IEICE The Institute of Electronics, Information and Communication Engineers

Editorial Board Japanese Page

SITE TOP

Login

To browse Full-Text PDF.

> Login

- Forgotten your password?

Menu

- Search

Latest Issue

- A Fundamentals
- B Communications
- C Electronics
- D Information & Systems

- Abstracts of JPN Edition

Archive

- Volume List
- Volume List [JPN Edition]

Editorial Board

- Editorial Board
- Editorial Board[JPN Edition]

Open Access Papers

- Trans. Commun. (Free)
- Trans. Electron. (Free)
- Trans. Inf.&Syst. (Free)

Link

- Subscription
- For Authors
- Article Search(I-Scover)
- Statistics:Accepting ratio,review period etc.
- IEICE Home Page

Others

- Citation Index
- Privacy Policy
- Copyright & Permissions

Editorial Committee of IEICE Transactions on Information and Systems

Online ISSN : 1745-1361

IEICE TRANSACTIONS on Information and Systems

Archive Index

Volume E99-D No.8 (Publication Date:2016/08/01)
[Whole issue]:PDF (30.9MB)

Special Section on Security, Privacy and Anonymity of Internet of Things

Regular Section

Previous | Next

go to Page Top

PAPER

Hough Transform-Based Clock Skew Measurement by Dynamically Locating the Region of Offset Majority

Komang OKA SAPUTRA[†*], *Student Member*, **Wei-Chung TENG**[†a)], *and* **Takaaki NARA**[††], *Nonmembers*

SUMMARY
A network-based remote host clock skew measurement involves collecting the offsets, the differences between sending and receiving times, of packets from the host within a period of time. Although the variant and immeasurable delay in each packet prevents the measurer from getting the real clock offset, the local minimum delays and the majority of delays delineate the clock offset shifts, and are used by existing approaches to estimate the skew. However, events during skew measurement like time synchronization and rerouting caused by switching network interface or base transceiver station may break the trend into multi-segment patterns. Although the skew in each segment is theoretically of the same value, the skew derived from the whole offset-set usually differs with an error of unpredictable scale. In this work, a method called *dynamic region of offset majority locating* (DROML) is developed to detect multi-segment cases, and to precisely estimate the skew. DROML is designed to work in real-time, and it uses a modified version of the HT-based method [8] both to measure the skew of one segment and to detect the break between adjacent segments. In the evaluation section, the modified HT-based method is compared with the original method and with a linear programming algorithm (LPA) on accumulated-time and short-term measurement. The fluctuation of the modified method in the short-term experiment is 0.6 ppm (parts per million), which is obviously less than the 1.23 ppm and 1.45 ppm from the other two methods. DROML, when estimating a four-segment case, is able to output a skew of only 0.22 ppm error, compared with the result of the normal case.
key words: *clock skew, Hough transform, region of offset majority, time synchronization*

1. Introduction

All digital devices have an embedded internal digital clock. Since there exists an error in the manufactured frequency to the ideal one, the clocks tick slightly faster or slower than physical time. This error of ticking rate is known as the clock skew, and the ticking rate difference between two digital clocks is called the relative clock skew. When two devices communicate over a network, the relative clock skew between them may cause problems for applications that demand accurate time, e.g. communication delay measurements and time synchronization [1], [2]. On the other hand, clock skew can be used to identify devices according to the unique properties of clock skew: the measured skews are stable in parts per million (ppm) precision over time, and

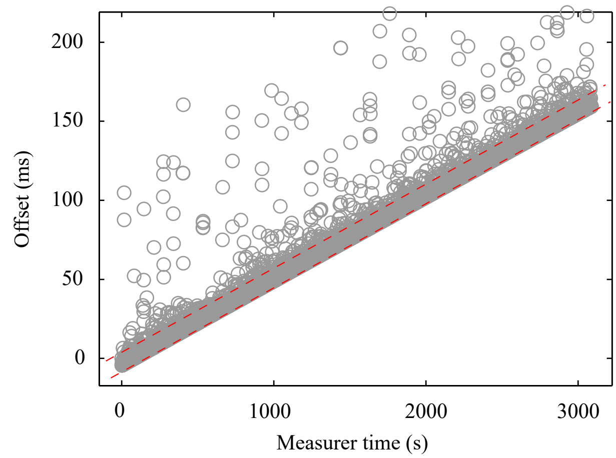

Fig. 1 A scatter diagram of offsets, or the time differences between the sending time and the receiving time of network packets.

the skews of different devices are generally distinguishable at the level of ppm precision [3]–[7]. However, irrespective of the pros and cons of clock skew, all related applications share the same fundamental requisite: an accurate clock skew measuring method.

