Clock Synchronization for Distributed Multi-hop Wireless Networks Using Markov Random Field

Wu CHEN, Xin JIN, Zehong LI, Xin ZHANG and Liang HONG
Department of Automation, Northwestern Polytechnical University, Xian 710072, China
chenwu@nwpu.edu.cn

Abstract. The global clock synchronization problem in distributed multi-hop wireless networks is studied in this paper. A fully distributed synchronization solution based on mean field is proposed. A neighbor system and a label problem are defined to describe clock synchronization problem. The clock offset and skew are mapped to clique potential, and an energy function is established based on clique potential. Then the clock synchronization is transformed to finding the minimum of the energy function. By minimizing the energy function, clock synchronization is reached. A distributed energy minimization algorithm is also proposed. The simulation results show the proposed solution achieves high precise global clock synchronization and has the virtue of low overhead, robustness and scalability.

1. Introduction
Clock synchronization is usually required for distributed (sensor or ad hoc) wireless networks. It’s crucial to perform a large number of applications. Data fusion, power management, cooperative transmission and coordinated medium access control (MAC) are enabled by, or benefit from, the availability of a high-accuracy synchronized clock across the network [1], [2].

Unfortunately, it is difficult to achieve high-accuracy clock synchronization because of the lack of infrastructure and common clock in distributed multi-hop wireless networks. Existing clock synchronization solutions can be divided into central node synchronization and mutual synchronization. The central node synchronization solution is that a special node is selected as reference clock, and a spanning tree or cluster-based structure is established to achieve synchronization through node to node pairwise synchronization. It is also called sender-receiver synchronization. Time synchronization protocol for sensor network (TPSN) [3] and pairwise broadcast synchronization (PBS) [4] are the representative solutions. Besides, there is an extension of PBS for global network synchronization which reduces the overall energy consumption [5]. But these solutions have some shortcomings. Most of the tree-based or cluster based solutions are vulnerable to sudden node failures and incur large overhead in building and maintaining the tree or cluster structure [6]. The mutual synchronization is that every node synchronizes with their neighbors by exchanging timing information. A well-known method of mutual synchronization is Reference Broadcast Synchronization (RBS) algorithm [7] which is a kind of receiver-receiver synchronization. There are also some efficient methods exist for RBS such as [8], which can accurately achieve the clock synchronization for nodes placed in the same neighborhood. Furthermore RBS methods can also reduce the time-critical path and improve the synchronization accuracy but it has few drawbacks too. It needs a reference node to send synchronization signals (beacons) periodically and has a mount of redundant broadcast messages. RBS also needs all nodes activity during broadcast. However, this condition is difficult to meet because nodes need to switch between wake and sleep to save energy.
In order to solve the problem of frequently switching between wake and sleep, there are some variations of two-way message exchange synchronization mechanisms that allow synchronization of passive nodes located in the neighborhood of two active nodes that synchronize with each other\[9\] [10]. In order to get rid of dependence on the central node, some fully distributed clock synchronization solutions that no special node needs to be maintained have started to arouse significant interest in recent years. Simeone et al. presented a good introductory survey on clock synchronization in distributed wireless networks [1]. Many solutions have been proposed in the wake of this paper.

The fully distributed solutions can be classified into two major categories. One is clock correction which allows nodes to update their clocks and converge to common values by continuously performing clock updates. The other is relative synchronization where every node has its own clock and runs independently and no clock update is needed. The relative synchronization solution just estimates relative clock skew and offset between neighbor nodes. The synchronization is performed on demand. That is the node which needs synchronization will calculate multi-hop relative parameters by local parameters of intermediate nodes. Many synchronizations based on average algorithms have been proposed [11][12][13][14][15][16]. Some average algorithms use pairwise messages such as average consensus and gossip algorithms. That is all nodes synchronize to an average clock by exchanging pair message with only one selected neighbor node. We call these solutions paired average algorithm. Gang and Shalinee gave the performance analysis of consensus protocol [15]. The conclusion is that the performance is not optimal and will deteriorate when message delay exists. Some more average algorithms get reference clock by averaging the clock offset of all neighboring nodes [16]. More recently, some solutions based on belief propagation (BP) have been proposed in [6] and [17]. The simulation result shows that the performance is superior to consensus algorithms. Leng et al. assumed that message delay is independent and identically distributed (iid) Gaussian random variables. But according to our experiment results, this assumption is not satisfied in many conditions. And a reference node is also assigned in [6] and [17]. There also exist other solutions using relative receiver-to-receiver paradigm. The protocol proposed in [18] just estimates the time difference between one-hop neighbors and no local-clock update is needed. And multi-hop synchronization is performed only on-demand according to the application need. The proposed solution in [6], [7] and [17] are relative synchronization. These solutions just give the offset between the nodes, and do not reach global clock synchronization.

