain

$$\gamma \ge (1 + \frac{1}{K})\epsilon.$$

□

Of course, from Theorem 5, we immediately obtain:

Corollary 2 *The logical clocks of $n \ge 3$ processes cannot be synchronized any more closely than $(1 + \frac{1}{K})\epsilon$ by using K $(n-1)$-casts and additional point-to-point communication, for $3 \le K \le n$.*

Corollary 3 *The logical clocks of $n \ge 3$ processes cannot be synchronized any more closely than $(1 + \frac{1}{n})\epsilon$ by using $(n-1)$-casts and additional point-to-point communication.*

Proof Since there are only n distinct subsets of size $n-1$, the result follows immediately from Theorem 5. $\square$

We remark that Corollary 3 improves the lower bound of $(1+\frac{1}{n(n-2)})\epsilon$ proved in [16].

The theorems we have proved so far completely characterize the precision of synchronization achievable as a function of the number of $(n-1)$-casts. In our analysis, we have essentially ignored the point-to-point communication required, since we have viewed point-to-point communication as being essentially free compared to the cost of an $(n-1)$-cast. In the case of $K=2$, our algorithm uses one point-to-point message in addition to the two $(n-1)$-casts. As we now show, this point-to-point transmission is essential.

Theorem 6 *The logical clocks of $n \geq 3$ processes cannot be synchronized using two $(n-1)$-casts and no point-to-point messages.*

Proof Suppose that $\mathcal{A}$ is an algorithm that synchronizes the processes to within γ and uses no more two $(n-1)$-casts and no point-to-point communication in every execution. Choose $D > \gamma$, and consider an execution where message transmission time for all messages is $2D$. Let s_1 be the scenario corresponding to this execution. Clearly every process must be the recipient of at least one broadcast message. (If not, suppose that P_1 did not receive any broadcast messages and shift P_1 by $2D$ to obtain an easy contradiction.) Since there are only two $(n-1)$-casts altogether, this means that two of the processes, say P_1 and P_2, are the recipients of only one $(n-1)$-cast each. For definiteness, suppose that P_1 is a recipient of a broadcast by P_i and P_2 is the recipient of a broadcast by P_j. Without loss of generality assume further that in s_1, process P_j sent its broadcast at or before the time that P_i sent its broadcast, which means that P_j sent its broadcast before P_i's broadcast was received. Now let s_2 be the scenario where (a) P_1 is shifted by $2D$, (b) if $i \neq 1$, then the broadcast from P_i that is received by P_1 takes time $4D$ to be delivered to each process, and (c) if $j=1$, then the broadcast from P_1 that is received by P_2 takes time 0 to be delivered to each process. Clearly s_2 is legal. Since P_j sent its broadcast in s_1 before receiving P_i's broadcast, it must be the case that P_j sends the same broadcast at the same time in s_2 and s_1; i.e., the delay in receiving P_i's broadcast in s_2 does not affect P_j's sending its broadcast. As a consequence, it is not hard to see that P_1 and P_2 cannot distinguish s_1 and s_2. The other processes may be able to distinguish s_1 from s_2, but since there is no further communication, they cannot tell P_1 and P_2 about it. Thus P_1 and P_2 perform the same adjustments in s_1 and s_2. We can now conclude by a similar argument to that of Theorem 1 that $\gamma \geq D$, giving us contradiction. We leave details to the reader. $\square$

In our algorithm that synchronized to within $(1+\frac{1}{K})\epsilon$ using K $(n-1)$-casts, we used $2n-2$ point-to-point transmissions ($n-1$ in each of phases 2 and 3). We do not know to what extent this number can be reduced. As we observed, we can replace the $(n-1)$ point-to-point transmissions used in phase 3 by one $(n-1)$-cast. We leave the minimization of the number of point-to-point transmissions as an open problem.

4 Clock synchronization using ℓ-casting

In this section, we consider the tradeoffs that arise if we allow ℓ-casting, for arbitrary values of ℓ, rather than restricting attention to $(n-1)$-casts. We start by considering the case of n-casting.

Theorem 7 *The logical clocks of n processes can be synchronized to within ϵ using one n-cast. They cannot be synchronized any closer than ϵ using any number of broadcasts and point-to-point transmissions.*

Proof The upper bound is trivial. One process, say P_1 sends an n-cast. All processes set their logical clocks to 0 on receipt of the n-cast. Clearly they are synchronized to within ϵ.

For the lower bound, fix an algorithm which synchronizes the logical clocks to within γ. Let s_1 be a scenario in which

1. the transmission time of every message (both broadcast and point-to-point) to P_i, $i \neq 1$, is 2ϵ, and

2. the transmission time of every message to P_1 is ϵ.

