Coexisting Parallelogram Method to Handle Jump Point on Hough Transform-based Clock Skew Measurement

Nyoman Putra Sastra, Komang Oka Saputra, and Dewa Made Wiharta

Original scientific article

***Abstract*—In this paper, we improve the robustness of the Hough transform-based clock skew measurement on the occurrence of a jump point. The current Hough transform-based skew method uses angle (θ), thickness (ω), and region (β), to create a parallelogram that covers the densest part of an offset-set. However, the assumption that all offsets are considered to line up roughly in only one direction restricts the ability of the current method when handling an offset-set in which its densest part is located separately, the jump point condition. By acquiring the parallelogram from coexisting angle-region tuples at the beginning and the ending parts of the offset-set, we completed the ability of the Hough transform-based method to handle the jump point. When handling the jump point problem, the proposed coexisting parallelogram method could reach 0.35 ppm accuracy compared with tens ppm by the current methods.**

***Index Terms*—Clock skew, Hough transform, jump point, coexisting parallelogram.**

I. INTRODUCTION

CLOCK of a device does not only provide time information or timestamps. Other parameters of the device's clock can be explored for other purposes. For example, clock skew, or the clocking rate difference between two digital clocks, is a phenomenon that causes timestamps between one device to be different comparing to a reference clock. This phenomenon occurs due to the low quality of the clock or the age of the clock on a device that makes the clock ticking slower than before. Clock skew has become a significant issue on delay measurement [4], [19], [21], [34], where when we measuring delay through sending packets between transmitter and receiver, the timestamps are not pure as they are containing clock skew where therefore the effect of clock skew must be removed. However, as the value of clock skew is stable over time and they are unique for each device, clock skew can be used for device identification [14], [17], [23], [24]. Here, clock skew is used as an ID for each device connected to the same server. Every time a device is connected to a server, its clock skew is measured, and then it is compared with the saved clock skew. When the skews are similar or they are both still at a certain threshold, the device is called a valid device. Other area exploring clock skew are distributed anonymity architecture [20], [27], non-cryptography security [28], [35] and wireless sensor network [32], [33]. All areas used clock skew depends on the accurate value of the measured clock skew. For this, many scholars have proposed methods to estimate its value on devices with digital clocks [2], [4], [11], [16], [19], [21]. These methods initiate the estimation through a measurer that collects a target device's timestamps over a network connection [30]. When measuring the skew, the timestamps collection is processed in a scatter diagram, like in Fig. 1 for instance, a sample from an experiment which would be introduced later in Section 4. The *x–axis* of the scatter diagram is the measurer's timestamp; the *y–axis*, meanwhile, is offset calculated by subtracting the device's timestamp from the measurer's timestamp [14], [30]. Basically, the skew is the slope of the offset-set on the scatter diagram. In the details when finding the slope, each method has its own characteristic. Linear regression [19], for example, would only be appropriate when the offsets gather to form a straight line near the bottom part of the scatter diagram [30]. Since the collection process of the timestamps involves delays between the device and the measurer [30], only a stable environment, a wired communication, for example, can produce that kind of data [30]. When the timestamps collection[1] process is implemented in a poorer situation, such as a wireless network, many offsets are separated up from the densest part in the lower region of the scatter diagram (outliers) [14]. With many outliers exist, the slope created by the linear regression would deviate to follow the outliers, and leads to an inaccurate clock skew [14], [19], [21]. Picking only offsets in the lower region of the scatter diagram (minimum offsets) to avoid the outliers would be a very promising solution. Paxson [21], Aoki [2], and Huang et al. [14] proposed this kind of solution with their own scheme. Paxson [21] combines the minimum offsets

Manuscript received January 28, 2021; revised June 29, 2021. Date of publication October 4, 2021. Date of current version October 4, 2021. The associate editor prof. Nikola Rožić has been coordinating the review of this manuscript and approved it for publication.

Authors are with the Electrical Engineering Study Program, Faculty of Engineering, Universitas Udayana, Bali, Indonesia (e-mails: {putra.sastra, okasaputra, wiharta}@unud.ac.id).

Digital Object Identifier (DOI): 10.24138/jcomss-2021-0028

method with a median technique, Aoki [2], meanwhile, uses several windows with similar size, and Huang et al. [14] proposed their sliding windows method. Later, Moon et al. [19] with their linear programming algorithm (LPA) shows a robust method that is unaffected by outliers, just by drawing a line below all the offsets. However, when the offsets at the lower region of the scatter diagram become unstable caused by the presence of the low-outliers, i.e., offsets below the crowded offsets [30], the estimation results by the aforementioned methods for short-term measurements become inaccurate. The Hough transform-based clock skew measurement [30] has succeeded to provide a new method for estimating the skew accurately both on the classic case of the normal lower bound, and for the unstable lower bound as well.

Unlike the classic methods, the Hough transform-based skew method maps the offsets in the scatter diagram into image points [30]. Through these image points, a Hough transform-like voting scheme is then applied to find a parallelogram-like region that bound the densest part of all offsets [30]. When the parallelogram-like region represented by angle θ, thickness ω, and region β is obtained, the clock skew is next produced from the slope of all offsets inside the parallelogram [30].

Another robustness criterion of a clock skew measurement is it should be able to handle as many as possible situations, that occur when a measurer collecting a device's timestamps [24]. Another study related to skew by Huang et al. [14], shows conditions when a connected device changes its network connection to a measurer can cause what is called the jump point. The first type of jump point, the time gap that is indicated by a period with blank packets [14, Fig. 9], will not give any problem for the skew measurement by any existing methods used. The second type [14, Fig. 7], conversely, a condition of offsets that starting to increase or decrease sharply as an effect of different delays on distinct transmission medium will cause inaccurate estimation by the current methods. Since the current version of the Hough trans-form based skew method follows the stable over-time concept, i.e., all offsets are considered to line up roughly in one direction [30], a weird condition such as the jump point where more than one group of offsets occur will be detected as one like in Fig. 2 a sample which would be analyzed later in Section 4. As a result, the Hough transform-based method will produce a large parallelogram-like region that covers all the offsets, including the jump point, and finally produce an inaccurate skew result. As far as we are concerned there is a solution to handle this jump point problem, a method called DROML presented in [31]. DROML created a dynamic region to cover the densest part of the offsets collection, which makes the calculated skew is spared from the effect of the jump point. However, DROML find the densest part of the offsets from the beginning of the collection to the end. We found that the jump point condition can also be detected through small pieces of a parallelogram that coexist in the beginning and ending parts of an offset-set that can be lengthened to cover all offsets.