The main contribution of this paper is to propose a fully distributed clock synchronization solution using mean field that each node only requires communicating with its one hop neighbors and is fully distributed and localized. It has the virtue of low overhead, robustness and scalability. The main contribution of mean field includes the following two aspects. One is the definition of the virtual reference clock of neighborhood system. Another is the energy function, which defines the optimal solution of clock synchronization and is a guide to search optimal solution by energy minimization. By minimizing the energy function and finding the minimum energy solution, clock synchronization is reached. The effectiveness of the solution is validated by simulation in Qualnet and Matlab.

The rest of the paper is organized as follows. In Section 2, the clock synchronization model is presented. The distributed clock synchronization algorithm based on mean field is proposed in Section 3. In Section 4, simulation results are presented to demonstrate the performances of the proposed algorithm. In the end, conclusions are drawn in Section 5.

2. Clock synchronization model

2.1. Neighborhood system
In the most common sense, network is represented by a graph $G=(V,E)$, in which $V$ is set of nodes and $E$ is set of links between nodes. In this paper, we take the model network as neighborhood system. A neighborhood system for $G$ is defined as:

$$N = \{N_i|\forall i \in V\}. \quad (1)$$

where $N_i$ is the set of nodes neighboring node $i$. The neighboring relationship has the following
properties [19]:
(1) A site is not neighboring to itself: \( i \notin N_i \)
(2) The neighboring relationship is mutual: \( i \in N_j \iff j \in N_i \)

2.2. Clock synchronization model
Before discussing the fully distributed global clock synchronization, we first consider the clock relationship between two nodes. The clock value \( C_i \) of node \( i \) at time \( t \) is given by:

\[
C_i(t) = \alpha_i \cdot t + \beta_i, \quad i = 1, 2, \ldots, L.
\]

where the parameters \( \alpha_i \) and \( \beta_i \) are called clock skew (frequency difference) and clock offset (phase difference) [2]. \( L \) is the total number of nodes in the network. The task of clock synchronization is to estimate these parameters accurately, and if possible, adjust them to reach clock synchronization. Since the absolute reference time \( t \) is not available in the fully distributed wireless networks, it is not possible to compute the parameters \( \alpha_i \) and \( \beta_i \). Only the relative clock skew and offset can be obtained by exchanging pairwise time stamp messages.

From (2), the clock relation between node \( i \) and node \( j \) can be given as follow:

\[
C_j(t) = \alpha_j \cdot C_i(t) + \beta_j. \tag{3}
\]

where \( \alpha_{ij} \) and \( \beta_{ij} \) mean the relative clock skew and offset between node \( i \) and node \( j \), as shown in Figure 1.

![Figure 1 Clock model between node i and node j](image1)

![Figure 2 Two-way time-stamp exchange between node i and node j](image2)

The value of \( \alpha_{ij} \) and \( \beta_{ij} \) cannot be computed directly. They are estimated by exchanging pairwise time stamp messages as shown in [11], [18] and [20]. That is each node \( i \) is assumed to periodically transmit a packet to all its neighbors with a synchronization period equals to \( T_B \), which is shown in Figure 2. In the k-th round(s), node \( i \) broadcasts a time stamp message, denoted by \( S^k_i \), as measured by its
clock just before the transmission. Upon receiving this packet, the receiving node \( j \) records the time (according to its local clock) just after it receives the packet, denoted by \( R^{k}_{i,j} \). The two-way message exchange signaling mechanism is described in detail in [21].

If clock synchronization between node \( i \) and node \( j \) is reached, \( \alpha_{ij} = 1 \) and \( \beta_{ij} = 0 \) will be realized. If every clock synchronizes to all of its neighbors, the network synchronization will be reached. This state is difficult to achieve but only close. That is to find \( \hat{\alpha}_{ij} \) and \( \hat{\beta}_{ij} \) for every node in the network so that:

\[
\lim_{t \to \infty} \sum_{j \in N_i} \left| 1 - \hat{\alpha}_{ij} \right| = 0 \quad \lim_{t \to \infty} \sum_{j \in N_i} \left| \hat{\beta}_{ij} \right| = 0 \quad i \in V
\] (4)

This is a kind of combinatorial optimization problem, and it is a NP complete problem. It cannot be solved quickly. But in many situations, quick clock synchronization is needed.

Our approach combines (2) and (4). A local reference clock is introduced, namely:

\[
C_i(t) = \alpha_i \cdot t + \beta_i.
\] (5)

where \( \alpha_i \) and \( \beta_i \) are estimated by \( \alpha_{ij} \) and \( \beta_{ij}, i \in N_j \) using mean field method. The goal of our approach is to synchronize all the nodes with respect to the virtual reference clock. Obviously, if two clocks are perfectly synchronized, \( \alpha_{ij} = 1 \) and \( \beta_{ij} = 0 \).

