Optimal Clock Synchronization under Energy Constraints in Wireless Ad-Hoc Networks
(Extended Abstract)

Hagit Attiya[1], David Hay[1], and Jennifer L. Welch[2]

[1] Computer Science Department, Technion, Israel
[2] Computer Science Department, Texas A&M University, USA

Abstract. Clock synchronization is a crucial service in many distributed systems, including wireless ad-hoc networks. This paper studies *external* clock synchronization, in which nodes should bring their clocks close to the value of some external reference time, which is provided in the system by one or more *source* clocks.
Reference broadcast synchronization (RBS) is a known approach that exploits the broadcast nature of wireless networks for a single hop. However, when networks are large in physical extent, additional mechanisms must be employed.
Using multi-hop algorithms that re-broadcast time information to short distances reduces the energy consumed for clock synchronization. The reason is that energy costs grow more than linearly with the broadcast distance. On the other hand, the quality of the clock synchronization, as measured in the closeness of the clocks, deteriorates as the number of hops increases.
This paper shows how to balance these two contradictory goals, achieving optimal clock synchronization while adhering to an energy budget at each node. In particular, a distributed algorithm is presented that uses multi-hop broadcasting over a shallow infrastructure to synchronize the clocks. The closeness of clock synchronization achieved by the algorithm is proved to be optimal for the given energy constraints.
Keywords: Clock Synchronization, Wireless Networks, Ad-hoc Networks, Multi-hop Broadcasts.

1 Introduction

Multi-hop ad-hoc networks consist of a collection of computing devices (called *nodes*) communicating by wireless broadcast. Their simplicity and the fact they do not require a pre-existing, wired infrastructure, have made them very popular. A broadcast transmission reaches all nodes within a specific *range* of its source; in order to reach a node outside this range, one or more intermediate broadcast transmissions are used. The transmission range of a broadcast is determined by the power used by the node: the power needed to broadcast to distance d is γd^{β}, where $\beta \geq 1$ is the *distance-power gradient*, which depends on the environmental conditions of the network, and $\gamma > 0$ is the *transmission quality parameter*.

Many applications in distributed systems need synchronized clocks to work correctly. Such applications need to relate the local occurrence time of events to some reference time that is provided in the system by (possibly multiple) *sources*, which are perfectly synchronized with each other (e.g., by relying on UTC [32]). *Reference broadcast synchronization* (RBS) [12] is a known approach for clock synchronization in wireless ad-hoc networks, which broadcasts reference time on a single hop. When networks are large in physical extent, RBS is extended to use multi-hop re-broadcasting [23].

An important parameter for evaluating clock synchronization algorithms is the *clock skew* it provides, namely, the maximal (absolute) difference between the reading of a local clock and the reference time. The theory of clock synchronization [3,18,25,28] indicates that clock skew depends on the *uncertainty* about the accumulated delay of delivering messages between the nodes. For example, in case the uncertainty about the delay experienced by each broadcast message is the same, the clock skew increases as the number of intermediate hops increases.

In wireless ad-hoc networks, energy is typically a scarce resource, which should be used sparingly. Additionally, the likelihood of interference among transmissions increases with their transmission power [19]. In the common case, where $\beta > 1$, the energy costs grow super-linearly with the broadcast distance, making it more energy-economic to broadcast information in several hops, rather than transmitting at the maximum power and reaching far away nodes in a single broadcast hop.

This paper considers clock synchronization in multi-hop ad-hoc networks from a new angle, relating the energy constraints on a clock synchronization algorithm with the optimal skew it may obtain. Given individual energy budgets indicating how much a node is willing to spend on each message transmission for clock synchronization, we give exact bounds on the optimal skew that can be achieved, as a function of the uncertainty in message delays.

As we show, the optimal skew is the minimal depth of a particular *spanning forest*—whose trees are rooted at the time sources—of the topology graph induced by respecting the energy budgets. For the upper bound, we describe how to construct a *shallow* spanning forest, whose depth is minimal with respect to the accumulated uncertainty. This is done by applying simple (centralized or distributed) algorithms for finding *breath-first search* trees from multiple sources. Then, the source time is propagated down the trees. The skew attained by this simple algorithm is the sum of the uncertainties along the path from a source to a node. The matching lower bound is shown using well-known shifting techniques.

This paper considers *external* clock synchronization, in which there is at least one node in the system with access to the reference, or source, time. Applications sometimes require only *internal* (or *mutual*) clock synchronization, in which the local clocks of the nodes are brought close to each other but with no necessary relation to a reference time. Clearly, by invoking an external clock synchronization algorithm **A** in which an arbitrary node plays the role of the time source, it is possible to achieve internal clock synchronization with skew that is at most twice the skew of **A**.

Theoretical study of the clock synchronization problem dates back to the early 1980's. In particular, Lundelius and Lynch [20], and later Halpern et al. [18], proved tight bounds on the best skew achieved by *internal* clock synchronization algorithms. Interestingly, our bound is a simple expression depending on the (weighted) depth of the tree, no matter how complicated the topology is. This stands in contrast to the result in [18] for internal synchronization, in which the tight bound on clock skew is only characterized as the solution to a particular linear program. Later work [3,25,28] considered the issue of internal and external clock synchronization algorithms that exploit specific assumptions on message delay. Recently, Fan and Lynch proved lower [16] and upper [15] bounds on clock skew for the problem of *gradient* clock synchronization, in which the skew between nodes' clocks should be smaller if they are closer to each other.