Our contribution in this paper is to build a new technique by obtaining coexisting parallelograms in two parts of offsets, the beginning and the ending. Since the clock skew has been revealed to be stable over time [17], [19], [21], a small piece of parallelograms that coexist in the beginning and ending parts can be lengthened to cover all offsets. When an offset-set contains the jump point, however, no coexisting parallelograms will be produced. Hence, we omit the effect of the jump point by obtaining the skew from skews of some subsets of the offset-set which are no longer contain the jump point. Our evaluation results show that compared with LPA [20], Aoki's method [2], and the Hough transform based method [31], only the proposed method can handle the problem accurately.

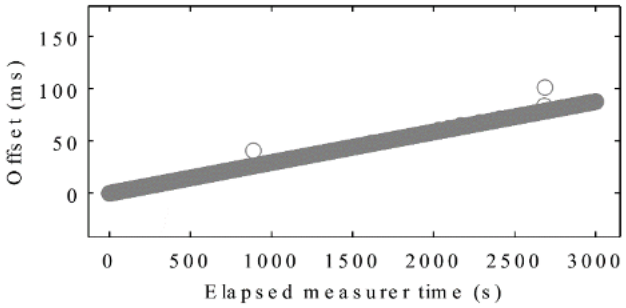

Fig. 1. An offset-set between a measurer and a device that are connected through a wired network.

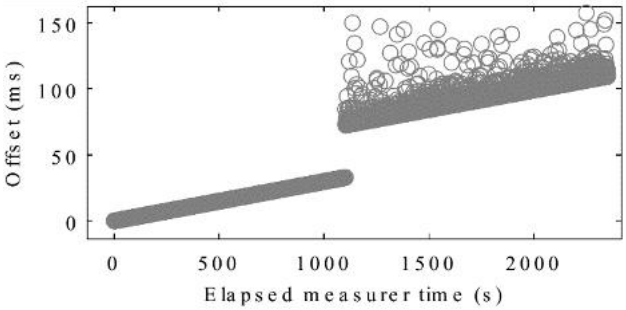

Fig. 2. An offset-set contains a jump point condition

The following section describes in detail the voting scheme we used in this paper. This voting scheme is similar to [15, Algorithm 1]. Next in Section III, the coexisting parallelogram method for handling the jump point problem is explained. Evaluation results are shown in Section IV. Finally, we summarize our work in Section V.

II. VOTING METHOD

Summarized from [30], in obtaining the densest part of an offset-set, the Hough transform-based clock skew method maps all the offsets into image points. From all the offsets that are sorted based on the time of the measurer, the origin of the image points is chosen at (t_1, o_{min}), where t_1 stands for the first measurer's timestamp, and omin is the minimum offsets of the offset-set. As illustrated in Fig. 3, for a given angle θ, a line of $\rho=x\cos\theta+y\sin\theta$ is drawn through (t_1, o_{min}). Afterward, many regions with an identical thickness ω indexed from β_1 to β_n are then created. Here β is a value when ρ of the line on the origin is rounded down to an integer multiple of ω. To mark the position of all points, the Hough transform-based voting method store the voting information of each point: the angle, the thickness, and the corresponding region index, in the form

of *Votes(θ, ω, β)*. The more points located in the same angle-region tuple, the higher the vote's number of that tuple. Since the purpose is to cover the densest part of the offset-set, an angle-region tuple with the highest vote's number that exceeds a certain threshold is then used to create the parallelogram.

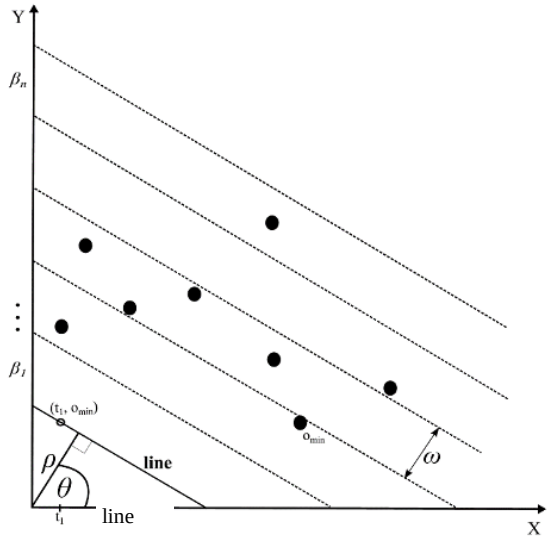

Fig. 3. Relationship between offsets, *ρ, θ, β*, and *ω* on the Hough transform based voting method

A. Dynamic-Region Method

To obtain the parallelogram, a voting method must be implemented, here we named the voting method as a dynamic-region voting method. This method is based on [31, Algorithm 1]. At first, we refuse of using the rigid pre-defined thickness that divides the image points into several fixed-size regions. On the contrary, we define regions from *ρ* of each point that is added by some distance d. Fig. 4 illustrates how this method works. For a given angle *θ*, each point has its own line of $\rho=x*cos\theta+y*sin\theta$. The first region, noted as β_1, is from ρ_1 to ρ_{1+d}, the second, β_2, is from ρ_2 to ρ_{2+d}, and so on until the last one, β_{n-1}, is from ρ_{n-1} to ρ_{n-1+d}. Even the size of all regions is similar to follow the added distance d, the offsets covered by each region are dynamic to follow the distribution of the offsets near a certain region. We can equate *d* with *ω* in the existing voting method. However, while *ω* creates many fixed-size regions that are located rigidly on the image points, combination of d with ρ on each point leads us systematically into regions that congregate near the densest part sought. With regions near the densest part that differ so small, it is like tracking the real character of each point with a small step whether a point is a portion of the densest part or not. Through this method, a close outlier or low-outlier to the densest part can be avoided to be a part of the densest part. As a result, an angle-region tuple with the highest vote's number represents the real densest part without any unwanted outliers or low outliers inside it. Since the real thickness of the densest part of an offset-set is not fixed, the effect of the jitter [30], we can increase the distance d to fulfill the need of a wider region caused by a wider densest part. Even the regions become larger caused by the escalation of *d*, they still converge near the densest part, following the distribution of the offsets. Hence, the accuracy can be maintained.