The skew parameter \( \alpha_{ij} \) and offset parameter \( \beta_{ij} \) can be obtained from indirect information about them by exchanging pairwise time stamps of one node I with respect to another node j. There are a lot of solutions to estimate clock skew and offset by exchanging clock information. The skew can be precisely determined as shown in [18]. The offset can only be obtained by estimating. Many quickly and precisely offset estimation methods have been proposed in recent years. Our approach does not focus on the estimation of skew and offset, but assumes they can be obtained.

3. Distributed clock synchronization using mean field

3.1. The model of mean field in clock synchronization

Because the full distributed method has many advantages such as scalability, robustness and flexibility, we try to get rid of the dependence on the central reference node. Inspired by the phenomenon of phase transition [22] which consider a large number of small individual components which interact with each other, we adopted the mean field theory and proposed a mutual synchronization method in this paper. The main idea is that a node sets its reference clock by a single averaged effect which is gotten through the interaction with its neighbor nodes. The proposed mutual synchronization only exchanges clock information among the neighbor nodes. And the clock of node only depends on the clock of its neighbor nodes. In other word, the clock has spatial Markov property.

We need a concept of clique to model the clock synchronization. A clique \( C \) for \( (V, E) \) is subset of nodes which interact with each other. Going a further step, we define single-node clique \( C_i \) and neighboring nodes clique \( C_{z+i} \), where

\[
C_i = \{ i | i \in V \}.
\] (6)

\[
C_{z+i} = \{ i, j | i \in V, j \in N_i \}.
\] (7)

Clique also includes triple neighboring nodes clique \( C_i \), quadruple neighboring nodes clique \( C_{4+i} \), and so on. But we just need \( C_i \) and \( C_{z+i} \) in this paper.

The collection of all cliques is

\[
C = C_i \cup C_{2+i} \cup C_{3+i} \cdots.
\] (8)

where “…” denotes possible sets of larger cliques.
We treat clock synchronization as labeling problem. A label is an event that may happen to a node [19]. A label set may be categorized as being continuous or discrete [19]. In clock synchronization, the label set is continuous and corresponds to the real line \( \mathbb{R} \). The labeling problem is to assign a label from the label set to each of the nodes in \( V \). That is set the clock of node. To assign a label is called configuration in the terminology of random fields. All the nodes have the same label set in clock synchronization, and the configuration space is:

\[
F = \{F_1, F_2, \ldots, F_L\} = \mathbb{R}^L.
\]

where \( L \) is the number of nodes in network. We use the notation \( F_i = f_i \) to denote the event that node \( i \) takes the clock value of \( f_i \), and the notation \( (F_1 = f_1, F_2 = f_2, \ldots, F_L = f_L) \) to denote the whole network clock status. For simplicity, we abbreviate it as \( f_a \), where \( f_a \) is a configuration of \( F \). And \( f_a \) also is a joint event. The probability of the joint event \( f_a \) is denoted \( P(f_a) \). Correspondingly, the probability that node \( i \) is assigned a label \( f_i \), that is node \( i \) takes the clock value \( f_i \), denoted \( P(f_i) \).

\( F \) is a Markov random field on \( G \) w.r.t. a neighborhood system \( N \), because it obviously satisfy the flowing two conditions [19]:

\[
P(f) > 0, \forall f \in F. \tag{10}
\]

\[
P(f_1 \mid f_{i-1}) = P(f_1 \mid f_N). \tag{11}
\]

When every node is assigned the same label:

\[
f_i = f, i \in V. \tag{12}
\]

It can be considered to achieve the clock synchronization. By adjusting the configuration of network, we maximize the probability of \( P(f = f) \). Then the clock synchronization is defined as finding the maximum a posteriori (MAP) configuration of the MRF. As mentioned in [19], Markov random fields are equivalent to Gibbs random fields. So clock configuration obeys Gibbs distribution, and is of the following form:

\[
P(f) = Z^{-1} \exp \left( -\frac{1}{T} U(f) \right). \tag{13}
\]

where

\[
Z = \sum_{f \in F} \exp \left( -\frac{1}{T} U(f) \right). \tag{14}
\]

is the normalizing constant. \( T \) is a free parameter. In statistical physics, it is temperature. It is assumed to be 1 in most application [19]. \( U(f) \) is the energy function and defined as follow:

\[
U(f) = \sum_{c \in C} V_c(f). \tag{15}
\]

The energy is a sum of clique potentials \( V_c(f) \) over all possible cliques \( C \) [19]. It can be expanded as sum of terms:

\[
U(f) = \sum_{|c| = 1} V_c(f) + \sum_{|c| = 2} V_c(f, f_j) + \sum_{|c| = 3} V_c(f, f_j, f_k) + \cdots. \tag{16}
\]

As mentioned above, in clock synchronization case, only cliques of size up to two are considered. Then (16) can be rewritten as:

\[
U(f) = \sum_{|c| = 1} V_c(f) + \sum_{|c| = 2} V_c(f, f_j). \tag{17}
\]

According to (13), minimizing energy \( U(f) \) is equivalent to maximizing the probability of \( P(f) \).

