A Regular Pattern of Timestamps Between Machines with Built-in System Time

Ni Made Ary Esta Dewi Wirastuti, Komang Oka Saputra, and Wei-Chung Teng

***Abstract*—This paper studied the effect of 15.6 ms time resolution where the collected timestamps are in a form of parallel dotted lines, instead of one straight line like in classical case. The dotted lines made the clock skew measurement of two devices to become incorrect as the measurement which normally follow the cluster of offsets but now follow the parallel dotted lines. Dotted lines pattern is required in order to understand how to correct the clock skew measurement on data containing dotted lines. To model the dotted lines pattern is through Dotted lines Grouping Method, a tools to find the characteristics of the dotted lines. The dotted lines grouping method was then tested data obtained from wired and wireless communication of two similar devices. The dotted line grouping method results equal maximum number of dot of 10 for both data, which indicated the robustness of the dotted lines grouping method.**

***Index Terms*—Clock skew, dotted lines, time resolution, timestamps, windows time.**

I. Introduction

TIMESTAMPS or the reported time of a current event, is the essence of time-based applications. Regarding time synchronization, for instance, all devices must report the same time at the same instant, regardless of their connection through network [1–3]. Meanwhile, timestamps of sending and receiving packets are collected to measure communication delay over network [3,4]. Another application, namely clock skew estimation [5,6], intends to measure the difference in the frequency rate between the time sources, or the clocks, of two devices. The fact that most digital gadgets presently come with a digital clock allows you to delve deeper into the subject of time information. In the literatures, explorations of time information are mostly conducted in wireless sensor networks (WSN). Its open-to-program nodes and easy-to-connect system with Bluetooth or Wireless Fidelity (Wi-Fi) make WSN an appealing subject for investigating time information, such as exact time-stamping algorithms [7-9], secure time synchronization [9,10], or timestamps validation method [9–11]. Meanwhile, the more commercial and ubiquitous equipment, such as a Personal Computer (PC), server, or laptop, are less likely to have their timestamps analyzed. This is because the operating systems installed in these machines, such as Linux or Windows, are built in with system time, where their resolution level rigorously governs the accuracy of the reported time.

Manuscript received November 14, 2022; revised December 8, 2022. Date of publication April 12, 2023. Date of current version April 12, 2023. The associate editor prof. Vladan Papić has been coordinating the review of this manuscript and approved it for publication.

Ni Made Ary Esta Dewi Wirastuti and Komang Oka Saputra are with the Study Program of Electrical Engineering, Faculty of Engineering, Udayana University, Bali, Indonesia (e-mails: {dewi.wirastuti, okasaputra}@unud.ac.id).

Wei-Chung Teng is with the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan (e-mail: weichung@csie.ntust.edu.tw).

Digital Object Identifier (DOI): 10.24138/jcomss-2022-0130

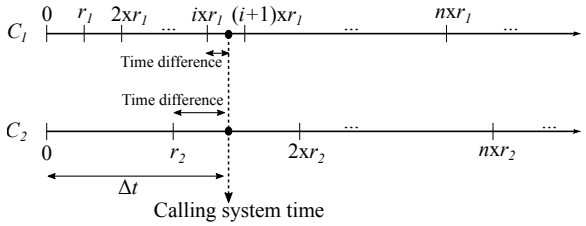

Fig. 1. Time diagrams of two digital clocks with different clock resolutions.

Time is a continuous function. When a computer invokes its system time to get timestamps, it treats the continuous time into discrete time series, where the time unit is called as *tick* [12]. The magnitude of the range between ticks, or the clock resolution, rules the accuracy of the reported time. For example, Fig. 1 shows time diagrams of two clocks, C_1 and C_2, where their resolutions are r_1 and r_2 respectively [13]. As r_1 is higher than r_2, for an equal range of time, C_1 ticks more often than C_2. When C_2 has ticked once, C_1 could have been for multiple times. For instance, when r_1 is 1 μs and r_2 is 1 ms, one tick of r_2 means 1000 ticks of r_1. The two black circles in Fig. 1 illustrate two requests at the same instant to C_1 and C_2. Here, while C_1 reports time of ixr_1, C_2 gives time of r_2. By notating the time-stamping location as t, and the elapsed time from a time reference (0 in this example) to t as Δt, the reported time for any clock resolution r can be expressed by

$$reported\ time = \lfloor \frac{\Delta t}{r} \rfloor * r \tag{1}$$

where $\lfloor \frac{\Delta t}{r} \rfloor$ only is the number of ticks that has been passed from the reference. In practice, a time reference can be the UNIX epoch time (January 1, 1970, 00:00:00) at Coordinated Universal Time (UTC), or other time references set by the operating system, like 00:00:00 UTC of the first day on the current year, or it can also be the time when the computer booting.

Another information in Fig. 1 is that the discretization process leaves differences for the reported times comparing to the real one. These time differences are caused by the rounded down process in Expression 1. The higher the clock

resolution r in Expression 1, the fewer the time difference will be, resulting in a more precise reported time as well [14]. If the requirement is only a precise time, choosing an Operating System (OS) with a high clock resolution is the answer. However, for a wider impact, it is known also that high clock resolutions can harm battery life, waste power, or even slow the computer. These facts make lower clock resolutions also a proper option. Among available OSs for computer, Linux is one of those embedded by a system time with a high clock resolution of 1 μs. Windows, on the other hand, by default has a resolution of 15.6 ms on its system time.

One of the time-stamping methods is by employing a measurer to collect timestamps of a sending device. Basically, the recorded timestamps are sorted based on the order of the receiving time, $\{(t_1, o_1), (t_2, o_2), \ldots, (t_n, o_n)\}$, where t is the receiver time, and o is offset (receiver time - sending device time). For further analysis, these offsets are modeled in a scatter diagram like in Fig 2(a). Each offset in this figure lines up into a straight line from the first offset to the last one. The decreasing trend of the offsets meanwhile, is caused by the clock skew of both devices [5, 6], where its value can be estimated by using linear regression [5], minimum-offsets approaches [17], linear programming algorithm (LPA) [5], or the Hough transform (HT)-based method [18].

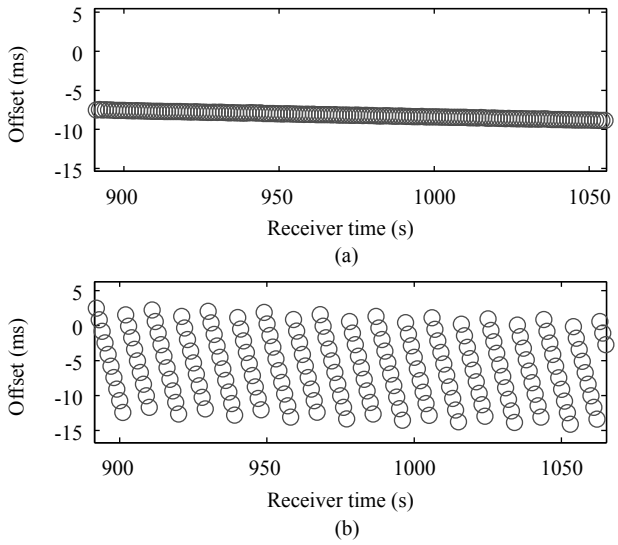

Fig. 2. Parts of the 3000 pairs of timestamps of a Linux notebook and a Windows PC. Offset here is Windows' receiver time - Linux's sender time. (a) Classical case. (b) Dotted-lines case.