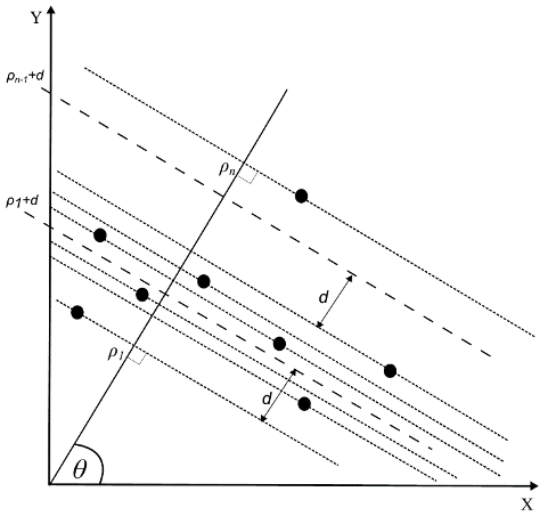

Fig. 4. Relationship between offsets, *ρ, θ*, and *d* on the dynamic-region voting method

B. Pseudo Code of The Dynamic Region Method

To implement the dynamic-region voting method described above, we modify Function *OffsetVote()* in [30, Algorithm 1] into Function *DynamicRegion()* which is detailed Algorithm 1. Since we no longer use a pre-defined thickness when creating regions, the thickness *ω*, its lower bound ω_{min}, and the thickness increment ωinc in [30, Algorithm 1] are no longer used. Meanwhile, we keep the other variables: *S*, the set of all points; θ_{min} and θ_{max}, the boundary of the angles; *p*, the angle step-size; and *N*, the threshold. Related to the dynamic region, we introduce two new variables: *d* stands for a distance from a certain *ρ* for creating the regions, and d_{inc} which is used to increase the distance when necessary.

For each the scanned angle in the first for loop, the second for loop of Algorithm 1 creates a line of $\rho=x*cos\theta+y*sin\theta$ on each point (*x*, *y*), which results in an array of many *ρ* that is ordered from its minimum to maximum values. In the next for loop, the regions are created by adding d to each *ρ*. And then, we count how many points are located inside each region by counting the number of *ρ* inside it, and store the value in *Votes(β, θ)*. Finally, inside the other for loop near the end of the algorithm, an angle-region tuple with the highest vote's number that exceeds the threshold *N* is used to form the parallelogram. If there are no candidate regions found, we increase the distance (*d = d + dinc*), and start a new round of voting.

III. Coexisting Parallelogram Method

A. Coexisting Parallelogram Method

Our solution to handle the jump point problem is as follows. First of all, since the skew has been revealed to be constant from the beginning to the end of the measurement [12], [13], [14], small pieces the parallelogram that coexist in the beginning and ending parts of an offset-set can be lengthened to cover all offsets. To explain how it works, observe a set of offsets (*Measurer time*, *Offset*) in Fig. 5: (t_1, o_1), ..., (t_n, o_n). Let *part*

Algorithm 1 Dynamic-region voting method

```
function DYNAMICREGION(S, θ_min, θ_max, p, d, d_inc, N)
    Votes = 0
    L = null
    while L == null do
        for θ = θ_min; θ ≤ θ_max; θ = θ + p do
            for i = Min(S); i ≤ Max(S); i++ do
                ρ_i = x_i * cos θ + y_i * sin θ
            end for
            for j = 1; j < i; j++ do
                (β_j, θ) = ρ_j to ρ_j + d
                for k = j; k < i; k++ do
                    if ρ_j ≤ ρ_k and ρ_k ≤ ρ_j + d then
                        Votes(β_j, θ) = Votes(β_j, θ) + 1
                    end if
                end for
            end for
        end for
        for all (β, θ) do
            if Votes(β, θ) ≥ N then
                if L == null or Votes(β, θ) > L.Votes
then
                    L = (β, d, θ, Votes(β, θ))
                end if
            end if
        end for
        if L == null then
            d = d + d_inc
        end if
    end while
    return L
end function
```

denotes the number of offsets for the beginning and ending parts of all offsets. Then, the beginning part that is notated by S_{beg} contains offsets from (t_1, o_1) to (t_{part}, o_{part}); and the ending part that is notated by S_{en} contains offsets from (t_{n-part}, o_{n-part}) to (t_n, o_n). After the voting process on both parts is done, the parallelogram-like region for the whole collection is a prolongation of angle-region tuples that occur coexisting in both parts, e.g., θ_1 and β_1 in Fig. 5.

To implement the method above, we developed Algorithm 2, Function *CoexistRegion()*. In this algorithm, Function *DynamicRegion()* is called twice to process S_{beg} and S_{en}. The results of both processes are stored in L_1 and L_2, for S_{beg} and S_{en} respectively. The while loop in Algorithm 2 shows that the process will be terminated only when similar angle-region tuples are found in both parts. Otherwise, both processes are repeated with a higher initial distance value, which is the smallest last used distance on both processes. To prevent the region grows too large, we define d_{max} as the upper-bound limit of the region size.

With the region size that is bounded into a maximum size, usage of only the beginning and ending parts has an advanced benefit that there will be no coexisting angle-region tuples when an offset-set contains a jump point because the beginning and ending parts are obviously in different regions. To obtain the solution of this problem, we developed Algorithm 3 where it still uses a three-stage process with major improvements comparing with the original three-stage process in [30, Algorithm 2].