As mentioned previously, we adopted the mean field theory. Ising model is the basic model in mean field theory. The energy function of Ising model is [23]:

\[
E = -H \sum_i S_i - \sum_{ij} J_{ij} S_i S_j. \tag{18}
\]
where $J_{ij}$ is coupled matrix, representing that $i$ and $j$ has an interaction when $J_{ij} \neq 0$. In wireless ad hoc networks, $J_{ij}$ amounts to the relationship among neighbor nodes who can synchronize to each other. $S_i$ amounts to the local clock of node $i$. Obviously, (17) and (18) have the same expression. The goal of Ising model is to minimize the energy function. Finding the MAP configuration of the MRF is equivalent to minimizing energy function. Thus, the clock synchronization can be regarded as a kind of mean field problem.

The following tasks of this paper are to define a suitable function as the measure of the global clock synchronization and to find a distributed minimization algorithm to get global clock synchronization solution more quickly and more precisely.

### 3.2 Define energy function

Energy function is the objective function of clock synchronization and is to be minimized. The role of energy function has two aspects: (1) It defines the optimal solution of clock synchronization; (2) It is a guide to search optimal solution by energy minimization.

As described in (17), energy function is composed of clique potential functions. With reference to [19], pair-site clique potentials are defined by

$$V_{ij}(f_i, f_j) = g(f_i - f_j).$$

(19)

where $g(f_i - f_j)$ is the penalty function of clock difference between nodes $i$ and $j$. The function $g$ must be even and nondecreasing. Quadratic function is the mostly used penalty function, and it is also adopted in this paper:

$$g(f_i - f_j) = (f_i - f_j)^2.$$  

(20)

As to single-site clique potential, it depends on the offset of node clock and reference clock and also has the form of quadratic. It can be described as:

$$V_i(f_i) = (f_i - f_R)^2.$$ 

(21)

where $f_R$ is the reference clock. But as we mentioned in the prior sections above, in fully distributed network, no reference clock needs to be maintained. Therefore, only the statistical method is suitable for the calculation of the reference clock.

Mean field theory gives the idea that the effect of all the other individuals on any given individual is approximated by a single averaged effect [23]. So we use mean field approximation to calculate the reference clock of a neighborhood system. The virtual reference clock of neighborhood system of node $i$ is defined by the mean value:

$$\langle f_R \rangle_i = \sum_{j \in N_i} f_j P(f_j).$$ 

(22)

And now, the whole energy function has the form as follows:

$$U(f) = \sum_{(i,j) \in V} (f_i - \langle f_R \rangle_i)^2 + \sum_{i \in V} \sum_{j \neq i} (f_i - f_j)^2.$$ 

(23)

### 3.3. Energy minimization in parallel

By minimizing (23), and finding the minimum energy solution, the MAP solution is determined and so clock synchronization is reached. The remaining problem is how to find the solution. In (23), the energy function is defined as a global variable. It needs to collect every node’s clique potentials to calculate energy function and find minimum energy solution. This is the approach that is not feasible in multi-hop wireless networks and it will cause large payload. We should find a minimization procedure that is distributed and parallel.

As a minimum must be a stationary point where the gradient of $U(f)$ is a zero vector, the following equation must be satisfied:
\[
\frac{\partial U(f)}{\partial f_i} = 2\sum_{(j \in N_i)} (f_i - f_j)^2 + \sum_{(j \in N_i)} (f_i - f_j)^2, \quad \forall i \in V. \tag{24}
\]

To solve (24), simultaneous clock configuration of every node in network is needed. This is unfeasible in distributed multi-hop wireless networks. We rewrite (24) as follow:

\[
\frac{\partial U(f)}{\partial f_i} = 2\sum_{(j \in N_i)} (f_i - f_j)^2 + \sum_{(j \in N_i)} (f_i - f_j)^2, \quad \forall i \in V. \tag{25}
\]

We define:

\[
\frac{\partial U(f)}{\partial f_i} = 2(f_i - f_j)^2 + 2\sum_{(j \in N_i)} (f_i - f_j), \quad \forall i \in V. \tag{26}
\]

Then:

\[
\frac{\partial U(f)}{\partial f_i} = \sum_{(j \in N_i)} \frac{\partial U(f)}{\partial f_j}, \quad \forall i \in V. \tag{27}
\]

According to (27), we can obtain minimum of global energy function by calculating every node’s minimum energy function. Then, \(U(f)\) can be updated in parallel on every node in the network and only neighborhood information is needed. Therefore, a global minimum can be computed by considering \(f_i\) of each node locally. So we get a fully distributed energy minimization algorithm.