Basically, the relative clock skew of a measurer m to a device s can be formulated as follows. For any physical time t, the time reported by the clock of m and by the clock of s are denoted by $C_m(t)$ and $C_s(t)$, respectively. Their first derivatives $C'_m(t) \equiv dC_m(t)/dt$, and $C'_s(t) \equiv dC_s(t)/dt$, $\forall t \geq 0$ are the speeds at which both clocks progress at time t. The relative clock skew of m to s, denoted by $s_{ms}(t)$, can then be calculated by $s_{ms}(t) = C'_m(t) - C'_s(t)$. However, as all digital clocks tick at a constant frequency at normal temperature, it is assumed that the relative clock skew to be measured is a constant s_{ms}.

To measure s_{ms} over the network, m collects the sending time of each network packet from s, and calculates the time offset of each packet by subtracting the sending time from the receiving time. The value of an offset is then the sum of time difference, packet processing time, and packet transmission delay. Since s_{ms} can hardly be zero, the time difference increases or decreases steadily as time passes by. Consequently, the offsets values change at a steady pace. As an example, Fig. 1 shows a scatter diagram of offsets from an experiment in Sect. 4. The increasing trend of the offsets near the bottom of the distribution reveals that the relative

[†]The author is with the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan.
[††]The author is with the Graduate School of Interdisciplinary Information Studies, University of Tokyo, Tokyo 113-0033, Japan.
[*]The author is also with the Department of Electrical and Computer Engineering, Udayana University, Bali 80361, Indonesia.
a) E-mail: weichung@csie.ntust.edu.tw
DOI: 10.1587/transinf.E0.D.1

skew of the measurer to the sending device is positive. On the other hand, the outliers near the top of the distribution illustrate how large the jitter is. A measurement conducted in an environment with stable network delay, e.g., a wired local area network, usually has offsets gathering around a thick line, as in the 1^{st} and 3^{rd} segments shown in Fig. 2(b). Conversely, when the network traffic is crowded, or when the network path is long, the offsets tend to spread to a larger area, and outliers from longer-delay packets also appear, as shown in Fig. 1. In either case, the *region of offset majority* (ROM) can be bounded by two parallel lines, like the dashed lines in Fig. 1.

Linear regression is the fastest and simplest method of deriving the relative skew from a collected offset-set. However, this method is vulnerable to outliers. Alternatively, the linear programming algorithm (LPA) developed by Moon et al. [1] is known to be robust in obtaining an accurate clock skew, which is the slope of a line that lies below all the offsets, but passes through as many offsets as possible. Since LPA uses offsets of minimal delay, which are in fact low outliers, to determine the line, it requires a large amount of time to collect enough low outliers in order to stabilize its estimation. Recently, Oka et al. proposed the Hough transform (HT)-based method, which uses the gradient of the ROM's lower boundary line as the estimated clock skew [8]. The HT-based approach provides the same level of precision as LPA, but as the ROM becomes stable with just a few hundred offsets, the measurement time can be reduced to less than 10 minutes.