Clock synchronization in wireless ad-hoc networks has been the topic of extensive research over the last few years (see the survey papers [30, 31] as well as surveys in Elson and Römer [14] and in Cao's thesis [9]). RBS [12], mentioned above, broke new ground by exploiting the nature of wireless broadcasts to get improved performance. Römer's algorithm [29] assigns timestamps to events, which can be compared to estimate the relative times at which events occurred; it was designed for systems with mobile nodes. The local nature of gradient clock synchronization [15, 16] makes it particularly well-suited to sensor networks.

Much of the previous work is not directly comparable to the results in this paper, by focusing, for instance, on one-hop networks, on internal clock synchronization, on how neighboring nodes estimate each other's clock values, or by evaluating the performance solely through experiments or simulation. Here we discuss more related papers, in which either the approach or algorithm is similar to ours or some notion of optimality is proved.

Several papers have addressed the issue of trading off energy and accuracy of the clock synchronization. For instance, PalChaudhuri et al. [26] describe a probabilistic version of RBS [12] and provide an analysis that gives the number of messages and synchronization overhead required to achieve a certain accuracy in the synchronization. van Greunen and Rabaey [33] have a way to devise an algorithm with minimum number of messages to achieve a certain accuracy. Finally, in the context of RBS-style algorithms, Elson et al. [13] show how to achieve an optimal tradeoff between energy consumption and accuracy. In all these papers, the energy usage is equated with the number of messages sent.

Several papers (e.g., [17, 33]) contain algorithms for clock synchronization that work by constructing and using a spanning tree of the network, preferably one of low depth, and propagating time information down the tree. These papers measure the depth of the trees by hop-count, in contrast to our work, which measures depth based on the accumulated uncertainty; since we allow non-uniform uncertainties on links, the two measures are not equivalent. Our matching lower bound shows that uncertainties must be taken into account when constructing the spanning forest to achieve optimality. Thus, our approach yields more accurate clock synchronization when uncertainties are non-uniform.

The Network Time Protocol (NTP) [22], used to synchronize clocks in the Internet, also uses a spanning forest rooted at time sources to broadcast synchronization messages. This spanning forest is built in an ad-hoc manner, and it minimizes the message delay, while our algorithm minimizes message uncertainty.

A few papers provide lower bounds on some complexity measures, which can be used to show that various algorithms are optimal in that regard. Blum et al. [8] prove a lower bound on the size of the interval for algorithms in which each node keeps an interval in which the real time should lie. Meier et al. [21] present an algorithm for internal synchronization that uses the data in a communication pattern optimally in order to obtain the best synchronization.

2 The System Model

Next we sketch a model of computation that captures our assumptions about nodes and their communication. The formalism is based on that in [4, 7].

We consider a set of n nodes $V = \{v_1, \ldots, v_n\}$, located in the Euclidean plane $\mathbb{R}^2$, that communicate with each other through wireless broadcasts. We assume that nodes are not mobile.

Associated with each broadcast is a distance d; we define the *recipients* of a broadcast performed by node v_i to be the set of nodes whose Euclidean distance from v_i is at most d. The power required for broadcasting to distance d is γd^β, where $\beta \geq 1$ is the *distance-power gradient*, which depends on the environmental conditions of the network, and $\gamma > 0$ is the *transmission quality parameter*; typically, $1 \leq \beta \leq 8$ and γ is normalized to one. We assume messages are of a fixed size, therefore γd^β is proportional to the energy consumed when broadcasting to distance d.

Various *events* can occur at a node, including the arrival of a message as well as internal happenings, e.g., internal timers going off.

Each node v_i has a *hardware clock*, denoted HC_i, which is a function from real time to hardware clock time of the form $HC_i(t) = t + o_i$, where o_i is the *offset* of the hardware clock from real time. This form of the hardware clock corresponds to the situation when hardware clocks have no drift, i.e., these clocks run at the same rate as real time.

Each node is modelled as a deterministic state machine, with a set of states, including subsets of initial and final states, and a transition function. The transition function takes as input the current state of the node, the current value of the hardware clock, and the current event, and produces as output a new state and possibly a message to be broadcast to a certain distance (equivalently, with a certain power). The hardware clock cannot be modified by the node. The state machine encodes the local algorithm executed by the node.

A *history* of a node is an infinite sequence of alternating states and pairs, where each pair consists of an event ϕ and a hardware clock value T. The first state must be an initial state of the node and each subsequent state must follow correctly, according to the node's transition function, from the previous state

and pair. Furthermore, the hardware clock values must form a strictly increasing sequence that is unbounded.