Apart from the common case in Fig. 2(a), the main point of this paper is a case like in Fig. 2(b), where the offsets fall into some parallel lines, instead of one straight line. As the parallel lines are formed in a dotted style by the offsets, we call this phenomenon as *dotted lines*.

The following section introduces the contribution of this paper. Section III explains the dotted lines basic concept. Section IV then detailed the proposed Dotted lines Grouping Method. Next, in Section V, the evaluation results of the proposed method on wired and wireless data are shown. Section V also presents several discussions related to the proposed method. The future work is then presented in Section VI and the conclusion of this paper is given in Section VII.

II. Contribution

The first contribution of this work is to create a method to obtain the statistics of the dotted lines, namely Dotted lines Grouping Method. The proposed method marks all lines in the dotted lines including their dots member, the start and the end dot for each lines, as well as the pattern of the number of each dot in each line. Based on the result of the proposed method we can then contribute a concept of the dotted lines, such as how the dotted lines are formed, the relationships between the slope of each line and system time resolution, and how system time resolution produces the length of each line in the dotted lines. In terms of potential practical applications, this paper used clock skew measurement as the analysis's foundation. The effect of the dotted lines on clock skew is next examined in comparison to typical circumstances. In the opposite direction, how different clock skews affect the pattern of the dotted lines is investigated. All the concepts have been evaluated through simulations, and also experiments on wired and wireless networks.

III. Dotted Lines: Analysis and Characteristics

A. The Occurrence and Characteristics of the Dotted Lines

To explain how the dotted lines arise, let use an example as follows. There is a device D that sends its timestamps, one per 1000 ms, to two measurers, $M1$ and $M2$, where their clock resolutions are 1 μs, 1 μs and 15.6 ms respectively. For the shake of simplicity, the clocks of all machines are fully synchronized, and the delays between them are neglected. In advance, D, $M1$, and $M2$ implemented Expression 1 when producing their timestamps.

Table I shows the time diagrams of D, $M1$ and $M2$, from their first 21 timestamps. For a fully-time synchronized and neglected-delay system, all timestamps of $M1$ and $M2$ should be similar with C's timestamps. Moreover, the clock resolutions of $M1$ and $M2$ play their role here. While $M1$'s microsecond resolution produced precise timestamps as shown in "$M1$ time" column, the rounded down process of the 15.6 ms resolution at $M2$ let the reported timestamps contain time differences as shown in "$M2$ time" column. Accordingly, the time differences in $M2$'s timestamps give direct impacts to the offsets between D and $M2$ (see "$DM1$ offset" and "$DM2$ offset" columns).

The pattern of the offsets on both cases can then be found in Figs. 3(a) and 3(b). It is clear from Fig. 3(a) that D and $M1$ result a horizontal pattern of offsets due to the zero skew between them (a fully-time synchronized system). Fig. 3(b), meanwhile, shows how the offsets of D and $M2$ are formed by the rounded down process at $M2$. Observe the first line on Fig. 3(b) and the first ten packets in Table I for the rationale. On each $M2$ timestamp, 1.6 ms, the remainder of the rounded process of the 15.6 ms resolution, is gathered.

TABLE I
TIME DIAGRAMS BETWEEN D, $M1$, AND $M2$

Packet number	D time	$M1$ time	$DM1$ offset	$M2$ time	$DM2$ offset
1	0	0	0	0	0
2	1000	1000	0	998.4	-1.6
3	2000	2000	0	1996.8	-3.2
4	3000	3000	0	2995.2	-4.8
5	4000	4000	0	3993.6	-6.4
6	5000	5000	0	4992	-8
7	6000	6000	0	5990.4	-9.6
8	7000	7000	0	6988.8	-11.2
9	8000	8000	0	7987.2	-12.8
10	9000	9000	0	8985.6	-14.4
11	10000	10000	0	9999.6	-0.4
12	11000	11000	0	10998	-2
13	12000	12000	0	11996.4	-3.6
14	13000	13000	0	12994.8	-5.2
15	14000	14000	0	13993.2	-6.8
16	15000	15000	0	14991.6	-8.4
17	16000	16000	0	15990	-10
18	17000	17000	0	16988.4	-11.6
19	18000	18000	0	17986.8	-13.2
20	19000	19000	0	18985.2	-14.8
21	20000	20000	0	19999.2	-0.8

These accumulated values form $DM2$'s offsets into -1.6 ms, -3.2 ms, and so on until -14.4 ms, where these offsets fall into one straight line as illustrated in Fig. 3(b).

As the 15.6 ms resolution fully bounds the time-stamping process, the accumulation of the rounded down remainder is also bounded by the value of 15.6 ms. The offsets show that they terminate at -14.4 ms since the next value, which should be -16 ms, has past the boundary of -15.6 ms. When the boundary is exceeded, a new accumulation process is started, along with the creation of a new straight line similar to the second line in Fig. 3(b), which is formed by packets 11 to 20. For this example, where the sending interval at D is 1000 ms, the number of offset (dot) in a line is 10. Expression 2 then shows how to calculate the number of dots of each line for any sending interval int (in the unit of ms).

$$number\ of\ dot = \lfloor \frac{15.6}{mod(int, 15.6)} \rfloor + 1 \qquad (2)$$

From the example above, the skew of each line is the rate of the offsets accumulation, -1.6 ms for every 1 second, or -1.6 ms/s (-1600 ppm). On practical applications however, the line skew is the ratio between the offsets range and the elapsed time from the offsets of a particular line as illustrated in Fig. 3(b) and then expressed in Expression 3.

$$line\ skew = \frac{\Delta o}{\Delta t} = \frac{offset_{last} - offset_{first}}{rec.\ time_{last} - rec.\ time_{first}} \qquad (3)$$

To this point, the condition between $M2$ and D, which is set to be fully time synchronized, has not been figured yet in Fig. 3(b). Even if LPA is implemented, the lower bottom of the two lines cannot be bounded as a horizontal line to indicate a zero skew.