As the widespread use of clock skew now ranges from cases of notebooks communicating inside a WLAN [4], [5] to smart phone applications accessing cloud services [7], [9], there is significant demand for an approach able to provide accurate estimations even to non-classical offset distributions. A well-known non-classical case is caused by time synchronization, e.g. network time protocol (NTP) [10], during clock skew measurement. Since the time of a clock jumps ahead or backward after being synchronized, the value of the following offset jumps up or down accordingly. Therefore, the continuity of offsets breaks, resulting in two or more clusters with unpredictable scales of jumps, as shown in Fig. 2(a). The solid line and the two dashed lines in the figure are the results of the LPA and HT-based method respectively. It is clear that both methods fail to give a correct estimation. It has also been reported that this kind of segmentation phenomena occurs when the route changes due to, for example, the sending device switching from a wired connection to a wireless one, or changing its mobile network relay base station [9]. Figure 2(b) illustrates a somewhat overdone example.

The main contribution of this work is a new method called *dynamic region of offset majority locating* (DROML) for estimating the skews of multi-segment offset-sets. DROML uses the HT-based method both to determine the range and to estimate the skew of each segment. When the current ROM no longer enfolds the offset majority, i.e. a jump is detected, DROML dynamically relocates its ROM to fit the offset majority of the next segment. It then merges the clock skews of all segments into a reasonable estimation. Finally, DROML is designed to adaptively estimate the global skew such that it is able to provide real-time estimation. As the second contribution, the original HT-based method [8] is also improved in order to locate the most representative ROM for a given number-of-bounded-offsets threshold. The improved version, wrapped as a function called LocateROM(), is able to provide more precise estimation than LPA and the original HT-based method.

The next section introduces the concept of the HT-based clock skew measurement and how it is improved. Next, Sect. 3, describes the design of the DROML algorithm. Section 4 examines the DROML evaluation results for both normal and multi-segment cases. Finally, Sect. 5 offers conclusions.

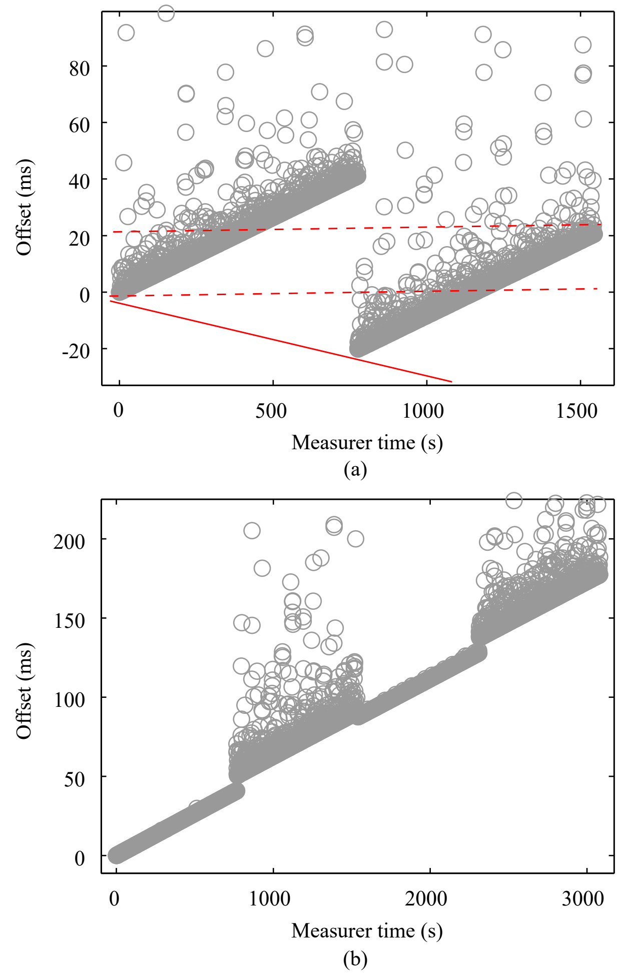

Fig. 2 Multi-segment cases: a) caused by executing NTP synchronization once during skew measurement; b) caused by switching the connecting adapter three times during skew measurement.