There still exists a problem that the minimum value determined by (24) is a local minimum or global minimum, because the solution of (24) is just a stationary point where the gradient vanishes. It may be an inflection point rather than an extremum. Theoretically, it can be a local minimum because the gradient there is exactly zero. But the local and global minimums of our energy function are exactly the same one. The proof is provided as follows:

1. Property of convex function: Any local minimum of a convex function is also a global minimum. A strictly convex function will have at most one global minimum \([24]\).

2. \(\forall i \in V, \frac{\partial^2 U(f)}{\partial f_i^2} > 0 \Rightarrow \frac{\partial^2 U(f)}{\partial f_i^2} > 0\), So \(U(f)\) is a strictly convex function.

\[
\therefore \text{The local minimum of } U(f) \text{ is the global minimum.}
\]

As mentioned above, the energy function is a nonlinear function, so the solution of energy minimization can be solved only by numerical iterative method.

The algorithm of energy minimization is an iteratively updated procedure using the rule as follow:

\[
f_i^{k+1} = f_i^k - 2\mu \sum_{(j \in N_i)} (f_i^k - f_j^k) \tag{28}
\]

where \(\mu\) is a small constant which is the step size in gradient descent.

3.4. Clock skew synchronization

Obviously, if two clocks of node \(i\) and \(j\) are perfectly skew synchronized, \(\alpha_i - \alpha_j = 0\). Unfortunately \(\alpha_i\) and \(\alpha_j\) are unavailable in the absence of a reference clock. The main idea of our proposal is to make all nodes synchronize to virtual reference clock skew, and minimizes \(\hat{\alpha}_i - \hat{\alpha}_j\) concurrently using Eq.(28).

Local clock skew \(\hat{\alpha}_i\) is estimated by (29), as follow:

\[
\hat{\alpha}_i^{k+1} = \hat{\alpha}_i^k - 2\mu \sum_{(j \in N_i)} (\hat{\alpha}_i^k - \hat{\alpha}_j^k) \tag{29}
\]

where \(\hat{\alpha}_j^k\) is the reference clock skew of node \(i\).

The neighbor clock skew \(\hat{\alpha}_j\) is estimated by the relative skew which is defined as:

\[
\alpha_j = \frac{\alpha_j}{\alpha_i}, \quad i, j \in V. \tag{30}
\]

\(\alpha_j\) is the total skew of neighbor node \(j\) with respect to \(i\).
The value of $\alpha_j$ cannot be computed by (30) directly. It is estimated by exchanging pairwise time stamp messages as mentioned previously. Then the relative skew $\alpha_{ij}$ can be obtained as below:

$$\hat{\alpha}_{ij}^{k+1} = \frac{R_{ij}^{k+1} - R_{ij}^k}{S_{ij}^{k+1} - S_{ij}^k}. \tag{31}$$

where $(R_{ij}^{k+1}, R_{ij}^k)$ and $(S_{ij}^{k+1}, S_{ij}^k)$ are two pair of adjacent time stamps.

Based on (30), the neighbor clock skew $\hat{\alpha}_i$ can be calculated as below:

$$\hat{\alpha}_i^k = \hat{\alpha}_o^k \cdot \hat{\alpha}_i^k. \tag{32}$$

The reference clock skew $\{\hat{\alpha}_k^k\}$ can be estimated by first substituting $\hat{\alpha}_k^k$ into Eq.(22) to get:

$$\langle \hat{\alpha}_k^k \rangle = \sum_{i \in N} \hat{\alpha}_i^k P(\hat{\alpha}_i^k). \tag{33}$$

where $P(\hat{\alpha}_i^k)$ is calculated using Eq.(13).

### 3.5. Clock offset synchronization

After the skew synchronization is reached, all clocks will run at the same speed. In order to achieve full synchronization, it is also necessary to compensate clock offset.

If there is no delay in the message transmission, clock offset between nodes can be immediately known after time stamp exchange. Unfortunately, in a real wireless network, various delays exist and affect the accuracy of clock offset estimation. Many solutions have been proposed to remove the effects of random delays from the time stamp message. We adopted a Kalman filter based delay filter algorithm as proposed in [19] to get precise clock offset of neighbor nodes. Therefore, node $i$ records the pair time stamp $(S_{ij}^k, R_{ij}^k)$ as shown in Fig.2. The offset of node $j$ can be evaluated as follow:

$$\hat{\beta}_j^k = \hat{\beta}_j^k + (R_{ij}^k - S_{ij}^k - d_{ij}^k). \tag{34}$$

where $d_{ij}^k$ is the random transmission delay which can be dealt by Kalman filter algorithm. To improve the accuracy, MAC-layer time-stamping is used.