Clearly s_1 is legal. Now let s_2 be the result of shifting P_1 by 2ϵ and not shifting the other processes at all. It is easy to check that in s_2 we have

1. the transmission time of every message to P_i, $i \neq 1$, is 2ϵ,

2. the transmission time of every message from P_1 to P_1 is ϵ,

3. the transmission time of every message from P_i to P_1, $i \neq 1$, is 3ϵ, and

4. the transmission time of every message from P_i to P_j, $i, j \neq 1$, is 2ϵ.

Thus s_2 is a legal scenario. By using the same arguments as in the proof of Theorem 1, we can now show that $\gamma \geq \epsilon$. This gives us the desired lower bound. $\square$

We now turn our attention to ℓ-casting with $2 \leq \ell \leq n-1$. We again restrict attention to $n \geq 3$, since otherwise, with $\ell \leq n-1$, we do not have any broadcast messages, so we know by Theorem 1 we cannot synchronize at all. Using $C(n, \ell)$ to denote n choose ℓ (i.e., $\frac{n!}{(n-\ell)!\,\ell!}$), we can generalize the upper bound argument of Theorem 4 to get:

Theorem 8 *The logical clocks of $n \geq 3$ processes can be synchronized to within $(1 + \frac{n-\ell}{n})\epsilon$ by using $C(n, \ell)$ ℓ-casts and $2n-2$ point-to-point messages, for $2 \leq \ell \leq n-1$.*

Proof The proof is similar to that of Theorem 4, so we just sketch the details here.

Again, the algorithm consists of three phases. It uses $C(n, \ell)$ ℓ-casts in Phase 1 and $n-1$ point-to-point messages in each of Phases 2 and 3.

Phase 1 Let $S_1, S_2, \ldots, S_M$ be the distinct ℓ-element subsets of $\{1, 2, \ldots, n\}$, where $M = C(n, \ell)$. For each $1 \leq h \leq M$, choose an arbitrary process and let it ℓ-cast to the processes P_i such that $i \in S_h$.

Phase 2 Each process sends one fixed process, say P_1, a message containing the times (on its physical clock) that each of the broadcast messages arrived. From this information P_1 computes a view V_h with domain S_h for each $1 \leq h \leq M$, and then computes $A_1, A_2, \ldots, A_n$ as follows. For $1 \leq i \leq n$,

$$A_i = \frac{1}{n} \sum_{1 \leq k \leq n,\, k \neq i} D_{k,i}$$

where for $1 \le k \le n$ such that $k \neq i$,

$$D_{k,i} = \frac{1}{C(n-2,\ell-2)} \sum_{1 \le h \le M,\, k,i \in S_h} (V_h(k) - V_h(i)).$$

Phase 3 For each $2 \le i \le n$, P_1 sends P_i a message containing A_i.

Again, by Lemma 2, for each view V_h there exist t_h and a function $\alpha_h : S_h \to [0, \epsilon]$ such that $V_h(i) = C_i(t_h) + \alpha_h(i)$ for each $i \in S_h$. In order to prove the correctness of the algorithm, we again use a sequence of lemmas analogous to those used in Theorem 4. Fix an execution of the algorithm and take t to be a time after the algorithm has terminated in that execution.

Lemma 8 *For* $1 \le i \le n$,

$$L_i(t) = \frac{1}{n} \sum_{1 \le k \le n} C_k(t) + \frac{1}{nC(n-2,\ell-2)} \sum_{1 \le k \le n,\, k \neq i} \; \sum_{1 \le h \le M,\, k,i \in S_h} (\alpha_h(k) - \alpha_h(i)).$$

Proof The proof of this lemma is very similar to that of Lemma 3, so again we leave details to the reader. □

Lemma 9 *For any* $1 \le i, j \le n$ *such that* $i \neq j$, $|L_i(t) - L_j(t)| \le (1 + \frac{n-\ell}{n})\epsilon$.

Proof This lemma is a generalization of Lemma 5. The details of the proof are essentially the same; so we omit them here. □

This completes the proof of Theorem 8. □

We conjecture that the upper bound attained in Theorem 8 is tight, but we have only been able to prove this thus far for the case $\ell = 2$ and $\ell = n-1$. The case $\ell = n-1$ follows from Theorem 5. The case $\ell = 2$ follows from the following theorem.