Since the second and third stages only aim to pursue higher precisions from the result of the first stage [30], we locate all processes of handling the jump point in the first stage. Hence, the next two stages will only process the output of the first stage without worrying whether the offset-set contains a jump point or not. Other than S_{beg}, S_{en}, and the *part* which are introduced before, Algorithm 3 uses S_{BEG}, S_{EN}, and L_{temp} to store all the corresponding S_{beg}, S_{en}, and L which are resulted by the first-stage process. Later, all values in S_{BEG}, S_{EN}, and L_{temp} are processed in the second and third stages. At the end of the method, to store the results of the whole process, we defined a variable L_{final}.

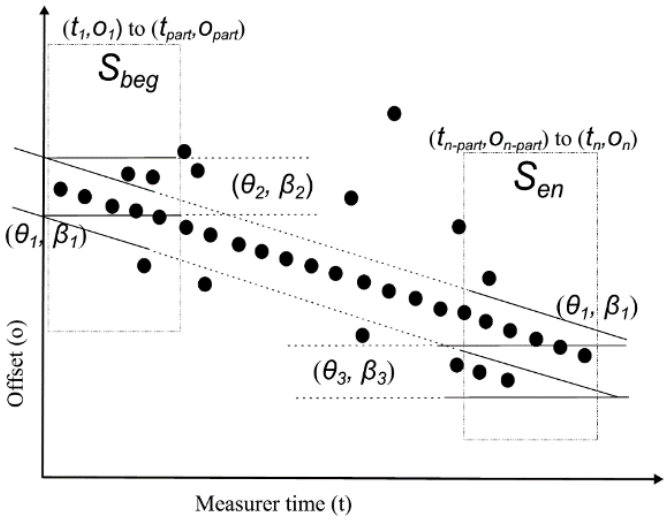

Fig. 5. Usage of coexisting parallelograms that occur at the beginning and ending parts of an offset-set on the Hough transform based clock skew measurement.

In Algorithm 3, after S_{beg} and S_{en} are created from S, the first-stage process is started by calling Function *CoexistRegion()*. If this process produces results, that means the offsets collection is normal, and the second and third stages will only run one time to process S_{BEG}, S_{END}, and L_{temp} in the for loop near the end of the algorithm. Otherwise, the offset-set might contain a jump point, and the first-stage process must preprocess the offset-set to produce several subsets of the offset-set which no longer contain the jump point. To do that, two variables are used: $S_{succeed}$ denotes the number of offsets that have already been processed, and S_{failed} which stands for the unprocessed part of the offset-set. At this point, $S_{succeed}$ is zero and S_{failed} is the whole

Algorithm 2 Coexisting dynamic-region method

```
function COEXISTREGION(S_beg, S_en, θ_1, θ_2, p, d, d_inc,
d_max, N)
    L, L_1, L_2 = null
    while (L_1.β, L_1.d, L_1.θ) ≠ (L_2.β, L_2.d, L_2.θ) or
d ≤ d_max do
        L_1 = DynamicRegion(S_beg, θ_1, θ_2, p, d, d_inc, N)
        L_2 = DynamicRegion(S_en, θ_1, θ_2, p, d, d_inc, N)
        if (L_1.β, L_1.d, L_1.θ) ≠ (L_2.β, L_2.d, L_2.θ) then
            if L_1.d ≤ L_2.d then
                d = L_1.d
            else
                d = L_2.d
            end if
        end if
    end while
    L = L_1
    return L
end function
```

collection. If the jump point is located in the middle of S_{failed}, we only need to divide S_{failed} into two parts, and use both parts to produce a result. However, most likely the location is not in the middle of S_{failed} because it depends on the time when the device changes its network connection. Inside the while loop, is a mechanism to handle the uncertain location of the jump point. Here *SS* stands for a subpart of the processed S_{failed}, two parts for each S_{failed}. For each *SS*, S_{beg} and S_{en} are created, and then Function CoexistRegion() is run. When no results from Function CoexistRegion(), *SS* is then marked as a new S_{failed}. Otherwise, S_{beg}, S_{en}, and the result *L*, are stored in S_{BEG}, S_{EN}, and L_{temp} respectively. To store the information of a successful process, $S_{succeed}$ is then added by the number of offsets in *SS*. Later, after the while loop finish which is indicated by all offsets are already processed ($S_{succeed} = S.length$), all S_{BEG}, S_{EN}, and L_{temp} are processed in the for loop of the second and third stages process. It is important to be noted that only S_{failed} which is longer than twice of *part* that will be processed. Otherwise, we leave that S_{failed} unprocessed by adding it to $S_{succeed}$ because *SS* from S_{failed} in this condition is already shorter than *part*.

For a normal offset-set, Algorithm 3 will produce only one tuple of $L_{final}.\theta$, $L_{final}.DR$, and $L_{final}.d$. When the offset-set contains a jump point, however, L_{final} stores many tuples. Each of the tuple will result in skew, and finally a median value of all resulted skews is then used as the final skew.