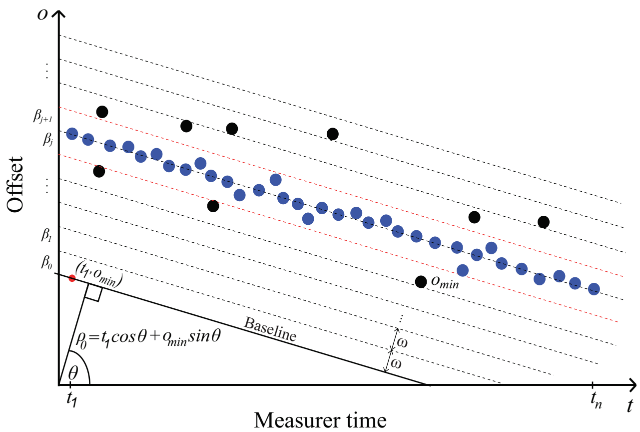

Fig. 3 Offset, ROM, and bounded offsets on the HT image points.

2. Improving the ROM Stability of the HT-Based Clock Skew Measurement

2.1 The HT-Based Method

The HT-based skew measurement can be summarized as follows. First, all collected offsets are mapped to image points, as illustrated in Fig. 3. The coordinates of these points, represented by (t_i, o_i), are retained as the measurer's time and the values of offsets. Given $S = \{(t_1, o_1), \ldots, (t_n, o_n)\}$ with $t_1, \ldots, t_n$ in increasing order, the next step is to perform HT on S. When line segments are present in the image, HT is able to gather enough votes in some distance ρ with angle θ which matches the gradient of a line. However, most offsets are close to a line instead of forming one in the skew measurement case. Accordingly, the rounded distances of these offsets do not accumulate sufficiently on any specific integer to determine a line. To overcome this problem, a lower resolution of scale ω is introduced to separate the space into many parallel yet adjoining regions. These regions, indexed from β_0, are constructed as follows: for angle θ, the baseline passes (t_1, o_{min}), and the normal vector of the baseline forms an angle of θ with the t-axis. Here o_{min} denotes the minimum offset in the whole offset-set. The lower boundaries of region β_1 and above then pile up on the baseline at a regular interval of ω, as depicted in Fig. 3. All distances are rounded accordingly to one of these regions before voting.

The next step is to choose the most appropriate region among all candidates. To do this, the modified offset voting method [8, Algorithm 1] scans all possible angles and has every point vote for its region by the following formula:

$$\beta = \left\lfloor \frac{\rho - \rho_0}{\omega} \right\rfloor \tag{1}$$

The voting results are stored in a three dimensional array, $Votes(\theta, \omega, \beta)$. Once all angles have been scanned for a certain region thickness, the region with the most votes, if the number exceeds the threshold k, is defined as the ROM. The threshold k is, by default, set to 50% of n. If there is no region fulfilling the threshold requirement, the voting process repeats with a larger ω. This method ends when the ROM is found.

A three-stage wrapper function [8, Algorithm 2] is further developed to efficiently search for the correct θ inside the range of -750.0 to 750.0 ppm in a 0.1 ppm resolution. The detail of this function is skipped, as this part is not our concern in this work. However, the result skew is derived from the angle θ of the ROM by the following formula:

$$skew = \tan(\theta - \frac{\pi}{2}) \simeq (\theta - \frac{\pi}{2}) \tag{2}$$

as $\theta - \frac{\pi}{2}$ is very close to 0.

The output of the HT-based method, the ROM, is supposed to be the thinnest region which includes at least k offsets. In other words, the ROM is the densest zone in the image. However, this is not always the case. Figure 3 above depicts a counterexample. Most offsets in Fig. 3 are very close to the boundary between β_j and β_{j+1}. If the upper half of β_j and the lower half of β_{j+1} are combined into one region, it is obvious that this region qualifies as the ROM. However, since neither β_j nor β_{j+1} earn enough votes in this case, the HT-based method has to try a larger ω. Although some thicker region will eventually be assigned as the ROM, there is no guarantee that this region of lower density will have the same gradient as the ROM of minimal thickness. This flaw stems from the fixed-region design inherited from the voting process of HT. As long as the image space is divided into equal-sized fixed regions, there remains a chance that the returned region will not be the zone with the highest density, and, accordingly, the gradient of the ROM will not be the best estimation for the clock skew measurement. To solve this problem, a modified version is proposed in the next subsection.