Theorem 9 *The logical clocks of* n *processes cannot be synchronized any closer than* $(1 + \frac{n-2}{n})\epsilon$ *using* 2*-casting and any number of point-to-point transmissions.*

Proof Fix an algorithm which synchronizes the logical clocks to within γ. Let s_0 be a scenario in which

1. in every 2-cast to P_i and P_j where $i < j$, the transmission time of the messages to P_i and P_j is 2ϵ and 3ϵ, respectively, and
2. the transmission time of every point-to-point message is 2ϵ.

Clearly s_0 is legal. Now for each $1 \le m \le n-1$, let s_m be the scenario obtained from s_0 by shifting each of $P_1, \ldots, P_m$ by 2ϵ and not shifting the other processes at all. It is easy to check that in s_m, $1 \le m \le n-1$, we have

1. the transmission time of every message (both broadcast and point-to-point) is non-negative,

2. in every 2-cast to P_i and P_j where $1 \leq i < j \leq m$ or $m+1 \leq i < j \leq n$, P_i receives the message ϵ earlier than P_j, and

3. in every 2-cast to P_i and P_j where $1 \leq i \leq m$ and $m+1 \leq j \leq n$, P_i receives the message ϵ later than P_j.

Thus s_m is a legal scenario.

Let t be any time after the algorithm terminates at every process in $s_0, s_1, \ldots, s_{n-1}$. Let $L_{i,s_0}(t) = T_i$ for $1 \leq i \leq n$. By Lemma 1, for $1 \leq m \leq n-1$ we have $L_{m,s_m}(t) = T_m - 2\epsilon$ and $L_{m+1,s_m}(t) = T_{m+1}$. By the assumption on γ, we have

$$T_1 \leq T_n + \gamma$$

and

$$T_{m+1} \leq T_m - 2\epsilon + \gamma$$

for $1 \leq m \leq n-1$. By adding the n inequalities we obtain

$$\gamma \geq (1 + \frac{n-2}{n})\epsilon.$$

□

Although we cannot prove a matching lower bound to the upper bound of Theorem 8 for $2 < \ell < n-1$, we can prove some nontrivial lower bounds.

Theorem 10 *For $2 < \ell < n-1$, the logical clocks of n processes cannot be synchronized any closer than $\max(\frac{n-1}{n}\frac{\ell}{\ell-1}\epsilon, (1+\frac{1}{\ell+1})\epsilon)$ using ℓ-casting and any number of point-to-point transmissions.*

Proof The lower bound of $(1+\frac{1}{\ell+1})\epsilon$ follows using essentially the same proof as the lower bound for $(n-1)$-casts in Theorem 5 with $K = n$. The only difference is that we treat processes $P_{\ell+1}, \ldots, P_n$ as one process, and then use the lower bound for ℓ-casting with $\ell+1$ processes. If an ℓ-cast is sent to a subset that includes k processes among $P_{\ell+1}, \ldots, P_n$, messages to all these processes take exactly the same amount of time as it would take a message to reach process $P_{\ell+1}$ if there were exactly $\ell+1$ processes in the system. The scenario ends up looking as if an $(\ell-k+1)$-cast were sent instead of an ℓ-cast. We leave details to the reader. The proof of the lower bound of $\frac{n-1}{n}\frac{\ell}{\ell-1}\epsilon$ is similar to that of the lower bound for 2-casts in Theorem 9. The only difference is that we start with a scenario s_0 in which

1. in every ℓ-cast to $P_{i_1}, P_{i_2}, \ldots, P_{i_\ell}$ where $i_1 < i_2 < \ldots < i_\ell$, the transmission time of the message to P_{i_j}, $1 \leq j \leq \ell$, is $\frac{\ell+j-1}{\ell-1}\epsilon$, and

2. the transmission time of every point-to-point message is $\frac{\ell}{\ell-1}\epsilon$,

and then for each $1 \leq m \leq n-1$, we let s_m be the scenario obtained from s_0 by shifting each of $P_1, \ldots, P_m$ by $\frac{\ell}{\ell-1}\epsilon$ and not shifting the other processes at all. The legality of the scenarios $s_0, s_1, \ldots, s_{n-1}$ is immediate, and the rest of the argument is basically the same as that in the proof of Theorem 9. We leave details to the reader. □

5 Concluding remarks

We have investigated the power of broadcasting in the context of clock synchronization. We have not solved (and, in fact, have not closely investigated) all the questions in the area. For example, one question we have not considered in the context of ℓ-casting is the number of ℓ-casts and point-to-point transmissions required in order to achieve optimal synchronization. Notice that the algorithm in Theorem 8 required $C(n, \ell)$ ℓ-casts. We conjecture that these ℓ-casts are all required.