Once again, the energy minimization algorithm is employed to update the estimated clock offset, previously defined in Eq. (29), as follows:

$$\hat{\beta}_{ij}^{k+1} = \hat{\beta}_{ij}^k - 2 \times \mu \times (\hat{\beta}_{ij}^k - \langle \hat{\beta}_{ij}^k \rangle) + \sum_{(j \in N_i)} (\hat{\beta}_j^k - \hat{\beta}_i^k). \tag{35}$$

The reference clock offset $\{\hat{\beta}_k^k\}$ can be estimated by first substituting $\hat{\beta}_{ij}^k$ into Eq.(22) to get:

$$\langle \hat{\beta}_k^k \rangle = \sum_{(j \in N_i)} \hat{\beta}_j^k P(\hat{\beta}_i^k). \tag{36}$$

We summarize the aforementioned process in the following algorithm (Mean Field based Time Synchronization).

**Algorithm 1: Mean Field based Time Synchronization (MFTS)**

1. Given the initial conditions for $\alpha_i^0 = 1$ and $\beta_i^0 = 0$, set the broadcast period $T_b$ to each node.
2. Perform iteration until convergence:
3. for the $k^{th}$ iteration do
4. Node $i$ with $i = 1: L$ in parallel
5. Node $i$ broadcasts its current sending time stamp $S_i^k$ and $(\hat{\alpha}_i^{k-1}, S_i^{k-1}, R_{ij}^{k-1})$ to neighboring nodes.
6. Node $i$ receives a time message from its neighbor node $j$. Record the sending time stamp
and the receiving time stamp $R_y^k$.

(7) Calculate the relative skew $\hat{\alpha}_y^k = \frac{R_y^k - R_y^{k-1}}{S_y^k - S_y^{k-1}}$ and the neighbor clock skew $\hat{\alpha}_y^k = \hat{\alpha}_y^k \cdot \hat{\alpha}_y^k$.

Evaluate the transmission delay $d_{ji}^k$ using Kalman filter algorithm and get clock offset $\hat{\beta}_j^k$ of node $j$ using Eq.(34).

(8) Just before the broadcast period is time out, update its virtual reference clock skew and offset as:

$$\hat{\alpha}_{ji}^{k+1} = \hat{\alpha}_i^k - 2 \times \mu \times (\hat{\alpha}_i^k - \hat{\alpha}_j^k) + \sum_{j \in N_i} (\hat{\alpha}_i^k - \hat{\alpha}_j^k) \hat{\beta}_{ji}^{k+1}$$

$$\hat{\beta}_{ji}^{k+1} = \hat{\beta}_j^k - 2 \times \mu \times (\hat{\beta}_j^k - \hat{\beta}_i^k) + \sum_{j \in N_i} (\hat{\beta}_j^k - \hat{\beta}_i^k)$$

(9) Update its local clock as:

$$\hat{\alpha}_i^{k+1} = \hat{\alpha}_i^k$$
$$\hat{\beta}_i^{k+1} = \hat{\beta}_i^k$$

(10) End parallel
(11) End for.

The updating formula just involves simple arithmetic operation and the elementary function operation. So the computational complexity of the algorithm is low.

4. Simulation results

Seeing that Matlab environment is ideal, the entire algorithm is simulated and its efficiency is validated in the MATLAB environment. Meanwhile, the realistic wireless environment can be well modelled in QualNet. So we implemented our algorithm in QualNet network simulator in order to evaluate its performance in realistic environment and compared it with other algorithms. The simulation results of Matlab and QualNet are presented in Sections A and B, respectively.

In order to show the synchronization accuracy, we define the maximum clock error of node $i$ among its whole neighboring nodes after $k$-th synchronization cycles, i.e.,

$$E_{\max}(k) = \max_{i, j \in V} \{C_i(k) - C_j(k)\}$$  \hspace{1cm} (37)

where $C_i(k)$ and $C_j(k)$ denote the clock value of local nodes $i$ and $j$ after $k$-th iteration, respectively. Clearly, clock synchronization is reached if $E_{\max}(k) = 0$.

For today’s microcontrollers provide Real Time Clock (RTC) with microsecond resolution and with an accuracy of about $10^{-5}$ to $10^{-6}$ second, we set $E_{\max}(k) \leq 10^{-6}$ second as convergence condition.