A *timed history* of a node v_i is a history together with an assignment of a real time to each pair such that for each pair (ϕ, T), the time t assigned to it satisfies $HC_i(t) = T$. The value of a node's variable at real time t is the value of the variable in the latest state whose preceding pair is assigned a real time at most t. If t is less than the real time assigned to the first pair in the history, then the value of the variable is that in the first state of the history.

An *execution* is a set of n timed histories, one for each node. For each pair appearing in one timed history, say that of v_i, which causes a message to be broadcast, there must be exactly one pair appearing in the timed history of each broadcast recipient v_j whose event is the receipt of v_i's broadcast. Furthermore, for each pair appearing in a timed history whose event is the receipt of a broadcast, there must be a pair containing the corresponding broadcast. Here we are modelling reliable communication via the broadcasts.

The *delay* of a message (received by a node in an execution) is the difference between the real time when the message is received by the node and the real time when the message is broadcast. Specific instances of the model restrict the *admissible* executions, by making assumptions on message delay. These assumptions can vary from one node to another or even from one broadcast to another.

3 Problem Statement

Our goal is to synchronize the clocks of the non-source nodes as closely as possible without using too much energy.

We start by explaining how nodes adjust their clocks. Each node v_i has an *adjustment variable* adj_i as part of its state whose value it adds to its hardware clock to produce its *logical clock*, denoted LC_i. That is, $LC_i(t) = HC_i(t) + adj_i(t)$ for all real times t.

There is a set $S \subseteq V$ of distinguished *source* nodes that have perfectly synchronized clocks; call this *source time*, denoted $ST(t)$. Thus, $LC_i(t) = ST(t) = t$ for all $v_i \in S$ and all real times t, i.e., the source time is the real time.

The non-source nodes have arbitrary logical and hardware clock values initially.

Informally, every node should set the value of its logical clock so as to minimize the difference between its clock and the source time. More formally, we say a system solves the *external clock synchronization problem* if there exists a value ϵ such that in every admissible execution there exists a real time t_f such that for each node v_i and every real time $t \geq t_f$, v_i is in a final state at time t and $|ST(t) - LC_i(t)| \leq \epsilon$. We call the quantity $|ST(t) - LC_i(t)|$ the *skew* of v_i's logical clock.

The way we model energy constraints and how they affect the clock synchronization problem is explained next.

Each node v_i has a value, denoted $energy_i$, which bounds the amount of energy it is allowed to use for each transmission. This constraint translates to a bound on the power with which each node is allowed to broadcast. We denote this bound by $power_i$; note that for fixed-size messages, $power_i = c \cdot energy_i$, for some constant c. For a node v_i, let its *neighborhood*, denoted $N(v_i)$, be the set of nodes that can correctly receive a message sent from v_i and send a message to v_i without violating the energy constraints, that is, the set of nodes within Euclidean distance $\frac{1}{\gamma^\beta}\left(\min\{power_i, power_j\}\right)^{\frac{1}{\beta}}$ from v_i.

The neighborhoods induce a *topology graph*, $TG = \langle V, E\rangle$, such that $\langle v_i, v_j\rangle \in E$ if and only if $v_j \in N(v_i)$. Since the neighborhoods are *bidirectional*, the topology graph is in fact undirected (i.e., $\langle v_i, v_j\rangle \in E$ if and only if $\langle v_j, v_i\rangle \in E$).

We also assume that the energy constraints are "feasible", in particular, that they allow each node to communicate (perhaps indirectly) with every other node. This means that TG is strongly-connected.

We assume that for each pair of neighboring nodes v_i and v_j, the behavior of the link from v_i to v_j is the same as that of the link from v_j to v_i regarding the median message delay and the uncertainty in the message delays. Therefore, we associate with each link e in the topology graph, real numbers δ_e and u_e, with $\delta_e > u_e \geq 0$, such that each message on the link e has delay in the range $[\delta_e - u_e, \delta_e + u_e]$, regardless of its direction.

The relationship to the definition of external clock synchronization is that we now consider *admissible* executions to be executions in which the recipients of a broadcast by node v_i are all the neighbors of v_i in TG, and every message on link e has delay within the range $[\delta_e - u_e, \delta_e + u_e]$.

Crucial to our analysis of clock synchronization algorithms is the *weighted* variant of TG, denoted WTG, in which the weight of each edge e is u_e. Consider any path Π in WTG. Let the *uncertainty* of Π, denoted $u(\Pi)$, be $\Sigma_{e\in\Pi} u_e$, that is, the sum of the weights on all the edges in Π. Let $u_{i,j}$ be the minimum, over all paths Π in WTG between v_i and v_j, of the uncertainty $u(\Pi)$ of the path. A path between v_i and v_j that achieves the minimum uncertainty is called a *minimum-uncertainty* path. We define u_i^{min} to be the minimum, over all source nodes v_s, of $u_{s,i}$, i.e., the shortest (weighted) distance from v_i to any source.

We conclude this section with a few observations about our problem statement.

Since the clocks do not drift, we are considering a "one-shot" problem: the nodes execute some algorithm during which they can reset their adjustment variables. Eventually each node finishes the algorithm and makes no further changes to its adjustment variable. After this point, the relative values of their logical clocks stay the same, no further resynchronization takes place, and the skew of a node is unchanged thereafter.