However, our primary goal has not been to solve all the problems that can be solved in the area, but rather to demonstrate the tradeoffs between the type and number of broadcasting, and the precision of synchronization attainable. Our results suggest, for example, that for values of ℓ for which $(n-1)$-casting is not much more expensive than ℓ-casting, $(n-1)$-casting is preferable for achieving clock synchronization. If broadcasting is expensive, then our results suggest that 2-casting might be the method of preference.

There have been other recent papers considering tradeoffs in the presence of k-casting. Dolev and Dwork have observed tradeoffs in the context of Byzantine agreement [4]. They consider a primitive they call a *conference call of size k*, where for any set S of k participants, whenever any one participant p utters a message m, the statement "p sent m" becomes common knowledge among the members of S. Thus, a conference call of size k is essentially a k-cast where all messages are guaranteed to arrive simultaneously. Dolev and Dwork show that more faults can be tolerated by using conference calls of size k than by using conference calls of size ℓ for $k > \ell$. Other tradeoffs are considered in [2]; roughly speaking, this paper shows that if we restrict to omission failures (where a process may be faulty by omitting to send a message), then if there are up to t faulty processes, Byzantine agreement can be achieved in $t-k+3$ rounds using k-casting, for $2 \leq k \leq t+1$. Moreover, they prove a matching lower bound. Finally, in [1] it is shown (at least for the particular algorithm they consider), the same precision of clock synchronization using k-casting is achievable in the presence of failures, independent of the k, for all $k < n$. It would be interesting to understand these tradeoffs for other application areas.

Acknowledgments: We would like to thank Vassos Hadzilacos for his helpful comments on an earlier version of this paper. We are also grateful to the anonymous referees for their helpful comments and Fred Schneider for bringing [1] to our attention.

References

[1] Ö. Babaoğlu and R. Drummond, "(Almost) no cost clock synchronization," *Proc. of the 17th International Symposium on Fault-Tolerant Computing*, 1987, pp. 42–47.

[2] Ö. Babaoğlu, P. Stephenson and R. Drummond, "Reliable broadcast and communication models: tradeoffs and lower bounds," *Distributed Computing* **2**:4, 1988, pp. 177–189.

[3] D. Cheriton, "The V distributed system," *Communications of the ACM* **31**:3, 1988, pp. 314–332.

[4] D. Dolev and C. Dwork, "On-the-fly generation of names and communication primitives," unpublished manuscript, 1990.

[5] D. Dolev, J. Halpern and R. Strong, "On the possibility and impossibility of achieving clock synchronization," *J. Computer and System Sciences* **32**, 1986, pp. 230–250.

[6] M. J. Fischer and N. A. Lynch and M. Merritt, "Easy impossibility proofs for distributed consensus problems," *Distributed Computing* **1**:1, 1986, pp. 26–39.

[7] J. N. Gray, "The Cost of Messages," *Proc. of the 7th ACM Symp. on Principles of Distributed Computing*, 1988, pp. 1–7.

[8] J. Halpern, N. Megiddo and A. Munshi, "Optimal precision in the presence of uncertainty," *Journal of Complexity* **1**, 1985, pp. 170–196.

[9] J. Halpern, B. Simons, R. Strong and D. Dolev, "Fault-tolerant clock synchronization," *Proc. 3rd Annual ACM Symposium on Principles of Distributed Computing*, Vancouver, Canada, 1984, pp. 89–102.

[10] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM* **21**:7, 1978, pp. 558–565.

[11] L. Lamport and P. M. Melliar-Smith, "Synchronizing clocks in the presence of faults," *Journal of the ACM* **32**:1, 1985, pp. 52–78.

[12] J. Lundelius and N. Lynch, "An upper and lower bound for clock synchronization," *Information and Control* **62**, 1984, pp. 190–204.

[13] J. Lundelius and N. Lynch, "A new fault-tolerant algorithm for clock synchronization," *Information and Computation* **77**, 1988, pp. 1–36.

[14] R. Metcalf and D. Boggs, "Ethernet: distributed packet switching for local computer networks," *Communications of the ACM* **19**:7 1976, pp. 395–404.

[15] T. K. Srikanth and S. Toueg, "Optimal clock synchronization," *Journal of the ACM* **34**:3, 1987, pp. 626–645.

[16] K. Sugihara and I. Suzuki, "Nearly optimal clock synchronization under unbounded message transmission time," *Proc. 1988 International Conference on Parallel Processing* **III**, St. Charles, Illinois, 1988, pp. 14–17.