More offsets are clearly needed to show the time-synchronized condition. Fig. 3(c) shows an extension of Fig. 3(b), where 117 offsets of $M2$ and D are involved. It can be seen that the lower bottom of the offsets in Fig 3(c) can now

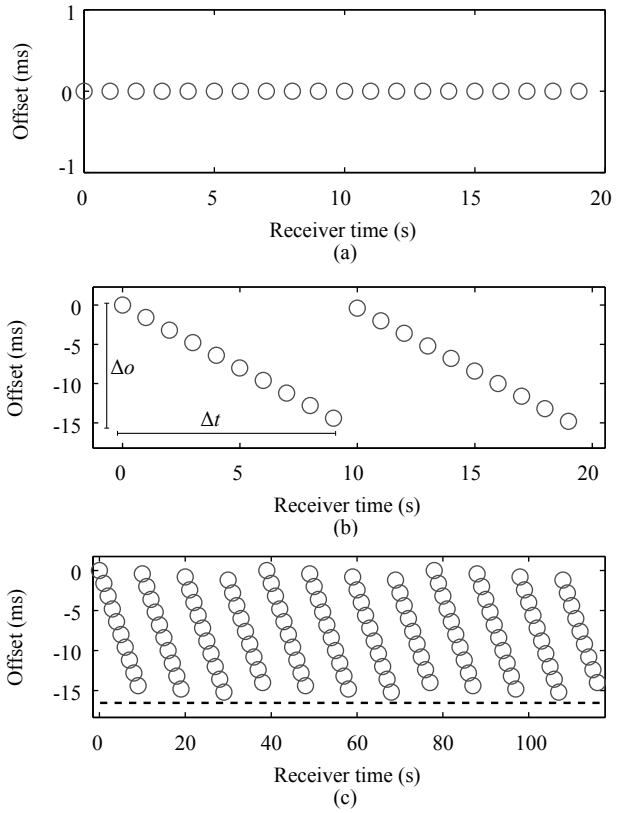

Fig. 3. Offset-set of the case in Table I. (a) 20 offsets of D and M1. (b) 20 offsets of D and M2. (c) 117 offsets of D and M2.

be bounded by a horizontal LPA line (the dashed line) to show a 0 ppm skew.

One fact from the above explanations is, even they need more offsets, the dotted lines do not alter the clock skew of the observed devices comparing to the condition when they are in the normal case.

Another unique fact from Fig. 3(c) is that the number of dot in each line does not always follow Expression 2. Among several lines with ten dots, there exist lines with nine dots only. Like the other lines, these lines are also started when the offsets accumulation has exceeded the -15.6 ms. However, in this case, the -15.6 ms is exceeded sooner than usual. This anomaly happens due to the starting offset of each line also encounters accumulation every time the -15.6 ms is exceeded. This condition can be found in Table I, "$DM2$ offset" column, at the packet number 11 and 21, where the starting offsets of the second and third lines are -0.4 ms and -0.8 ms (accumulated by -0.4 ms). The accumulations on the starting offset affect the ending offset as well. From -14.4 ms as the ending offset of the first line, it then becomes -14.8 ms in the second line. The third and fourth values (they are unwritten in Table I), should be -15.2 ms and -15.6 ms respectively. However, as -15.6 ms has exceeded the resolution boundary, the last offset of the fourth line is only -14 ms, or the fourth line has one dot fewer than normal.

As there exist lines with number of dot that is fewer than the one obtained by Expression 2, for the rest of this paper, result of Expression 2 (and also later Expression 4) is named

TABLE II
TIME DIAGRAMS BETWEEN D, $M3$, AND $M4$

Packet number	D time	$M3$ true time	$M3$ time	$DM3$ offset	$M4$ true time	$M4$ time	$DM4$ offset
1	0	0	0	0	0	0	0
2	1000	999.9	998.4	-1.6	1000.2	998.4	-1.6
3	2000	1999.8	1996.8	-3.2	2000.4	1996.8	-3.2
4	3000	2999.7	2995.2	-4.8	3000.6	2995.2	-4.8
5	4000	3999.6	3993.6	-6.4	4000.8	3993.6	-6.4
6	5000	4999.5	4992	-8	5001	4992	-8
7	6000	5999.4	5990.4	-9.6	6001.2	5990.4	-9.6
8	7000	6999.3	6988.8	-11.2	7001.4	6988.8	-11.2
9	8000	7999.2	7987.2	-12.8	8001.6	7987.2	-12.8
10	9000	8999.1	8985.6	-14.4	9001.8	9001.2	1.2
11	10000	9999	9984	-16			
12	11000	10998	10998	-2			

as "maximum number of dot", instead of "number of dot" only. The pattern of the dotted lines can then be figured as "some lines with maximum number of dot that are somehow intercalated by some lines with one fewer number of dot".

B. Relations Between Dotted Lines and Clock Skew

Another case is used to explain how clock skews affect dotted lines. D is now communicating with two other measurers, $M3$ and $M4$, where both are in 15.6 ms resolution. Meanwhile, the relative clock skew between $M3$ to D is -100 ppm, and $M4$ to D is 200 ppm. To support the explanations, Table II provides the time diagrams.

In Table II, contents in "$M3$ true time" and "$M4$ true time" columns are the true times of $M3$ and $M4$ before they are time-stamped by using Expression 1. It can be seen from the two columns that $M3$ has decreased times, 100 μs per second (-100 ppm), while $M4$' times are increased 200 μs per second (200 ppm).

On the previous case, the system is fully controlled by the rounded down process when timestamps are produced. Now, there is a clock skew accumulation that adds/reduces the true time before the time-stamping process. However, as clock skew accumulates in a small magnitude, at least in hundreds of microsecond per second [14], its accumulation does not always affect the reported time, but when the value is large enough to make a 15.6 ms change in the numerator of Expression 1. For instance, in "M3 time" column, the accumulation of -100 μs/s takes effect after 11 timestamps, where the reported time by $M3$ is one tick (15.6 ms) fewer than the normal case without skew (see "$M2$ time" in Table I). Meanwhile, the 200 μs/s accumulation in $M4$ takes effect after 10 timestamps, where in this case, the reported time by $M4$ is one tick more than the normal case.

As offset is a function of receiver time, the one tick fewer or one tick more than normal that occurs in the receiver time change also the offsets accumulation boundary. From the two examples, while the positive skew makes the value to be exceeded in offsets accumulation is less than 15.6 ms, the negative skew makes it larger. The direct result from the change in the offsets accumulation boundary is the maximum number of dot. Mathematically, the maximum number of dot in dotted lines for any interval int, and clock skew $skew$ (in the unit of ms/ms), can be expressed by

$$number\ of\ dot = \lfloor \frac{15.6}{mod(int * (1 + skew), 15.6)} \rfloor + 1 \quad (4)$$

Theoretically, the larger the observed $skew$ in Expression 4 the less the maximum number of dot be, and vice versa.

C. Marking the Member of Each Line

Apart from analyzing the offsets, the number of tick of each receiver time can also be used to show how every lines are formed in the dotted lines. On every time-stamping event at the receiver, the ideal number of tick, namely "base tick", is expressed in Expression 5

$$base\ tick(i) = i * \lfloor \frac{RecT(i)\ -\ RecT(i-1)}{15.6} \rfloor \quad (5)$$

where $RecT$ is used to notate the receiver time, i is started from 2, and the first receiver time, $RecT(1)$, is zero as exemplified in Tables I and II.