As we have defined the problem, the goal of a clock synchronization algorithm is to minimize the maximum, over all nodes v_i, of the skew of v_i's clock. In fact, in this paper we introduce clock synchronization algorithms that achieve even stronger property: Not only do they minimize the maximum clock skew over all nodes, but in addition they minimize the clock skew for each node v_i separately.

The assumption of bidirectional links in the topology graph is common in protocols for ad-hoc networks, e.g., the *temporally-ordered routing algorithm* (TORA) [27]. The IEEE 802.11 specification also requires bidirectional signaling. The justification for the assumption that the message delays and uncertainties are symmetric is that variation in the delay is mostly due to processing in the nodes and contention in the physical layer. We assume that nodes are roughly the same and running the same software. Since the number of neighbors affects the level of contention and the number of neighbors of a node v_i is approximately the same as the number of neighbors of any neighbor v_j of v_i, we expect the uncertainty of messages from v_i to v_j to be about the same as that of messages from v_j to v_i.

Note that we measure the delay of a specific message as the time elapsing from when the transmitting node starts broadcasting until the time the recipient finishes its processing. Therefore, this quantity includes processing time in both transmitting and recipient nodes, and the accumulated delays of any retransmissions of the message necessary in case the message is not successfully transmitted at first. This implies that the delay of messages corresponding to the same broadcast may be significantly different, and therefore, *receiver-to-receiver synchronization* [12, 31], which exploits the fact that the propagation delay of such correlated messages is about the same, cannot be applied in our model.

4 Multi-Hop Broadcast-Based Clock Synchronization

In this section, we present algorithms for clock synchronization in the presence of multiple sources. First, we describe a generic algorithm for clock synchronization which assumes that there is a spanning forest of the topology graph TG, with each source being the root of a tree in the spanning forest. We prove that the clock skew achieved by the generic algorithm, depends linearly on the *depth* of the forest, namely, the maximum weighted distance of a path from a leaf to a root. (Recall that edge weights are their uncertainties.) Then, we discuss simple methods to construct a shallow spanning forest. Finally, we present an algorithm that synchronizes clocks while constructing the spanning forest.

4.1 A Generic Synchronization Algorithm

We assume there is a spanning forest of TG such that each source is the root of a separate tree, called its *broadcast tree*. Each source wakes up at some time and sends its current clock value over its broadcast tree. When a node gets the message, it adopts that clock value, after adding D to account for the message delay, where D is the sum of the median delays along the path the message has travelled so far. The detailed pseudocode is omitted from this version due to space limitations.

Specifically, the spanning forest is represented by storing in each node v_i an indication of its parent, $parent_i$; if v_i is a source node, we have $parent_i = \perp$. Non-source nodes also keep in a local variable $delay_i$ the sum of the median delays on

all the edges in the path in the broadcast tree between the node and the source. Broadcasts received from a non-parent node are ignored. All broadcasts by node v_i are performed with power level $power_i$.

It is easy to see that the largest error that a neighbor v_i of a source can make is u_e, where e is the edge between v_i and the source. The largest error occurs if the message from the source took the minimum time $\delta_e - u_e$, or the maximum time $\delta_e + u_e$, to arrive. This maximum error propagates linearly with the (weighted) distance, as the wave of messages moves down the tree away from the source.

Clearly, the clock skew of the algorithm is minimized when the spanning forest is *shallow*, namely, when the maximum (weighted) depth over all the trees is minimized. It is straightforward to prove the next theorem:

Theorem 1. *In every admissible execution of the generic algorithm over a shallow spanning forest, the clock skew of each node v_i is eventually at most u_i^{min}.*

4.2 Pre-Computing a BFS Forest

It is simple to construct a shallow spanning forest of the topology graph TG, needed for the generic clock synchronization algorithm. The spanning forest complies with the energy constraints, since it is a subgraph of TG.

One option is to use a *centralized* algorithm to compute the topology graph TG, and then run a generalization of Dijkstra's algorithm [10, 11] to construct a *shortest paths* forest from the sources.

Another option is to use a *distributed* algorithm to compute the spanning forest. Here we describe a two-step process. A simple distributed algorithm constructs the topology graph TG, by computing the neighborhood $N(v_i)$ of each node v_i; the latter task is achieved by broadcasting a message and waiting for acknowledgments. Once the neighborhoods are computed, we can construct the spanning forest by invoking well-known distributed algorithms for finding a Breadth-First Search (BFS) forest rooted at multiple sources (e.g., [1, 5] or OSPF [24]). These algorithms require a node to send messages using its maximal power under the energy restrictions. The reason is to ensure that the calculated communication links actually "exist".

4.3 Computing the BFS Forest On-the-Fly

In this section, we describe an algorithm that does not rely on pre-processing for constructing the broadcast forest. Instead, the forest is built *on the fly*, while doing the clock synchronization for the first time. The broadcast forest constructed by the algorithm can be used in future computations, including, for instance, clock resynchronizations to handle clock drift.