Algorithm 3 Three-stage process

Require: S, d, d_{inc}, d_{max}, N, $part$
  $L_{temp}, L_{final}, S_{BEG}, S_{EN} = null$
  $S_{beg} = S(1)$ to $S(part)$
  $S_{en} = S(S.length - part)$ to $S(S.length)$
  $p = 10^{-5}$
  θ_{min} = (π / 2) - $750 * 10^{-6}$
  θ_{max} = (π / 2) + $750 * 10^{-6}$
  CoexistRegion(S_{beg}, S_{en}, θ_{min}, θ_{max}, p, d, d_{inc}, d_{max}, N)
  if $L \neq null$ **then**
    $i = 1$
    $S_{BEG}(1) = S_{beg}$
    $S_{EN}(1) = S_{en}$
    $L_{temp}(1) = L$
  else
    $j, k, l, m = 1$
    $S_{succeed}$ = 0
    $S_{failed(j)} = S$
    while $S_{succeed} \neq S.length$ **do**
      for $k = 1; k \leq l; k++$ **do**
        for $m = 1; m \leq 2; m++$ **do**
          if $S_{failed(k)}.length/2 > part$ **then**
            $SS = S_{failed(k)}((m-1) * (S_{failed(k)}.length/2) + 1)$ to $S_{failed(k)}(m * S_{failed(k)}.length/2)$
            $S_{beg} = SS(1)$ to $SS(part)$
            $S_{en} = SS(SS.length - part)$ to $SS(SS.length)$
            CoexistRegion(S_{beg}, S_{en}, θ_{min}, θ_{max}, p, d, d_{inc}, d_{max}, N)
            if $L \neq null$ **then**
              $S_{BEG}(i) = S_{beg}$
              $S_{EN}(i) = S_{en}$
              $L_{temp}(i) = L$
              $S_{succeed} = S_{succeed} + SS.length$
  for $n = 1; n \leq i; n++$ **do**
    $\theta_{min} = L_{temp}(n).\theta - 5 * 10^{-6}$
    $\theta_{max} = L_{temp}(n).\theta + 5 * 10^{-6}$
    $p = p$ / 10
    CoexistRegion($S_{BEG}(n)$, $S_{EN}(n)$ θ_{min}, θ_{max}, p, d, d_{inc}, d_{max}, N)
    $\theta_{min} = L.\theta - 5 * 10^{-7}$
    $\theta_{max} = L.\theta + 5 * 10^{-7}$
    $p = p$ / 10
    CoexistRegion($S_{BEG}(n)$, $S_{EN}(n)$, θ_{min}, θ_{max}, p, d, d_{inc}, d_{max}, N)
    $L_{final}(n) = L$
  end for
  return L_{final}

IV. Evaluation Results

For the coexisting dynamic-region method on handling the jump point, the data set was obtained by connecting a client device (ASUS A46C Notebook with UBUNTU 14.04 as the operating system) to a measurer (IBM server with an OS of UBUNTU 12.04) on two different networks. First, we connected the client to the measurer through a wired LAN. After several packets were transmitted by the client, the wired connection was disconnected, and then a multi-hops wireless connection was started automatically. For a purpose of comparison with other methods, we also derived a normal collection of the aforementioned setting, and use it as a reference. A relatively close clock skew when comparing the result of the jump point with the reference is an indication of the success of a measurement method. To show the effect of the jump point as well as to confirm the performance of our proposed method, three methods were exploited: LPA [8], Aoki's method [10], and the current version of the Hough transform-based method [30].

On each evaluation, the parameters used on Algorithms 1, 2, and 3 are: d = 500 μs, d_{inc} = 100 μs, d_{max} = 2000 µs, N = 50%, and *part* = 500 offsets.

First of all, Fig. 1 shows the offset-set of the reference skew. This offset-set was derived from a wired peer-to-peer communication between the aforementioned client and measurer. When estimating the skew of this collection, all four methods: LPA, Aoki's method, the original Hough transform method, and the proposed method produced almost similar skews of 29.23 ppm, 29.25 ppm, 29.21 ppm, and 29.2 ppm, respectively. Let use a median value of these four as the skew reference, 29.25 ppm.

To produce the final skew when measuring the offset-set contains a jump point in Fig. 2, eight steps were needed by the proposed method as detailed in Table 1. From this table we can see that when all the offsets (1–4693) were processed, step–1 failed to produce results due to the absence of any coexisted angle-region tuple in the beginning and ending parts of the 4693 offsets. After the offset-set was divided into (1–2347), step-2, and (2348–4693), step-3, step-3 succeeded to produce a clock skew of 29.8 ppm, but step-2 failed. The division process for the failed step is then continued, and completed after eight steps. From all the processes, three clock skews were produced: 29.8 ppm on step-3, 29.4 ppm on step-4, and another 29.4 ppm on step-6. As the final skew, a median value of 29.6 ppm from these three clock skews is then used.

TABLE I. RESULTS OF THE PROPOSED METHOD WHEN ESTIMATING AN OFFSET-SET CONTAINS A JUMP POINT

Step	Offset	First-stage	Second-stage	Third-stage	Clock skew
1	1–4693	$L = null$	No process	No process	-
2	1–2347	$L = null$	No process	No process	-
3	2348–4693	θ = 1.570826 rad, $size$ = 700 μs	θ = 1.5708259 rad, $size$ = 600 μs	θ = 1.5708264 rad, $size$ = 600 μs	29.8 ppm
4	1–1174	θ = 1.570826 rad, $size$ = 500 μs	θ = 1.5708259 rad, $size$ = 500 μs	θ = 1.5708260 rad, $size$ = 500 μs	29.4 ppm
5	1175–2347	$L = null$	No process	No process	-
6	1175–1761	θ = 1.570826 rad, $size$ = 500 μs	θ = 1.5708259 rad, $size$ = 500 μs	θ = 1.5708260 rad, $size$ = 500 μs	29.4 ppm
7	1762–2347	$L = null$	No process	No process	-
8	1762–2347	S_{failed}.length/2 $<part$	No process	No process	-
Median value of all skews					29.6 ppm