Based on this expression, the base tick of every receiver time in Table I are: {0, 64, 128, 192, 256, 320, 384, 448, 512, 576, 640, 704, 768, 832, 896, 960, 1024, 1088, 1152, 1216, 1280}.

However, as time passes by, the remainder accumulation of the rounded down processes alter each number of tick to not follow the base tick rule. Below is how the remainder accumulation when affecting the number of tick is modeled, where the number of tick for every receiver time is called as "real tick".

$$real\ tick(i) = \lfloor \frac{i * (RecT(i)\ -\ RecT(i-1))}{15.6} \rfloor \quad (6)$$

By using Expression 6, the real tick of each receiver time in Table I can be listed as {0, 64, 128, 192, 256, 320, 384, 448, 512, 576, 641, 705, 769, 833, 897, 961, 1025, 1089, 1153, 1217, 1282}. The difference between the real tick and the base tick, namely "diff tick", can then be calculated as {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2}.

Diff tick always increases by one every time a new line is started, where, therefore, diff tick can be used to mark the member of each line on the dotted lines. Packets with diff tick of 0 indicate that they are the members of the first line, 1 is the second line, and so on and so forth.

IV. DOTTED LINES GROUPING METHOD

Based on the diff tick concept, an algorithm is designed to collect the statistics of the dotted lines. However, in practice, the receiver times are not always in a fixed interval, due to the possible occurrence of packet losses. When packet losses occur before the ith received packet, the value of "$RecT(i)$ $-$ $RecT(i-1)$" in Expression 5 and 6 to become incorrect. To overcome this issue, let first simplify both expressions by combining them into Expression 7, where $dtick$ here notates the diff tick value.

$$dtick(i) = \lfloor \frac{RecT(i)\ -\ j * \lfloor \frac{int}{15.6} \rfloor * 15.6}{15.6} \rfloor \quad (7)$$

Inside the rounded down bracket, the left hand side is the real tick part, and the right hand side is the base tick part. The

Algorithm 1 Dotted lines grouping method

Require: $RecT$, $PackL$, $SenT$, int
1: $DotL$ = $null$
2: $PackL$ = $null$
3: j = 0
4: **for** $i = 1$; $i < RecT$.length; i++ **do**
5: **if** $i > 1$ **then**
6: $loss$ = floor(($SenT(i) - SenT(i-1)$)/int)
7: **if** $loss > 0$ **then**
8: Create a new row in $PackL$
9: Add i to $PackL$($PackL$.rowlength - 1)
10: Add $loss$ to $PackL$($PackL$.rowlength - 1)
11: $j = j + loss$
12: **end if**
13: **end if**
14: $dtick$ = floor(($RecT(i) - j$*floor((int/15.6)*15.6))/15.6)
15: **if** $DotL == null$ **or** $DotL$.rowlength $\leq dtick$ **then**
16: Create ($dtick - DotL$.rowlength $- 1$) new row in $DotL$
17: Add i to $DotL$($DotL$.rowlength $- 1$)
18: **else**
19: Add i to $DotL(dtick)$
20: **end if**
21: j = j + 1
22: **end for**

receiver time after the occurrence of packet losses is related only to the real tick. Therefore, to obtain a correct $dtick$, the base tick part has to be adjusted to run in the same number of packet as the real tick does. To obtain that purpose, the base tick part uses variable j instead of i. In the case of no packet losses occur, j is equal to $i - 1$, but not when packet losses exist. To keep the distance between j and i, j must be added by the number of the packet losses. Hence, the relation between the base tick and the real tick is corrected.

Algorithm 1 is the practical implementation of Expression 7. $RecT$, int, and $dtick$ here remain: all the recorded receiver times, the interval between consecutive packets sent to the receiver, and the diff tick value for each receiver time respectively. The other parameters: $DotL$, is an array list to store all detected lines including their member of dot; $PackL$, is an array list to store the position of detected packet loss as well as its number; $SenT$, is an array that is composed by all the sender times; and $loss$, is a variable to save the number of packet losses.

Before $dtick$ can be obtained in line 14, the occurrence of packet losses is counted first in line number 5 to 12. This simple yet effective procedure state that packet losses exist only when the sequence of sender times is broken, indicated by the difference between two consecutive sender times observed in line 6 is larger than int. The number of detected packet losses, $loss$, is then added to j in line 11, to follow the rule of Expression 7. Meanwhile, i, the offset number of the detected packet losses and its corresponding $loss$, is stored in $PackL$.

Lines 14 to 20 are intended to input all detected lines and their member (variable i) based on their $dtick$ into $DotL$. The row index of $DotL$, {0, 1,..., N}, are used to indicate the $dtick$ values. It means that the values stored in the columns of the first row (indexed by 0), are the packets number of the first line of the observed dotted lines. The second row is for the second line, and so on and so forth. To input i of a $dtick$ resulted in line 14, the latest condition of $DotL$ has to be checked first (line 15). If the associated row of the current

Algorithm 2 Creating dotted lines for simulation data

Require: $skew$, int, N
1: **for** $i = 0$; $i < N$; i++ **do**
2: $true\_t$ = int*i
3: $SenT(i)$ = $true\_t$
4: $RecT(i)$ = floor(($true\_t$ + ($skew$*int/1000))/15.6)*15.6
5: **end for**

$dtick$ does not exist, new row(s) is/are created first in $DotL$ (line 16). Afterwards, i is put in the first column of the newly created row (line 17). Meanwhile, when the associated row is available, i of the current $dtick$ is placed directly in the first empty column of the corresponding row (line 19). This process is repeated for all the recorded receiver times in $RecT$.

After the grouping method ends, some information can be extracted from $DotL$. The number of row in $DotL$ is the number of detected lines. The maximum number of dot (Expression 4) is found from the row with the longest column. And also, the skew of each line can be obtained by running Expression 3 on the values of the first and the last columns of each row. Meanwhile, the total number of packet losses is the sum of all field in the second row of $PackL$.

V. Result and Discussion

Dotted lines from simulations as well as from experiments on a real environment were used for evaluations. The simulation samples were created based on Algorithm 2. int, $RecT$, and $SenT$ are the same parameters with those in Algorithm 1. $skew$ remains also the relative skew between the sender and receiver. The other parameters: N, is the number of timestamps being created; and $true\_t$, is the accumulated true time from the first timestamp to the $N(th)$ timestamp. As the sender is the reference (skew receiver relative to sender), $SenT$ is set similar for every interval to $true\_t$ (line 3). On the other hand, line 4 shows how $RecT$ is manipulated to form the desired dotted lines. Meanwhile, the real-environment experiments were conducted on a wired local area network, where a notebook using Ubuntu 14.04 OS sent its timestamps, one per 1000 ms, to a PC with dual OSs of Windows 7 and Ubuntu 14.04.