4.1. Matlab simulation results

Simulations are carried out in Matlab 2009b for a network of 100 nodes with the topology of 2-D standard grid as shown in Figure 3.
Every node only communicates with its neighbor in the horizontal and vertical directions as a 4-neighborhood system. At the beginning of simulation, we set initial clock of every node as a random number in the range of $0 \mu s \sim 10^6 \mu s$. In each iteration process (synchronization cycle), every node exchange time stamp information with its neighbor and update local clock simultaneously. The convergence process of clock offset is shown in Fig. 4. In Fig. 4, sub figure (a) is the initial global clock skew status. (b) - (f) are the updated global clock status after 1, 2, 5, 10, and 20 synchronization cycles, respectively. (g) - (p) shows the updated global clock status of every 50 synchronization cycles from 50 cycles to 500 cycles.

The simulation of 5*5, 7*7 nodes are also carried out. Simulation results and comparison are presented in Fig. 5. It clearly shows that our clock synchronization solution takes about 80 cycles to reach the accuracy of microsecond under the simulation example of 5*5 nodes. In addition, simulations take about 81 cycles and 82 cycles to obtain comparable accuracy under the scenarios of 7*7 nodes and 10*10 nodes, and will continue converge. Obviously, the convergence procedure of the proposed solution is insensitive to network scale. Fig 6 shows the maximum and average clock error of scenarios of 5*5, 7*7, 10*10 nodes.

![Figure3](image1.png)

**Figure3** Multi-hop simulation topology in Matlab

![Figure4](image2.png)

**Figure4** The convergence process of clock offset in 500 cycles of Matlab simulation
4.2. QualNet simulation results

QualNet is a widely used simulator to predict the performance of wireless networks especially MANETs. A study done by Hsu et al. validated the ability of QualNet to model realistic wireless environmental effects [25]. So we implemented the clock synchronization in the QualNet network simulator in order to evaluate its performance in realistic environment.

In this section, the proposed protocol is compared by simulation with ATS[11] and GTSP[16] in QualNet. ATS and GTSP are both fully distributed clock synchronization solution. They both use average-based algorithm similarly to our proposal. And ATS also introduced virtual clock as we did.

The simulation configuration is given in Table I.

| Parameter          | Values                                      |
|--------------------|---------------------------------------------|
| Map Size(km*km)    | 0.65*0.65/1.5*1.5/1.5*1.5                  |
| Number of Nodes (N)| 9/16/25/36/49/64/81/100                    |
| Topology           | Grid                                        |
| Simulation Time    | 2000 s                                      |
| Initial offset     | uniform (0, +106) us                        |
| Channel            | 1                                           |
| Antenna Model      | Omnidirectional                             |
| Physical Layer     | 802.11b Radio                               |
| Noise Factor(dB)   | 10                                          |
| MAC Layer          | 802.11                                      |
The convergence process of MFTS is similar to matlab simulation, so we do not demonstrate it anymore. Fig. 7-Fig.10 compare the convergence performance of MFTS, ATS and GTSP in the scenarios of 5*5, 7*7, 9*9, 10*10 nodes in Qualnet simulation. Its y-axis is logarithmic. It clearly shows that our clock synchronization solution convergence slower than ATS in the scenarios of 5*5 and 7*7 nodes. But it takes about 118 cycles to reach the accuracy of microsecond under the simulation example of 9*9 nodes, while it takes 130 cycles under ATS, and 348 cycles under GTSP. In addition, our clock synchronization solution takes 120 cycles and 125 cycles to obtain comparable accuracy under the simulation example of 7*7 nodes and 10*10 nodes. The convergence time is only a very small increase. But the convergence time of ATS and GTSP increase to 178 cycles and 450 cycles respectively in the scenarios of 10*10 nodes.

![Figure 7](image1.png) The convergence performance of scenarios of 5*5 nodes in QualNet

![Figure 8](image2.png) The convergence performance of scenarios of 7*7 nodes in QualNet

![Figure 9](image3.png) The convergence performance of scenarios of 9*9 nodes in QualNet
We also compare the convergence time of MTFS, ATS and GTSP in different network scale as shown in Fig 11. Fig.11 clearly shows, with the increasing scale of the network nodes, the number of iterations required to achieve synchronization will also increase under ATS and GTSP. But MTFS remains stable. As the network scale increases, the superiority of MFTS gradually reflected. The updating formula can be written as $x = Ax$. We have analyzed the updating formula and it turns out that the characteristic roots of $A$ locate in a fixed area. So the convergence is insensitive to network scale. We are working on the mathematical proof.

4.3. Discussion
Since the loss of packet, MFTS convergence time is a little longer in QualNet. The transmission delay can get precisely in simulation of Matlab and QualNet. But in the real wireless environment, the transmission delay is disturbed by many random factors and cannot be measured precisely. So the clock synchronization accuracy of simulation is definitely much better than that in real environment. The following work is to design a more precise transmission delay estimate algorithm to improve the overall performance of clock synchronization in real environment.