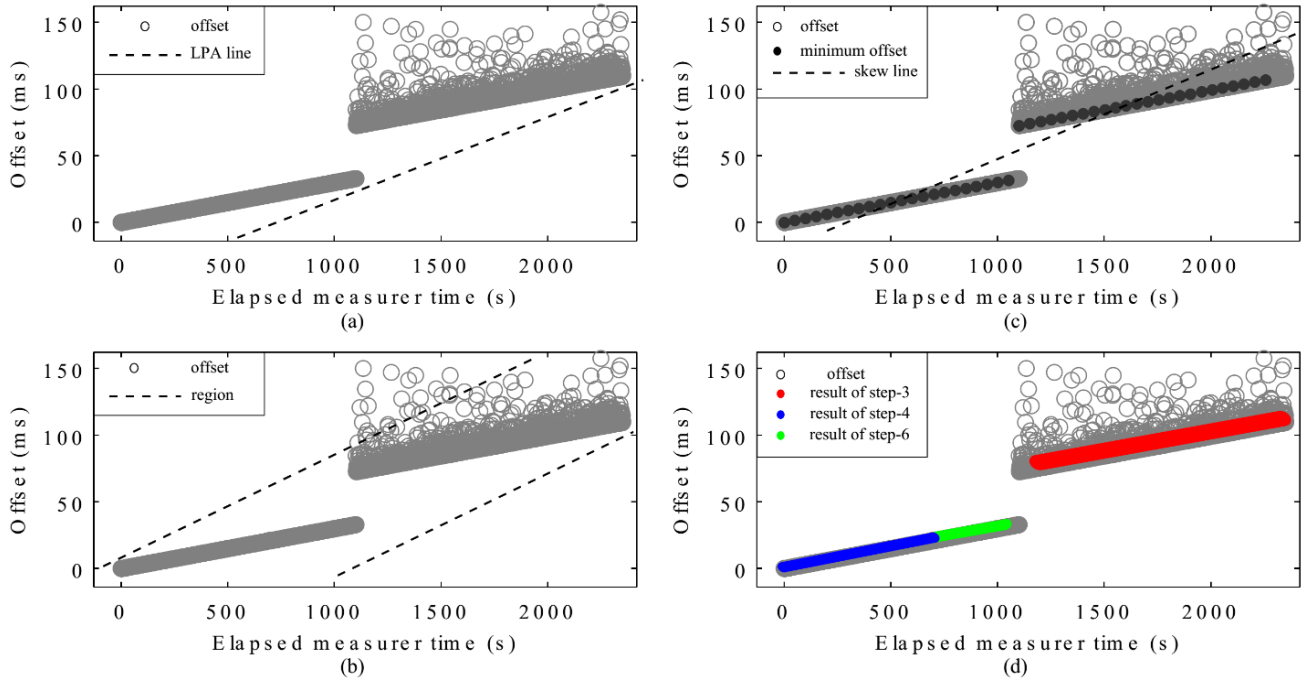

Fig. 6. Estimation results on a jump point condition by four different methods. (a) LPA. (b) Hough Transform. (c) Aoki's method. (d) Proposed method.

To show the effect of the jump point, Fig. 6 illustrates the results of the four methods when estimating the skew of the offset-set in Fig. 3. We can observe from Fig. 5a and Fig. 5c that the slopes, i.e., the skew line of LPA and Aoki's method are inaccurate due to the inability of detecting the presence of the jump point. For the original Hough transform method in Fig 5b, meanwhile, they just abandon the jump point, the resulted parallelogram becomes too big, which results in an inaccurate skew as well. On the other hand, our proposed method succeeded to produce skews that do not contain the jump point anymore. Three densest parts as produced in step-3, step-4, and step-6 are depicted with different color in Fig. 5d. When compared the result of the proposed method (29.6 ppm) with the results of LPA (61.29 ppm), the original Hough transform method (70.5 ppm), and Aoki's method (55.49 ppm), the proposed method obtained the closest clock skew to the clock skew reference (29.25 ppm), only differ 0.35 ppm. The results of this evaluation indicate that the proposed coexisted dynamic-region method has succeeded to be implemented to handle the jump point.

V. DISCUSSION AND FUTURE WORK

To show more the robustness of the proposed method, here we presented the evaluation results on three other datasets containing jump points. The three datasets are from different client devices: a Macbook Pro, an ASUS ROG, and an IBM Thinkpad, connected to the same existing server. The scenario is similar where the connections are changed from using a LAN into WLAN at a random time point. We named the three datasets as Dataset-2, Dataset-3, and Dataset-4. We have also measured the reference skews from the three devices connected to the server, where the skews are 52.3 ppm, 34.8 ppm, and 20.1 ppm respectively.

Table 2 shows the measurement results of the proposed method into the three datasets. Dataset-2 and Dataset 3 have similar patterns where the proposed method requires 8 steps to finish the skew measurement. Meanwhile the proposed method only uses two steps to measure Dataset-4. These facts inform us that the jump point of Dataset-4 occurs in the middle of the dataset, while Dataset-2 and Dataset-3 near the beginning or the end of the dataset.

TABLE II. RESULTS OF THE PROPOSED METHOD ON THREE DIFFERENT DEVICES CONNECTED TO EXISTING SERVER

Step	Dataset-2		Dataset-3		Dataset-4	
	Range of offsets	Result	Range of offsets	Result	Range of offsets	Result
1	1--5004	No Tuple	1--4908	No Tuple	1--3998	No Tuple
2	1--2502	No Tuple	1--2453	Skew = 34.7 ppm	1--1999	Skew = 19.9 ppm
3	2503--5004	Skew = 52.2 ppm	2454--4980	No Tuple	2000--3998	Skew = 20.1 ppm
4	1--1251	Skew = 51.9 ppm	2454--3717	No Tuple		
5	1252--2502	No Tuple	3718--4980	Skew = 34.9 ppm		
6	1252--1878	Skew = 52.1 ppm	2454--3086	Skew = 34.7 ppm		
7	1879--2502	No Tuple	3089--3716	No Tuple		
8	1879--2502	S/2 < *Part*	3089--3716	S/2 < *Part*		

The position of the jump point of the dataset can cause the proposed method to work longer or shorter. Jump point near the middle of the dataset takes fewer steps than those near the beginning or end, as the method can cover all the datasets quicker.

The number of *Part* also takes an importance rule as we have to choose an adequate value for *Part.* The bigger *Part* value ensure the more accurate skew, when the dataset is already in one densest part. However, when *Part* is too big, the proposed method can miss a jump point, result in no skew can be measured. Previous work shows that 500 offsets are enough to measure skew accurately, and when we compare to our scenario, the jump point of changing connection will not occur before 500 offsets are transferred by the client to the server.

From Table 2, we can see that the median skew of Dataset-2 is 52.1 ppm, Dataset-3 is 34.7 ppm, and Dataset-4 is 20 ppm. These median skews are close to the reference skews of each dataset; 52.3 ppm, 34.8 ppm, and 20.1 ppm. The differences are no more than 0.2 ppm. From these results, we can conclude that the proposed method has succeeded to measure the skews of those three datasets containing jump point accurately.