The first evaluation is to demonstrate the correctness of the proposed dotted lines grouping method. The proposed method was tested on 3000 offsets collected when the PC receiver is set to use its Windows OS. Part of the offsets of this sample are figured in Fig. 2(b). It has been mentioned in Section III.A that the skew of each line is close to -1.6 ms/s. However, any wrong detection by the proposed method, can make the detected lines to have wrong members, which definitely can cause wrong line skews as well. Therefore, as a proof of the robustness of the proposed method, the skew of all detected lines must be all as close as possible to -1.6 ms/s. Next, the second check is related with the last member of each detected line. One of the methods for estimating clock skew is by obtaining the slope of offsets positioned in the lower bottom of the scatter diagram [16,17]. Theoretically, all offsets of the last member of each line are positioned in the lower bottom of the scatter diagram, where the slope of a linear regression method [5] on them all is the clock skew of the collection. However, any wrong

offset placed as the last member by the proposed method can create outliers on the collected last-member offsets, which can alter the slope and then degrade the clock skew. Therefore, as the second proof of the robustness of the proposed method, the resulted clock skew must be as accurate as possible. A clock skew reference is used to validate the resulted clock skew, where it is obtained from a classical offset-set collected when the PC receiver is set to use its Ubuntu 14.04. Part of the classical offset-set are shown in Fig. 2(a). As both samples are taken from the same two devices, their clock skews should be very similar, if not the same. Therefore, the closer the clock skews of both samples, the more robust the proposed method is.

The second evaluation is addressed as a proof of concept for the relations between the maximum number of dot with the value of clock skew. The lower the skew the more the maximum number of dot be, and in the opposite way, the higher the skew the less the maximum number of dot be, stated in Section III.B. To obtain a skew trend from high to low, eight values of skew were observed: -400 ppm, -300 ppm, -200 ppm, -100 pm, 100 ppm, 200 ppm, 300 ppm, and 400 ppm. For those eight skews, eight dotted-lines simulation samples were created by using Algorithm 2. However, the unavailability of devices with the targeted clock skews is a crucial problem to produce real-environment data. As a solution, a method to imitate any clock skews introduced in [15] can be employed. Through this method, eight experiments were conducted, where in each experiment the notebook imitated one of the targeted skews to become its skew. Samples from simulations and real-network experiments were then input into the dotted lines grouping method to count their maximum number of dot. The trend of the maximum number of dot of all samples should follow the concept stated above.

A. Evaluating the Dotted Lines Grouping Method

1) Estimating Line Skew: Table III shows some results of the dotted lines grouping method when estimating the full-sample of Fig. 2(b). Three of the 309 detected lines, line number 9, 71, and 226, are detailed in Table III. The skews of these lines that are very close to -1.6 ms/s indicate that the detected lines are correct lines that contain correct members as well. Furthermore, the average value of -1.6581 ms/s from the 309 line skews with a fluctuation of only 0.0335 ms/s show that all lines are detected correctly by the proposed method.

2) Estimating the Global Clock Skew: At first, the full-sample of Fig. 2(a) was estimated by using LPA. The clock skew of -7.51 ppm was obtained, and then it is used as the reference for the clock skew of the dotted lines grouping method when estimating the full-sample of Fig. 2(b).

All the 309 offsets of the last member of all lines produced by the grouping method are depicted in Fig. 4. The skew estimations by using linear regression on these offsets are shown in Table IV. The skew of all the 309 offsets is -7.81 ppm, very close to the skew reference with an error of only 0.3 ppm. The shorter-time estimations were also conducted by using only 50 offsets for each estimation. The skew fluctuation

TABLE III
RESULTS OF THE GROUPING METHOD ON THE FULL SAMPLE OF FIG. 2(B)

Number of offset	3000
Number of line	309
Sample #1 (line number 9)	
Number of dot	10 (78 to 87)
Line skew	-1.6207 ms/s
Sample #2 (line number 71)	
Number of dot	9 (679 to 687)
Line skew	-1.6190 ms/s
Sample #3 (line number 226)	
Number of dot	10 (2190 to 2199)
Lines kew	-1.6187 ms/s
Average of all line skews	-1.6581 ms/s
Max value of all line skews	-1.6181 ms/s
Min value of all lines kews	-1.6516 ms/s
Max - Min	0.0335 ms/s

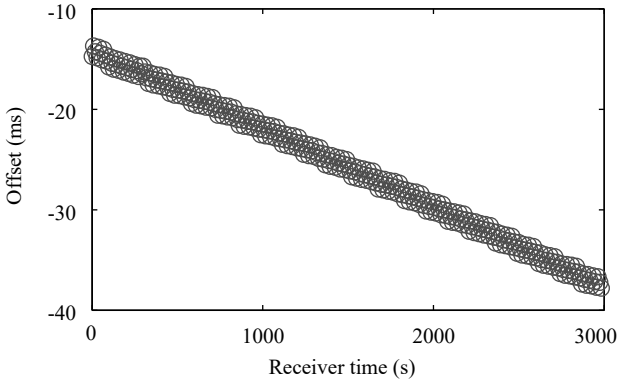

Fig. 4. 309 offsets of the detected lines in the full sample of Fig. 2(b).

of only 0.46 ppm as depicted in Table IV shows that the trend of the offsets is stable. From the distribution of the offsets meanwhile, the way they are gathered densely with no outliers occur, can conclude that all the offsets must be from the lower bottom of the scatter diagram. All these facts indicate that the proposed method has grouped all the dots into their corresponding lines correctly.

3) Evaluating the Relations Between the Maximum Number of Dot and Clock Skew: The maximum number of dot of the experimental and simulation samples are depicted in Table V. From two values of interval, 500 ms and 1000 ms, both the experiments and simulations show that clock skew can change the maximum number of dot in dotted lines. The 500 ms interval gives maximum number of dot from 16 to 26. The 1000 ms interval meanwhile, are from 8 to 14. The consistent results between simulations and experiments in Table V confirmed that the larger the skew value makes the dotted lines to have less number of dot, and vice versa.

B. Discussion

The previous experiments analyzed the concept of dotted lines and evaluated their correctness. However, the analysis are bounded in a neglected and stable delay, the wired network. In practice however, the sending device and the receiver are most likely connected in a network with higher delay jitter such as wireless network. To complete the scope of the dotted lines study, this section provides some discussions and

TABLE IV
CLOCK SKEWS OF THE 309 OFFSETS IN FIG. 4

Long-term measurement		Short-term measurements	
Offset	Clock skew (ppm)	Offset	Clock skew (ppm)
1 to 309	-7.81	1 to 50	-7.45
		51 to 100	-7.43
		101 to 150	-7.89
		151 to 200	-7.59
		201 to 250	-7.71
		251 to 309	-7.89
		Average	-7.66
		Max - Min	0.46

TABLE V
RELATIONS BETWEEN CLOCK SKEW AND MAXIMUM NUMBER OF DOT

Clock skew (ppm)	Maximum number of dot			
	Experiment		Simulation	
	500 ms interval	1000 ms interval	500 ms interval	1000 ms interval
400	16	8	16	8
300	17	9	17	9
200	18	9	18	9
100	19	10	19	10
-7.8 (original skew)	20	10	20	10
-100	21	11	21	11
-200	23	12	23	12
-300	24	13	24	13
-400	26	14	26	14

considerations for the case when the dotted lines stand on a wireless network.