5. Conclusion
We investigate the clock synchronization for multi-hop wireless networks in this paper. A fully distributed global clock synchronization solution using mean field is proposed. The main idea is to synchronize local neighbor nodes to achieve global synchronization. It drives all clocks to synchronize to a local virtual reference clock. Considering the spatial Markov property of clock, we model distributed clock as mean field and the virtual reference clock is gotten only by the average effect of one hop neighbor clocks. The solution requires only communicating with one hop neighbor node and is fully distributed and localized. Compared to current algorithms, our solution does not require a reference
clock and only communicates with neighbors. Therefore, it has low overhead and virtues of robustness and scalability. And it can achieve high precise global clock synchronization. Simulation results show that our proposed algorithm achieves better accuracy and faster convergence. Moreover, MFTS can also achieve high precise synchronization with only offset compensation. The results are very satisfactory.

References
[1] O. Simeone, U. Spagnolini, Y. Bar-Ness, and S. H. Strogatz, “Distributed synchronization in wireless networks,” IEEE Signal Process. Mag., 2008, vol. 25, no. 5, pp. 81-97, 2008.
[2] Y. C. Wu, Q. Chaudhari, and E. Serpedin, “Clock synchronization of wireless sensor networks,” IEEE Signal Process. Mag., vol. 28, no. 1, pp. 124-138, 2011.
[3] S. Ganeriwal, R. Kumar, and M. B. Srivastava, “Timing-sync protocol for sensor networks,” presented at the ACM SenSys, 2003.
[4] K. L. Noh, E. Serpedin, and K. Qaraqe, “A new approach for time synchronization in wireless sensor networks: pairwise broadcast synchronization,” IEEE Trans. Wireless Commun., vol. 7, no. 9, pp. 3318-3322, 2008.
[5] J. Du, Y. C. Wu, “Fully distributed clock skew and offset estimation in wireless sensor networks,” presented at the ICASSP, 2013.
[6] J. Elson, L. Girod, and D. Estrin, “Fine-grained network time synchronization using reference broadcasts,” presented at the SIGOPS, 2002.
[7] L. Schenato and F. Fiorentin, “Average timeSynch: a consensus-based protocol for time synchronization in wireless sensor networks,” Automatica, vol. 47, no. 9, pp. 1878–1886, 2011.
[8] M. K. Maggs, S. G. O’Keefe, and D. V. Thiel, “Consensus clock synchronization for wireless sensor networks,” IEEE Sensors J., vol. 12, no. 6, pp. 2269 - 2277, 2012.
[9] J. Wu, L. Jiao, and R. Ding, “Average time synchronization in wireless sensor networks by pairwise messages,” Compu. Commun., vol. 35, no. 2, pp. 221-233, 2012.
[10] J. He, H. Li, J. Chen, et al., “Study of consensus-based time synchronization in wireless sensor networks,” ISA T., vol. 53, no. 2, pp. 347-357, 2014.
[11] X. Gang and K. Shalinee, “Analysis of distributed consensus time synchronization with gaussian delay over wireless sensor networks,” EURASIP J. Wirel. Comm., 2009.
[12] Philipp Sommer, Roger Wattenhofer, “Gradient Clock Synchronization in Wireless Sensor Networks”,
[13] N. Freris, S. Graham, and P. Kumar, “Fundamental limits on synchronizing clocks over networks,” IEEE Trans. Automatic Control, vol. 56, no. 6, pp. 1352–1364, Jun. 2011.
[14] M. Leng, and Y. C. Wu, “Distributed clock synchronization for wireless sensor networks using belief propagation,” IEEE Trans. Signal Process., vol. 59, no. 11, pp. 5404-5414, 2011.
[15] D. Djenouri, N. Merabtine, F. Z. Mekahlia, and M. Doudou, “Fast distributed multi-hop relative time synchronization protocol and estimators for wireless sensor networks,” Ad Hoc Netw., vol. 11, no. 8, pp. 2329–2344, 2013.
[16] S. Z. Li, “Markov Random field modeling in image analysis,” in Heidelberg: Springer, 3rd ed., 2009, pp. 5.
[17] Baierlein R. Thermal physics. Britain: Cambridge University Press, 1999
[18] Chandler D. Introduction to modern statistical mechanics. Britain: Oxford University Press, 1987.
[19] J. He, P. Cheng, L. Shi, and J. Chen, “Time synchronization in wsns: a maximum value based consensus approach,” presented at the CDC, 2011.
[20] J. Hsu, S. Bhatia, M. Takai, and R. Bagrodia, “Performance of mobile ad hoc networking routing protocols in realistic scenarios,” presented at the MILCOM, 2003.
[21] Wu Chen, Jianhua Sun, Lu Zhang, Xiang Liu and Liang Hong, “An implementation of IEEE 1588 protocol for IEEE 802.11 WLAN”.

14