A Jump point in a dataset can occur by network communication change or by NTP synchronization. These two factors rarely occur when we connected to a site. NTP synch might happen once a day. Change network connection between cable and wireless can occur more, but not also frequently occur, and the gap between timestamps on cable and wireless is quite high where the proposed method can differentiate the densest part of that jump from the original. The next challenge for the work on clock skew is the condition when client travel from one Access Point (AP) to another one in wireless connection only. This condition can occur more frequently as a lot of APs is available in one area. The other issue is the speed between APs is different, which makes the time to send data will differ as well, cause the densest part of the data will not be in one part or there will be a jump point. However, the jump will not be as high as those data caused by a change connection between wired and wireless. To detect more jump point in a dataset as well as to detect small size of the jump in a dataset is the future work that interesting and challenging to solve.

VI. CONCLUSION

We have proposed a new coexisting dynamic-region method to improve the robustness of the Hough transform-based clock skew measurement to find a coexisting parallelogram at the beginning and ending parts of the offset-set to handle the jump point condition accurately. Experiments results show that the proposed coexisting parallelogram method could successfully measure the jump point data with 0.35 ppm accuracy compared with tens ppm by the current methods. When the experiment expanding to three other devices connected to a similar server, the proposed method has succeeded to maintain its robustness by measuring the skews with an accuracy of 0.2 ppm comparing to the reference skews.

REFERENCES

[1] T. E. Abrudan, A. Haghparast, and V. Koivunen, "Time synchronization and ranging in OFDM systems using time-reversal," *IEEE Trans. Instrum. Meas.*, vol. 62, no. 12, pp. 3276–3290, Dec. 2013. DOI: 10.1109/TIM.2013.2272840.

[2] M. Aoki, E. Oki, and R. Rojas-Cessa, "Measurement scheme for one-way delay variation with detection and removal of clock skew," *ETRI J.*, vol. 32, no. 6, pp. 854–862, Dec. 2010. DOI: 10.1109/HPSR.2010.5580276

[3] C. Arackaparambil and S. Bratus, "On the reliability of wireless fingerprinting using clock skews," Presented at the 3rd ACM Conf. Wireless Network Security, 2010. [Online]. Available: https://dl.acm.org/doi/10.1145/1741866.1741894

[4] J. Bi, Q. Wu, and Z. Li, "On estimating clock skew for one-way measurements," *Comput. Commun.*, vol. 29, no. 8, pp. 1213–1225, May 2006. DOI: 10.1016/j.comcom.2005.07.023

[5] Z.-H. Chen, A. W. Y. Su, and M.-T. Sun, "Resource-efficient FPGA architecture and implementation of Hough Transform," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 8, pp. 1419–1428, Aug. 2012. DOI: 10.1109/TVLSI.2011.2160002