1) The Shape of the Dotted Lines: When packets travel with a similar delay, the arriving times are constant, where the rounded down process of the receiver can freely form the offsets into dotted lines as explained in Section 1. However, when some packets travel on higher delay, they arrive longer than normal, where their offsets are normally separated up (outliers) from the other offsets with unpredictable distance [14,15]. Uniquely, in dotted lines, the distance between the outliers with the other offsets can be estimated, thanks to the 15.6 ms boundary when producing timestamps in Expression 1. For an illustration, Fig. 5 shows the first three lines of the dotted lines in Fig. 3(c) but with some outliers exist, noted by black circles. Let first observe Expression 1 again, where, it can be noted that all the reported times by this expression are in multiplies of 15.6 ms. As a result, the extra delay experienced by a packet can give effect to the offset also in multiplies of 15.6 ms. For example, packet number 4 can be said to arrive at the measurer between 31.2 ms to nearly 46.8 ms longer than normal, which makes the offset to jump 2x15.6 ms up from its normal position.

As every line has one tick larger (15.6 ms) than the previous line (explained in Section IV), when an offset jumps up due to an extra delay, it will be positioned on another line depends on the extra delay value. Packet number 12 for instance, it moves from the first line to the second line, due to an extra delay of only 15.6 ms from normal. Since the effect of higher delay only to move offsets from one line to another line, the dotted lines grouping method is still an adequate tool for observing the dotted lines' characteristics.

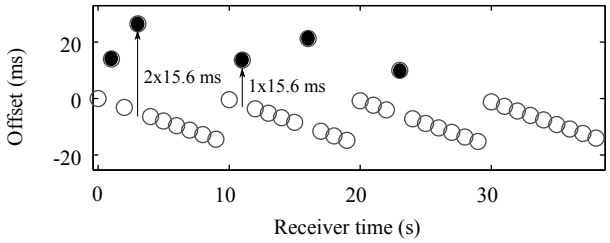

Fig. 5. Dotted lines when affected by high delay.

2) Line Skew and the Maximum Number of Dot: Theoretically, even some offsets move from their original lines to another lines, the line skews still remain close to -1.6 ms/s. The rationale is, even the delay variation of each received packets give some effect to the numerator of Expression 1 when the receiver producing timestamps, the remainder of the rounded down process does not change much.

The maximum number of dot meanwhile, is not likely to be found on the line with the highest member, as now each line can be added by some unwanted offsets from previous lines. However, there is one value that still represents the ideal number of dot in dotted lines, on whatsoever the lines condition is, the average number of dot per line. The average number of dot per line is calculated from the total number of received offsets, the number of packet loss, and the total number of detected lines. The relation with the maximum number of dot can then be expressed by Expression 8 below.

$$max\ dot = \lceil \frac{received\ packets\ +\ packet\ loss}{number\ of\ line} \rceil \quad (8)$$

To proof these two concepts, the notebook previously connected to the PC through a wired connection, was reconnected through a wireless one. The sending interval and the number of timestamps being sent remain 1000 ms and 3000 timestamps respectively. Part of the offsets from this experiment are shown in Fig. 6, where their statistics obtained by the grouping method are written in Table VI. From the three sample lines, number 79, 226, and 284, their number of dot are highly difference, 13, 10, and 7, where their members are also not in fixed orders. All these facts indicate the effect of high delay in the experiment.

Table VI also shows that the line skews fluctuate higher comparing with the results on the wired network. The small difference between the fluctuations of both cases (0.00335 ms/s and 0.1393 ms/s) proofs that the line skews are more influenced by the rounded down process than the delay effect.

Meanwhile, 13 as the highest number of dot of all lines is definitely not the maximum number of dot of this dotted-lines sample. From the 3000 received offsets (with no packet loss) and the number of detected line of 317, the average number of dot per line is 9.43, and then the maximum number of dot is 10. Even the detected lines do not provide correct number of dot, their statistics show that this sample should have maximum number of dot of 10, which in fact remains consistent with the result of the wired-connection experiment.

3) The Global Clock Skew: All offsets of the last member of the detected 317 lines are shown in Fig. 7. The offsets do not

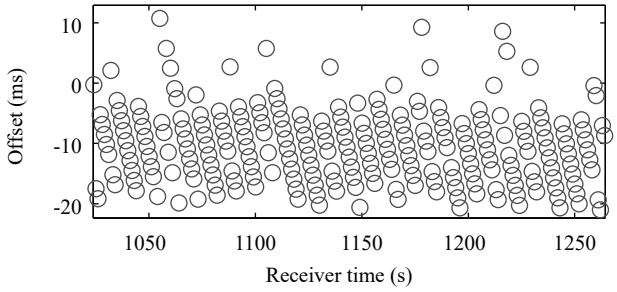

Fig. 6. Part of offsets from the experiment on wireless network.

TABLE VI
RESULTS OF THE GROUPING METHOD ON THE FULL SAMPLE OF FIG. 6

Number of offset	3000
Number of line	317
Sample #1 (line number 79)	
Number of dot	13 (720, 732, 734 to 744)
Line skew	-1.6614 ms/s
Sample #2 (line number 226)	
Number of dot	10 (2120, 2123 to 2131)
Line skew	-1.6601 ms/s
Sample #3 (line number 284)	
Number of dot	7 (2671 to 2674, 2677 to 2679)
Lines kew	-1.6617 ms/s
Average of all line skews	-1.6629 ms/s
Max value of all line skews	-1.5607 ms/s
Min value of all lines kews	-1.7001 ms/s
Max - Min	0.1393 ms/s
Average number of dot per line	9.43
Maximum number of dot	10

line up as dense as the distribution of the previous experiment on the wired network (See Fig. 4). It has been stated before that some offsets can jump up from their original line to another line. When incidentally the last member of a line to jump up, the upper dot of this line will be detected as the last member. As a result in the scatter diagram, the position of offsets that are not supposed to be the last member, are separated from the real-last member offsets. This is the reason why some outliers occur on the collected offsets in Fig. 7, which make the offsets are distributed with lower density comparing to those in Fig. 4.

The skew estimations of the 317 offsets in Fig. 7 are shown in Table VII. Due to the occurrence of some outliers, it is not surprised that the accuracy of the short-term measurements (1.61 ppm) is lower than the accuracy on the wired network (0.6 ppm). Meanwhile, the long-term measurement produces clock skew of -7.91 ppm, very close to the clock skew reference with an error of only 0.4 ppm.