The code is presented in Algorithm 1. Each node has a local variable *uncertainty* which keeps track of the uncertainty associated with the path to a source that has the smallest uncertainty discovered so far. Each node keeps a local variable *parent* indicating which neighbor is its parent in the spanning

Algorithm 1 Clock synchronization without a predefined spanning forest; code for node v_i.

```
Initialization
    parent_i ← ⊥
    updated_i ← false
    if v_i ∈ S then uncertainty_i ← 0                          // a source
    else uncertainty_i ← ∞

Upon beginning the algorithm:                                  // only performed by a source
    broadcast ⟨sync,ST,i,0⟩

Upon receiving ⟨sync,T,j,u⟩ on an edge e:
    if uncertainty_i > u + u_e then                            // u_e is the uncertainty over edge e
        adj_i ← (T + δ_e) − HC_i                               // δ_e is the median delay on the edge e
        uncertainty_i ← u + u_e
        parent_i ← j
        updated_i ← true

Upon a trigger causing a node to broadcast                     // depends on heuristics
    if updated_i then
        updated_i ← false
        broadcast ⟨sync,LC_i,i,uncertainty_i⟩
```

forest. The local variable *updated* is used to control when the node sends broadcasts. Each node knows the median delay δ_e and the uncertainty u_e on each edge e adjacent to it.

The algorithm uses a $\langle sync \rangle$ message with the following fields:

time: the logical time at the sender when the message is sent.
sender: the originator of the message.
uncertainty: the value of the *uncertainty* variable of the sender.

The algorithm is initiated by having each source $s \in S$ send $\langle sync,ST(t),s,0 \rangle$, at some time t. Each time a node v_i receives a $\langle sync \rangle$ message m with uncertainty u on edge e such that $u + u_e$ is smaller than v_i's current uncertainty, v_i has discovered a shorter path to one of the sources. In this case, v_i updates its adjustment variable based on the time in the message m and the median delay for that link, its current parent to be the originator of message m, and its current uncertainty to be $u + u_e$. At some later time, controlled by the *update* variable, v_i broadcasts a new $\langle sync \rangle$ message with the current value of its logical clock.

We show that the algorithm achieves the same properties as the generic clock synchronization scheme. Namely, every node v_i ultimately adopts the clock value $t + D$, where t is the real time at which the message is sent from some source s, and D is the median delay on a path with minimum weight from s to v_i.

The next lemma says that the *uncertainty* variable of a node is the uncertainty of some path to that node from a source and the clock skew is the same value.

Lemma 1. *For all admissible executions, all nodes v_i, and all times t, if $uncertainty_i(t) < \infty$, then there exists a path Π from some source node v_s to v_i such that $uncertainty_i(t) = u(\Pi)$, $parent_i(t)$ is the predecessor of v_i in Π, and $|ST(t) - LC_i(t)| \leq u(\Pi)$.*

Proof. Choose any admissible execution. We consider an alternative representation of the execution, called a "merged execution", in which the separate node history sequences are merged into a single sequence in order of real time, breaking ties by node id.

We proceed by induction on prefixes of the merged execution.

Initially, only source nodes have finite values for their *uncertainty* variables. Each source node v_i has $uncertainty_i$ equal to 0 and $parent_i$ equal to $\perp$, so the lemma is true with Π being the 0-edge path that starts and ends at v_i.

Suppose the lemma is true up to some point in the merged execution. We show that for all choices of the next (event, hardware clock time) pair, the lemma is still true after the event occurs. The only interesting case is when the event is the delivery of a message that causes the recipient to update its *uncertainty* and *parent* variables. Suppose node v_i receives $\langle sync,T,j,u\rangle$ from node v_j over edge e and $u + u_e$ is smaller than $uncertainty_i$. The message was broadcast at some previous time t' that is less than t, and by the construction of the merged execution, the event that caused the message to be broadcast precedes the current event. Thus by the inductive hypothesis, the invariant is true at real time t' with respect to v_j, meaning that there is a path Π' from a source v_s to v_j whose uncertainty is u, and $|ST(t') - LC_j(t')| \leq u(\Pi')$.

Consider the path Π obtained by appending the edge e between v_j and v_i to Π'. By the code, $uncertainty_i(t)$ is set to $u + u_e$, which is $u(\Pi)$, and $parent_i$ is set to v_j, which is v_i's predecessor in Π. We now show that v_i's clock skew is within the stated bound. Since the value T that v_i receives in the message is equal to $LC_j(t')$ and the code executed by v_i causes LC_i to be set to $T + \delta_e$, we have:

$$\begin{aligned}
|ST(t) - LC_i(t)| &= |ST(t) - LC_j(t') - \delta_e| \\
&= |t - LC_j(t') - \delta_e| && \text{source time is real time} \\
&\leq |(t - t') - \delta_e| + |t' - LC_j(t')| \\
&\leq |(t - t') - \delta_e| + u(\Pi') && \text{by the inductive hypothesis} \\
&\leq u_e + u(\Pi') && \text{actual delay differs from } \delta_e \text{ by at most } u_e \\
&= u(\Pi) && \blacksquare
\end{aligned}$$

The next lemma says that the *uncertainty* variable of each node continues to decrease until reaching the uncertainty of a minimum-uncertainty path.