[6] Y. Chae, L. C. DiPippo, and Y. L. Sun, "Trust Management for Defending On-Off Attacks," *IEEE Trans. Parallel Distrib. Syst*., vol. 26, no. 4, pp. 1178–1191, 2015. DOI: 10.1109/TPDS.2014.2317719
[7] M. Cristea and B. Groza, "Fingerprinting smartphones remotely via ICMP timestamps," *IEEE Commun. Lett*., vol. 17, no. 6, pp. 1081–1083, Jun. 2013.
[8] K. S. Yildrim and A. Kantarci, "Time Synchronization Based on Slow-Flooding in Wireless Sensor Networks," *IEEE Trans. Parallel Distrib. Syst*., vol. 25, no. 1, pp. 244–253, 2014. DOI: 10.1109/TPDS.2013.40
[9] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, Jan. 1972. DOI: 10.1145/361237.361242
[10] P. Hart, "How the Hough transform was invented," *IEEE Signal Process. Mag*., vol. 26, no. 6, pp. 18–22, 2009. DOI: 10.1109/MSP.2009.934181
[11] D.-J. Huang, W.-C. Teng, C.-Y. Wang, H.-Y. Huang, and J. M. Hellerstein, "Clock skew-based node identification in wireless sensor networks," Presented at the IEEE Global Telecommun. Conf. (GLOBECOM), 2008. [Online]. Available: https://ieeexplore.ieee.org/document/4698138
[12] D.-J. Huang and W.-C. Teng, "A defense against clock skew replication attacks in wireless sensor networks," *J. Network and Comput. Applicat.*, vol. 39, pp. 26–37, Mar. 2014. DOI: 10.1016/j.jnca.2013.04.003
[13] D. Huang, W. Teng, and K. Yang, "Secured flooding time synchronization protocol with moderator," *Int. J. Commun. Syst*., vol. 26, no. 9, pp. 1092–1115, 2013. DOI: https://doi.org/10.1002/dac.2614
[14] D.-J. Huang, K.-T. Yang, C.-C. Ni, W.-C. Teng, T.-R. Hsiang, and Y.-J. Lee, "Clock skew-based client device identification in cloud environments," Presented at the 26th IEEE Int. Conf. Advanced Inform. Networking and Applicat. (AINA), Mar. 2012. [Online]. Available: https://ieeexplore.ieee.org/document/6184915
[15] S. Jana and S. Kasera, "On fast and accurate detection of unauthorized wireless access points using clock skews," *IEEE Trans. Mobile Comput.*, vol. 9, no. 3, pp. 449–462, Mar. 2010. DOI: 10.1109/TMC.2009.145
[16] H. Khlifi and J. Gr´ egoire, "Low-complexity offline and online clock skew estimation and removal," *Comput. Networks*, vol. 50, no. 11, pp. 1872–1884, Aug. 2006. DOI: 10.1016/j.comnet.2005.08.009
[17] T. Kohno, A. Broido, and K. Claffy, "Remote physical device fingerprinting," *IEEE Trans. Dependable and Secure Computing*, vol. 2, no. 2, pp. 93–108, 2005. DOI: 10.1109/SP.2005.18
[18] X. Mei, D. Liu, K. Sun, and D. Xu, "On feasibility of fingerprinting wireless sensor nodes using physical properties," Presented at the 27th IEEE Int. Symp. Parallel and Distributed Process. (IPDPS), May 2013, pp. 1112–1121. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6569889
[19] S. Moon, P. Skelly, and D. Towsley, "Estimation and removal of clock skew from network delay measurements," Presented at the INFOCOM Conf., 1999. [Online]. Available: https://ieeexplore.ieee.org/document/749287
[20] S. Murdoch, "Hot or not: revealing hidden services by their clock skew," Presented at the 13th ACM Conf. Computer and Communications Security, 2006. [Online]. Available: https://dl.acm.org/doi/10.1145/1180405.1180410
[21] V. Paxson, "On calibrating measurements of packet transit times," Presented at the ACM SIGMETRICS Conf., 1998, pp. 11–21. [Online]. Available: https://dl.acm.org/doi/10.1145/277851.277865
[22] L. Polc´ ak, J. Jir´ asek, and P. Matousek, "Comment on "Remote Physical Device Fingerprinting"," *IEEE Trans. Dependable and Secure Computing*, vol. 11, pp. 494–496, 2014.
[23] S. Sharma, H. Saran, and S. Bansal, "An Empirical study of clock skew behavior in modern mobile and hand-held devices," Presented at the 3rd Int. Conf. Commun. Syst. and Networks (COMSNETS), Jan. 2011. [Online]. Available: https://ieeexplore.ieee.org/document/5716494
[24] S. Sharma, A. Hussain, and H. Saran, "Experience with heterogenous clock-skew based device fingerprinting," Presented at the LASER Workshop, 2012. [Online]. Available: https://dl.acm.org/doi/10.1145/2379616.2379618
[25] M. B. Uddin and C. Castelluccia, "Toward clock skew based wireless sensor node services," Presented at the 5th Annu. ICST Wireless Internet Conf. (WICON), 2010. [Online]. Available: https://ieeexplore.ieee.org/document/5452689
[26] H. Bagci, I. Korpeoglu, and A. Yazici, "A Distributed FaultTolerant Topology Control Algorithm for Heteregeneous Wireless Sensor Networks," *IEEE Trans. Parallel Distrib. Syst*., vol. 26, no. 4, pp. 914–923, 2015. DOI: 10.1109/TPDS.2014.2316142
[27] S. Zander and S. Murdoch, "An improved clock-skew measurement technique for revealing hidden services," Presented at the 17th Conf. Security Symp., 2008. [Online]. Available: https://dl.acm.org/doi/10.5555/1496711.1496726
[28] K. Zeng, K. Govindan, and P. Mohapatra, "Non-cryptographic authentication and identification in wireless networks," *IEEE Wireless Commun*., vol. 17, no. 5, pp. 56–62, 2010. DOI: 10.1109/MWC.2010.5601959
[29] L. Zhang, Z. Liu, and C. H. Xia, "Clock synchronization algorithms for network measurements," Presented at the INFOCOM Conf., 2002. [Online]. Available: https://ieeexplore.ieee.org/document/1019257
[30] K. O. Saputra, W.-C. Teng, and T.-H. Chen, "Hough Transform Based Clock Skew Measurement Over Network," *IEEE Trans. Instrum. Meas.*, vol. 64, no. 12, pp. 3209–3216, Dec 2015. DOI: 10.1109/TIM.2015.2450293
[31] K. O. Saputra, W.-C. Teng, and T. Nara, "Hough Transform-Based Clock Skew Measurement by Dynamically Locating the Region of Offset Majority," *IEICE Trans. Inf. and Syst.*, vol. E99-D, no. 8, pp. 2100–2108, Aug. 2016. DOI: 10.1587/transinf.2016EDP7011
[32] N. P. Sastra, W. Wirawan, G. Hendrantoro, "Energy efficiency of Image Transmission in Embedded Linux based Wireless Sensor Network", *J. of Commun Soft and Sys*, Vol. 11, No. 3, Sept 2015. DOI: 10.24138/jcomss.v11i3.103.
[33] H. Wang, F. Yu, M. Li and Y. Zhong, "Clock Skew Estimation for Timestamp-Free Synchronization in Industrial Wireless Sensor Networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 1, pp. 90-99, Jan. 2021, DOI: 10.1109/TII.2020.2975289.
[34] K. Karthik and R. S. Blum, "Robust Clock Skew and Offset Estimation for IEEE 1588 in the Presence of Unexpected Deterministic Path Delay Asymmetries," IEEE Transactions on Communications, vol. 68, no. 8, pp. 5102-5119, Aug. 2020, doi: 10.1109/TCOMM.2020.2991212.
[35] J. Zhou, G. Xie, S. Yu and R. Li, "Clock-Based Sender Identification and Attack Detection for Automotive CAN Network," *IEEE Access*, vol. 9, pp. 2665-2679, 2021, doi: 10.1109/ACCESS.2020.3046862.

Nyoman Putra Sastra received the B.Eng. and M.Eng. degree in Electrical Engineering from Institut Teknologi Bandung (ITB), Indonesia, in 1998 and 2001, respectively. In 2015 he receivied his Ph.D. degree from Institut Teknologi Sepuluh Nopember (ITS), Surabaya. Since 2001, he joined with Universitas Udayana, Bali as lecturer. His research interests include wireless multimedia sensor networks and multimedia signal processing.

Komang Oka Saputra received his Ph.D. degree in Computer Science and Information Engineering from National Taiwan University of Science and Technology, Taipei, Taiwan. Since 2008 he has been a Faculty Member with the Department of Electrical and Computer Engineering, Udayana University, Bali, Indonesia. His current research interests include computer network, software engineering, e-learning, and e-exam.

Dewa Made Wiharta. received the B.Eng. and M.Eng. degree in Electrical Engineering from Institut Teknologi of Sepuluh November (ITS), Indonesia, in 1996 and Gajah Mada University (UGM) in 2003, respectively. In 2016 he receivied his Ph.D. degree from Institut Teknologi Sepuluh Nopember (ITS), Surabaya. Since 1997, he joined with Universitas Udayana, Bali as lecturer. His research interests include wireless multimedia, multimedia signal processing and robotic technology.