VI. FUTURE WORK

As stated in the previous section that the line skews fluctuation are more influenced by the rounded down process of 15.6 ms than the delay effects, we can say that the position of outlier offsets that apart from the main group can be modeled into 15.6 ms pattern. Therefore, there is a chance that the outliers can be return info their original position. To reconstruct the dotted lines full of outliers into dotted lines without outliers will benefit info a more accurate skew

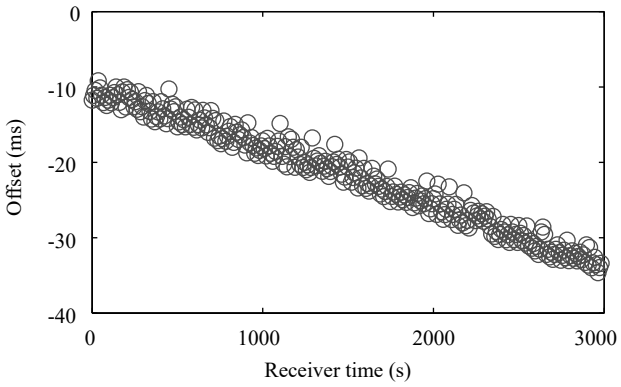

Fig. 7. 317 offsets of the detected lines in the full sample of Fig. 6.

TABLE VII
CLOCK SKEWS OF THE 317 OFFSETS IN FIG. 7

Long-term measurement		Short-term measurements	
Offset	Clock skew (ppm)	Offset	Clock skew (ppm)
1 to 317	-7.91	1 to 50	-7.07
		51 to 100	-8.04
		101 to 150	-6.61
		151 to 200	-8.22
		201 to 250	-7.18
		251 to 317	-7.58
		Average	-7.45
		Max - Min	1.61

estimation. The future work is then to develop a dotted lines reconstruction method.

VII. CONCLUSION

This paper first introduced a data when measuring timestamps on computer with low clock resolution of 15.6 ms, namely dotted lines. This paper then proposed the Dotted lines grouping method. Through this method the characteristics of the dotted lines can be obtained as follows: 1) dotted lines are composed by several dots that form parallel lines along the timestamps measurement period, 2) all lines have an equal skew with a value that is close to -1.6 ms/s, 3) dotted lines do not change the global clock skew of two devices comparing with when the two devices are in a normal case, and 4) the number of dot of their lines is affected by the observed clock skew, where, the higher the skew the less the number of dot and vice versa. The consistency of the results of the Dotted lines grouping method is then proven through dotted lines data obtained from wired and wireless communication, where on both data the proposed method show that: 1) the line skews and the maximum number of dot are consistent, and 2) the global skews of the samples taken from both networks are close to a clock skew reference. To this point, the dotted lines has been introduced and analyzed, and the supporting tool, the dotted lines grouping method, has also been confirmed to be robust for providing the characteristics of the dotted lines.

ACKNOWLEDGMENT

Authors thank to the Lembaga Penelitian dan Pengabdian Kepada Masyarakat (LPPM) of the Udayana University for the 2022 Invention Scheme of the PNBP Research Grant.

REFERENCES

[1] F. Shi, H. Li, S. X. Yang, X. Tuo and M. Lin, "Novel Maximum Likelihood Estimation of Clock Skew in One-Way Broadcast Time Synchronization," in IEEE Transactions on Industrial Electronics, vol. 67, no. 11, pp. 9948-9957, Nov. 2020, doi: 10.1109/TIE.2019.2955427.

[2] A. Y. Vinogradov and E. A. Suvorova, "Time Synchronization Influence on the SpaceFibre QoS Mechanisms," 2021 Wave Electronics and its Application in Information and Telecommunication Systems (WECONF), St. Petersburg, Russia, 2021, pp. 1-9, doi: 10.1109/WECONF51603.2021.9470644.

[3] R. Fan, W. Liu, M. Li and Z. Chai, "Clock Offset and Skew Estimation Based on Correlation Detection with One-way Dissemination in Wireless Sensor Networks," 2022 10th International Workshop on Signal Design and Its Applications in Communications (IWSDA), Colchester, United Kingdom, 2022, pp. 1-5, doi: 10.1109/IWSDA50346.2022.9870609.

[4] V. K. Chandrasegar, G. Park and J. Koh, "A Super Resolution Algorithms for Time Delay Measurement," 2021 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting (APS/URSI), Singapore, Singapore, 2021, pp. 1137-1138, doi: 10.1109/APS/URSI47566.2021.9704516.

[5] X. Liu and H. Wang, "Embedded Clock Skew Estimation in Industrial Networks," in IEEE Communications Letters, vol. 26, no. 8, pp. 1873-1877, Aug. 2022, doi: 10.1109/LCOMM.2022.3178158.

[6] H. Wang, F. Yu, M. Li and Y. Zhong, "Clock Skew Estimation for Timestamp-Free Synchronization in Industrial Wireless Sensor Networks," in IEEE Transactions on Industrial Informatics, vol. 17, no. 1, pp. 90-99, Jan. 2021, doi: 10.1109/TII.2020.2975289.

[7] Z. Wang, B. Yu, B. Pei and L. Zhang, "Research on AES encryption algorithm based on timestamp in Wireless Sensor Networks," 2nd International Conference on Information Technology and Computer Application (ITCA), Guangzhou, China, 2020, pp. 15-18, doi: 10.1109/ITCA52113.2020.00010.

[8] H. Wang, R. Lu, Z. Peng and M. Li, "Timestamp-Free Clock Parameters Tracking Using Extended Kalman Filtering in Wireless Sensor Networks," in IEEE Transactions on Communications, vol. 69, no. 10, pp. 6926-6938, Oct. 2021, doi: 10.1109/TCOMM.2021.3095155.

[9] S. Ashraf, T. Ahmed and S. Saleem, "NRSM: Node Redeployment Shrewd Mechanism for Wireless Sensor Network, " in Iran Journal of Computer Science, vol. 4, no. 9, pp. 171-183, 2021, doi:10.1007/s42044-020-00075-x.

[10] N. Ha, H. -S. Lee and S. Lee, "An algorithm for compensating synchronization error in IoT-based wireless sensor networks," 2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia), Seoul, Korea (South), 2020, pp. 1-3, doi: 10.1109/ICCE-Asia49877.2020.9276777.

[11] S. Ashraf, O. Alfandi, A. Ahmad, A. Khattak, B. Hayat, K. Kim and A. Ullah, "Bodacious-Instance Coverage Mechanism for Wireless Sensor Network," Wireless Communications and Mobile Computing, vol. 2020, pp. 1-11, 2020, doi:10.1155/2020/8833767.

[12] N. Knyazeva and E. Dukhan, "Timestamp Change Model in Windows OS," 2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT), Yekaterinburg, Russia, 2020, pp. 623-626, doi: 10.1109/USBEREIT48449.2020.9117698.