Lemma 2. *For every admissible execution, each source node v_s, and each node v_i, eventually $uncertainty_i \leq u_{s,i}$.*

Proof. Fix an admissible execution and a source node v_s. For each node v_i, consider all the minimum-uncertainty paths between v_s and v_i and let $\ell_{s,i}$ be the length of the shortest such path with respect to the number of edges.

We will show by induction on ℓ that the lemma is true for all nodes v_i with $\ell_{s,i} = \ell$.

Basis: $\ell = 0$. Then v_i is v_s and the lemma is obviously true.

Induction: Suppose the lemma is true for all nodes v_i such that $\ell_{s,i}$ is at most $\ell - 1$. We will show the lemma is true for all nodes v_i with $\ell_{s,i} = \ell$.

Consider any node v_i with $\ell_{s,i} = \ell$. Pick any one of the shortest-length minimum-uncertainty paths between v_s and v_i and let v_j be the predecessor of v_i on this path. By properties of shortest paths, $\ell_{s,j} = \ell - 1$.

By the inductive hypothesis, eventually $\mathit{uncertainty}_j \leq u_{s,j}$. Consider the first time that v_j sets $\mathit{uncertainty}_j$ to a value u' that is at most $u_{s,j}$. At some later time, v_j broadcasts a $\langle \mathit{sync} \rangle$ message containing a value u that is at most u', i.e., at most $u_{s,j}$. Let e be the edge between v_i and v_j. When v_i receives this broadcast, v_i sets $\mathit{uncertainty}_i$ to $u + u_e$ if it is not already at most that value. Since v_j is a predecessor of v_i on a minimum-uncertainty path, $u_{s,i} = u_{s,j} + u_e$. Thus $u + u_e \leq u_{s,i}$, which proves the inductive case. ■

Theorem 2. *For each admissible execution of Algorithm 1 and each node v_i, eventually the clock skew of v_i is u_i^{min}, and parent_i is the predecessor of v_i in the (weighted) path to the closest source.*

Proof. Lemma 2 implies that eventually v_i's *uncertainty* variable is at most that of a minimum-uncertainty path to a closest source. Let v_s be this closest source. Lemma 1 states that v_i's uncertainty always corresponds to some path to a source, so once v_i's uncertainty is at most $u_{s,i}$, it actually is exactly $u_{s,i}$, never gets any smaller, and the clock skew is $u_{s,i} = u_i^{min}$. Lemma 1 also implies that parent_i is the predecessor of v_i in the (weighted) path to v_s. ■

The worst-case message complexity of Algorithm 1 depends on the frequency with which $\langle \mathit{sync} \rangle$ messages are transmitted. It is possible to apply heuristics proposed by Banerjee and Khuller [6], and delay sending $\langle \mathit{sync} \rangle$ messages for a certain time; this creates the effect of waiting for messages on paths with less uncertainty that encounter higher total delay. A careful inspection of our correctness proof reveals that doing so does not compromise the *eventual* convergence of the tree. As Banerjee and Khuller show, careful choice of parameters in this heuristic can significantly reduce the message complexity.

5 Optimality

We prove that the synchronization achieved over a shallow spanning forest is optimal for the topology graph. This implies that our algorithms achieve optimal clock skew for the given energy budget. More specifically, we prove a lower bound of u_i^{min} on the skew of a node v_i. Recall that in our model, nodes may "overhear" any message that is broadcast within their reception distance, namely, by their neighbors in the topology graph. Potentially, overhearing may yield algorithms that provide lower skew than our spanning-forest-based algorithm, in which a

broadcast by a node is only used by its children; our lower bound shows that this is not the case.

The intuition for this result is to think of lifting up a piece of cloth from the table, holding the sources pinned to the table. The node of interest is some point on the cloth. We pinch the cloth at that location and lift it up by u_i^{min}. The nodes in the neighborhood of the node of interest correspond to points on the cloth that will be lifted by smaller amounts; these amounts form a gradient. Using the fact that u_i^{min} is the shortest distance from a source, we can show that the cloth will not tear, namely, the timing at adjacent nodes remains consistent.

The detailed proof relies on a well-known *shifting argument* [20]: Given an execution α and an n-vector $\mathbf{x}$ of real numbers, we define a new execution, denoted $\alpha' = \mathit{shift}(\alpha, \mathbf{x})$, by adding x_i to the real time associated with each event of node v_i's history in α, for all i. Shifting an execution affects the hardware clocks and the message delays, in ways that are quantified in the next lemma.

Lemma 3 (Shifting [4,20]). *Let α be an execution with hardware clocks HC_i, and let $\mathbf{x}$ be an n-vector of real numbers. Then $\alpha' = \mathit{shift}(\alpha, \mathbf{x})$ is an execution with hardware clocks HC_i', where*
(a) $HC_i'(t) = HC_i(t) - x_i$ for all t, and
(b) if the delay of a message in α from v_i to v_j is d, then the delay of this message in α' is $d - x_i + x_j$.