[13] N. Knyazeva, D. Khorkov and E. Vostretsova, "Building Knowledge Bases for Timestamp Changes Detection Mechanisms in MFT Windows OS," 2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT), Yekaterinburg, Russia, 2020, pp. 553-556, doi: 10.1109/USBEREIT48449.2020.9117712.

[14] D. Palmbach and F. Breitinger, "Artifacts for Detecting Timestamp Manipulation in NTFS on Windows and Their Reliability, " Forensic Science International: Digital Investigation, vol. 32, pp. s1-s9, Article 300920, 2020, doi:10.1016/j.fsidi.2020.300920.

[15] B. Singh and G. Gupta, "Analyzing Windows Subsystem for Linux Metadata to Detect Timestamp Forgery, "15th IFIP International Conference on Digital Forensics (DigitalForensics), Jan 2019, Orlando, FL, United States. pp.159-182, Jan. 2019, doi: 10.1007/978-3-030-28752-8_9.

[16] T. Cooklev, J. C. Eidson and A. Pakdaman, "An Implementation of IEEE 1588 Over IEEE 802.11b for Synchronization of Wireless Local Area Network Nodes," in IEEE Transactions on Instrumentation and Measurement, vol. 56, no. 5, pp. 1632-1639, Oct. 2007, doi: 10.1109/TIM.2007.903640.

[17] C. M. De Dominicis, P. Pivato, P. Ferrari, D. Macii, E. Sisinni and A. Flammini, "Timestamping of IEEE 802.15.4a CSS Signals for Wireless Ranging and Time Synchronization," in IEEE Transactions on Instrumentation and Measurement, vol. 62, no. 8, pp. 2286-2296, Aug. 2013, doi: 10.1109/TIM.2013.2255988.

[18] K. Oka Saputra, W. -C. Teng and T. -H. Chen, "Hough Transform-Based Clock Skew Measurement Over Network," in IEEE Transactions on Instrumentation and Measurement, vol. 64, no. 12, pp. 3209-3216, Dec. 2015, doi: 10.1109/TIM.2015.2450293.

[19] K. O. Saputra, W. -C. Teng and Y. -H. Chu, "A Clock Skew Replication Attack Detection Approach Utilizing the Resolution of System Time," 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), Singapore, 2015, pp. 211-214, doi: 10.1109/WI-IAT.2015.10.

[20] M. Aoki, E. Oki and R. Rojas-Cessa, "Scheme to measure One-Way Delay Variation with detection and removal of clock skew," 2010 International Conference on High Performance Switching and Routing, Richardson, TX, USA, 2010, pp. 159-164, doi: 10.1109/HPSR.2010.5580276.

[21] 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," 2012 IEEE 26th International Conference on Advanced Information Networking and Applications, Fukuoka, Japan, 2012, pp. 526-533, doi: 10.1109/AINA.2012.51.

[22] W. -C. Teng and J. -D. He, "Entropy-based clock skew measurements for mobile devices," 2016 Third International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC), Moscow, Russia, 2016, pp. 268-271, doi: 10.1109/DIPDMWC.2016.7529401.

[23] P. Orosz and T. Skopko, "Software-Based Packet Capturing with High Precision Timestamping for Linux," 2010 Fifth International Conference on Systems and Networks Communications, Nice, France, 2010, pp. 381-386, doi: 10.1109/ICSNC.2010.65.

[24] R. Okabe, J. Yabuki and M. Toyama, "Avoiding Year 2038 Problem on 32-bit Linux by Rewinding Time on Clock Synchronization," 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 2020, pp. 1019-1022, doi: 10.1109/ETFA46521.2020.9212079.

[25] Y. Li, B. Noseworthy, J. Laird, T. Winters and T. Carlin, "A study of precision of hardware time stamping packet traces," 2014 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), Austin, TX, USA, 2014, pp. 102-107, doi: 10.1109/ISPCS.2014.6948700.

[26] H. Marouani and M. R. Dagenais, "Comparing high resolution timestamps in computer clusters," Canadian Conference on Electrical and Computer Engineering, 2005., Saskatoon, SK, Canada, 2005, pp. 400-403, doi: 10.1109/CCECE.2005.1556956.

[27] S. B. Deb and A. Chetry, "USB Device Forensics: Insertion and removal timestamps of USB devices in Windows 8," 2015 International Symposium on Advanced Computing and Communication (ISACC), Silchar, India, 2015, pp. 364-371, doi: 10.1109/ISACC.2015.7377371.

[28] G. S. Cho, "An Intuitive Computer Forensic Method by Timestamp Changing Patterns," 2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, Birmingham, UK, 2014, pp. 542-548, doi: 10.1109/IMIS.2014.92.

[29] K. P. Chow, F. Y. W. Law, M. Y. K. Kwan and P. K. Y. Lai, "The Rules of Time on NTFS File System," Second International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE'07), Bell Harbor, WA, USA, 2007, pp. 71-85, doi: 10.1109/SADFE.2007.22.

Ni Made Ary Esta Dewi Wirastuti received the B.Eng. degree in electrical engineering from Udayana University, Bali, Indonesia, in 2000, the M.Sc. degree in mobile communication systems from University of Surrey, Guildford, United Kingdom, in 2002 and the Ph.D. degree in Telecommunication Systems from University of Bradford, West Yorkshire, United Kingdom, in 2007. From 2007 to 2009, she was a Post Doctoral Fellowship with the Mobile and Satellite Communication Research Centre (MSCRC), University of Bradford, United Kingdom worked to the VeSeL (Village e-Science for Life) project, Engineering and Physical Sciences Research Council (EPSRC) grant. Dr. Wirastuti was a recipient of the Best Paper Student in 2006 from University of Bradford for paper presentation at INTI College, Malaysia. Her research interest includes the development of physical layer model for the next wireless and mobile communication systems. She has been a lecturer in Department of Electrical Engineering, Faculty of Engineering at Udayana University, Bali, Indonesia, since 2001. She can be contacted at email: dewi.wirastuti@unud.ac.id.

Komang Oka Saputra received his B.Eng. degree from the Brawijaya University in 2004, and his M.Eng. degree from the University of Indonesia in 2006, both in electrical engineering, with the specialization of telecommunication engineering. Since 2008, he has been a faculty member of the Department of Electrical and Computer Engineering, Faculty of Engineering, Udayana University, Bali, Indonesia. Currently, he is pursuing his PhD in the Department of Computer Science and Information Engineering at the National Taiwan University of Science and Technology, Taiwan. His major research interests include computer network and security. He can be contacted at email: okasaputra@unud.ac.id.

Wei-Chung Teng received his Doctor of Engineering degree in 2001 from the University of Tokyo. He is Associate Professor of Department of Computer Science and Information Engineering at National Taiwan University of Science and Technology. His research interests include human computer interaction focusing on remote robot manipulation, network communication protocols of time synchronization, and network security issues. He can be contacted at email: weichung@csie.ntust.edu.tw.