Theorem 3. *For any external clock synchronization algorithm that complies with the energy constraints and every node v_i, there exists an admissible execution in which the final skew of v_i's clock is at least u_i^{min}.*

Proof. Fix an arbitrary external clock synchronization algorithm. Consider the admissible execution α of the algorithm in which the delay of each broadcast on each edge e of TG is the median delay δ_e. Choose any node v_i and let t_f be the real time when v_i has finished the algorithm.

Let σ be the skew of v_i at time t_f in α, i.e., $|ST(t_f) - LC_i(t_f)| = \sigma$. We first assume that $ST(t_f) - LC_i(t_f) \geq 0$.

We build execution α' by shifting execution α such that every node v_j is shifted by u_j^{min}, that is, $\alpha' = \mathit{shift}(\alpha, \mathbf{x})$ where $x_j = u_j^{min}$ for every j. Since the sources are at distance 0 (from themselves), they are not shifted (and they should not be, since they must have the real time).

We must verify that message delays are within the bounds so that α' is admissible. Consider a message m sent from node v_j to node v_k on an edge e in the topology graph TG. We distinguish between three cases:

If $u_j^{min} = u_k^{min}$, then the delay of m is are unaffected, since sender and receiver are shifted by the same amount.

If $u_j^{min} < u_k^{min}$, then since there is an edge between v_j and v_k, the fact that u_k^{min} is a shortest path distance implies that $u_k^{min} \leq u_j^{min} + u_e$ (recall that u_e is the uncertainty in message delay over the edge e). In α, the delay of m is δ_e; hence, by part (b) of Lemma 3, the delay of m in α' is $\delta_e + u_k^{min} - u_j^{min}$. This implies that the delay of m in α' is between δ_e and $\delta_e + u_e$, since $0 < u_k^{min} - u_j^{min} \leq u_e$.

If $u_j^{min} > u_k^{min}$, then since the weighted graph WTG is undirected, we have that $u_j^{min} \leq u_k^{min} + u_e$, and a similar argument shows that the delay of m in α' is between $\delta_e - u_e$ and δ_e.

We conclude that α' is admissible.

Consider what happens to v_i's skew in α'. At real time $t'_f = t_f + u_i^{min}$ in α', v_i has finished the algorithm. In α' we have:

$$\begin{aligned}
|ST'(t'_f) - LC'_i(t'_f)| &= |ST(t'_f) - LC'_i(t'_f)| \quad \text{source clocks do not change} \\
&= |ST(t'_f) - (LC_i(t'_f) - u_i^{min})| \quad \text{by part (a) of Lemma 3} \\
&= |ST(t_f) - LC_i(t_f) + u_i^{min}| \\
&\qquad \text{skew does not change after } t_f < t'_f \\
&\geq u_i^{min} \quad ST(t_f) - LC_i(t_f) \text{ and } u_i^{min} \text{ are positive}
\end{aligned}$$

In the symmetric case where $ST(t_f) - LC_i(t_f)$ in α is negative, we shift execution α by negative amounts instead of positive to produce α'. Namely, $\alpha' = \mathit{shift}(\alpha, \mathbf{x})$, where $x_j = -u_j^{min}$ for each j. The skew of node v_i in execution α' is $|ST(t_f) - LC_i(t_f) - u_i^{min}|$. Since $ST(t_f) - LC_i(t_f)$ is negative and u_i^{min} is positive, the skew is at least u_i^{min}. ■

6 Discussion

In this section we discuss generalizations of our approach, especially to make it more practical, as well as alternative approaches.

Our algorithm proposes a specific way to *estimate* clock readings over one hop (in a single broadcast), and a simple method to *combine* the estimations over several hops (re-broadcasts). However, we consider our key contribution to be the approach of picking the multi-hop broadcasts in a way that minimizes the worst-case accumulated uncertainty, while complying with the energy constraints. It is reasonable to apply other estimation and combination methods over shallow spanning forests of the topology, for example, methods that are geared towards optimizing the expected skew [26]. It would be interesting to prove that shallow energy-constrained spanning forests are optimal under these measures as well, as experimentally observed by van Greunen and Rabaey [33]. It would also be interesting to extend our results to deal with the more theoretical approach of optimizing the skew on a per-execution basis [3, 25, 28].

In this paper, we fixed the energy constraints and found algorithms that achieve optimal skew, under this budget. A complementary approach is to fix the desired skew and look for the minimal-energy algorithm achieving this skew. With uniform uncertainties, our techniques easily translate the desired skew into a desired number of hops, reducing the problem to the task of constructing a *minimal-energy* forest with this depth. For a constant depth and a single source, the tree can be approximated using an algorithm of Ambühl et al. [2]. It would be interesting to handle multiple sources and arbitrary depths.

Our approach can also be extended to apply when the sources are only approximately synchronized, as is the case when sources are in a wired network and run a separate algorithm to synchronize with each other. The approach can also accommodate clock *drift*; this requires clock synchronization to be invoked repeatedly and not only once. Once a shallow spanning forest is constructed, later resynchronization can use the same infrastructure. We leave the development of these ideas, and their analysis, as a topic for future research.

Finally, it is interesting to remove (at least some of) the assumptions we have made, most notably, the immobility of nodes and the reliability of communication links as well as their bidirectionality.

References

1. Y. Afek and M. Ricklin. Sparser: A paradigm for running distributed algorithms. *Journal of Algorithms*, 14:316–328, 1993.
2. C. Ambühl, A. E. F. Clementi, M. D. Ianni, N. Lev-Tov, A. Monti, D. Peleg, G. Rossi, and R. Silvestri. Efficient algorithms for low-energy bounded-hop broadcast in ad-hoc wireless networks. In *Proceedings of the 21st Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 418–427, 2004.
3. H. Attiya, A. Herzberg, and S. Rajsbaum. Optimal clock synchronization under different delay assumptions. *SIAM Journal on Computing*, 25(2):369–389, 1996.
4. H. Attiya and J. L. Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. John Wiley& Sons, second edition, 2004.
5. B. Awerbuch and R. G. Gallager. A new distributed algorithm to find breadth first search trees. *IEEE Transactions on Information Theory*, 33(3):315–322, 1987.
6. S. Banerjee and S. Khuller. A clustering scheme for hierarchical control in multi-hop wireless networks. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM)*, pages 1028–1037, 2001.
7. S. Biaz and J. L. Welch. Closed form bounds for clock synchronization under simple uncertainty assumptions. *Information Processing Letters*, 80:151–157, November 2001.
8. P. Blum, L. Meier, and L. Thiele. Improved interval-based clock synchronization in sensor networks. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 349–358, April 2004.
9. G. Cao. *Distributed Services in Mobile Ad Hoc Networks*. PhD thesis, Texas A&M University, April 2005.
10. T. H. Cormen, C. E. Lieserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.
11. N. Deo and C. Pang. Shortest path algorithms: Taxonomy and annotation. *Networks*, 14(2):275–323, 1984.
12. J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the 5th Symposium on Operating System Design and Implementation (OSDI)*, 2002.
13. J. Elson, R. Karp, C. Papadimitriou, and S. Shenker. Global synchronization in sensornets. In *Proceedings of the 6th Latin American Symposium on Theoretical Informatics (LATIN'04)*, pages 609–624, Buenos Aires, Argentina, 2004.
14. J. Elson and K. Römer. Wireless sensor networks: a new regime for time synchronization. *Computer Communication Review*, 33(1):149–154, 2003.

15. R. Fan, I. Chakraborty, and N. Lynch. Clock synchronization for wireless networks. In *Proceedings of the 8th International Conference on Principles of Distributed Systems (OPODIS)*, 2004.
16. R. Fan and N. Lynch. Gradient clock synchronization. In *Proceedings of the 23rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 320–327, 2004.
17. S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync protocol for sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 138–149, 2003.
18. J. Y. Halpern, N. Megiddo, and A. A. Munshi. Optimal precision in the presence of uncertainty. In *Proceedings of the 17th ACM Symposium on Theory of Computing (STOC)*, pages 346–355, 1985.
19. L. Li, J. Y. Halpern, P. Bahl, Y.-M. Wang, and R. Wattenhofer. Analysis of a cone-based distributed topology control algorithm for wireless multi-hop networks. In *Proceedings of the 20th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 264–273, 2001.
20. J. Lundelius and N. A. Lynch. An upper and lower bound for clock synchronization. *Information and Control*, 62(2/3):190–204, 1984.
21. L. Meier and L. Thiele. Gradient clock synchronization in sensor networks. In *Proceedings of the 24th ACM Symposium on Principles of Distributed Computing (PODC)*, page 238, July 2005.
22. D. L. Mills. Internet time synchronization: The Network Time Protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, Oct. 1991.
23. S. Mitra and J. Rabek. Energy efficient connected clusters for mobile ad hoc networks. In *3rd Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, 2004.
24. J. Moy. RFC 2328 - OSPF version 2, April 1998.
25. R. Ostrovsky and B. Patt-Shamir. Optimal and efficient clock synchronization under drifting clocks. In *Proceedings of the 18th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 3–12, 1999.
26. S. PalChaudhuri, A. K. Saha, and D. B. Johnson. Adaptive clock synchronization in sensor networks. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 340–348, 2004.
27. V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of the Conference on Computer Communications (IEEE INFOCOM)*, pages 1405–1413, 1997.
28. B. Patt-Shamir and S. Rajsbaum. A theory of clock synchronization. In *Proceedings of the 26th ACM Symposium on Theory of Computing (STOC)*, pages 810–819, 1994.
29. K. Römer. Time synchronization in ad hoc networks. In *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc)*, pages 173–182, 2001.
30. F. Sivrikaya and B. Yener. Time synchronization in sensor networks: A survey. *IEEE Network*, pages 45–50, July/August 2004.
31. B. Sundararaman, U. Buy, and A. D. Kshemkalyani. Clock synchronization for wireless sensor networks: A survey. *Ad Hoc Networks*, 3(3):281–323, May 2005.
32. The Official U.S. time. `http://www.time.gov/`.
33. J. van Greunen and J. Rabaey. Lightweight time synchronization for sensor networks. In *Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications (WSNA)*, pages 11–19, 2003.