2.2 Finding the Most Representative ROM

In order to search for the densest region from the distances calculated by HT, this work introduces a different problem definition. For any region with thickness given, its location can be determined by the distance from the baseline to its lower boundary line along the normal vector. For example, the lower boundary of region β_j in Fig. 3 is $j\omega$ above the baseline, so any point with its distance $\rho - \rho_0$ inside the range $[j\omega, j\omega + \omega)$ must fall within region β_j. Now, instead of dividing the image space into multiple equal-thickness regions, this work locates the densest region floating along the normal vector. Since the baseline is no longer necessary, it is possible to use ρ instead of $\rho - \rho_0$ in the floating-region version. As the densest region must contain the largest number of points, the problem can be redefined as follows: given a set of distances ρ of the points, find a lower bound l such that the range $[l, l + \omega]$ covers the largest number of distances.

In this work, a straightforward method is used to solve this problem. First, all distances are sorted in ascending order. The lower bound is then set at the smallest distance. Once the number of bounded distances is counted, the lower

Algorithm 1 Searching for the thinnest ROM

Require: $S, \theta_{min}, \theta_{max}, \theta_{inc}, \omega_{min}, \omega_{inc}, k$
Output: R

1: **var** $R = null$
2: **var** $D = null$
3: **var** $n = S.length$
4: **for** $(\omega = \omega_{min}$; R is $null$ or $R.count < k$; $\omega = \omega + \omega_{inc})$ **do**
5:     **for** $(\theta = \theta_{min}; \theta \leq \theta_{max}; \theta = \theta + \theta_{inc})$ **do**
6:         **for all** (t, o) **in** S **do**
7:             **var** $\rho = t * \cos\theta + o * \sin\theta$
8:             D.insert(ρ)
9:         **end for**
10:         sort_ascending(D)
11:         **for** $(i = 1, j = 2; j \leq n; i{+}{+})$ **do**
12:             **var** $count = j - i$
13:             **while** $(D(j) \leq D(i) + \omega$ and $j \leq n)$ **do**
14:                 $count{+}{+}$
15:                 $j{+}{+}$
16:             **end while**
17:             **if** $(R$ is $null$ or $count > R.count)$ **then**
18:                 $R = (\theta, \omega, D(i), count)$
19:             **end if**
20:         **end for**
21:     **end for**
22: **end for**

bound slides to the second smallest distance to perform the second count. This process continues until the largest distance is bounded. Finally, the range with the highest count is returned as the ROM.

Algorithm 1 details the pseudocode of this method. There are 7 parameters to this algorithm, and variable R is used to save the information of the ROM. Table 1 below gives a list of all parameters and important variables with short descriptions. The main body of Algorithm 1 is enclosed by two levels of **for** loop. The outermost **for** loop searches for the ROM of thickness ω, and the second level **for** loop scans all angles inside the range $[\theta_{min}, \theta_{max}]$. Since there is no upper bound to the thickness, it is guaranteed that Algorithm 1 will always return a ROM as long as the threshold k is not larger than n. For some given θ and ω, Algorithm 1 calculates the distance ρ for every point in S and then sorts all distances into an increasing sequence (lines 6–10). The following **for** loop in lines 11–20 counts the number of covered distances inside the $[D(i), D(i) + \omega]$ range for each iteration. Variable i in this loop is the index of the smallest distance in the current range, and variable j points to the next distance to be evaluated. This loop executes the search by adopting a sliding window of length ω, so i increases by 1 in each iteration. When the region has covered the largest distance, i.e. $j > n$, it becomes impossible to obtain a region covering more distances, so the process stops sliding the range. Since j slides with i from 2 to $n + 1$, it is clear that the time complexity of this **for** loop, including the inner **while** loop, is $O(n)$.

Since the only candidate region is floating instead of fixed along the direction of the normal vector, Algorithm 1 is able to detect any qualified ROM. As the thickness ω increases gradually, and Algorithm 1 stops when R covers at least k points, the returned ROM must be of minimal thickness. Algorithm 1, when combined with the three-stage process in [8, Algorithm 2], is able to determine the ROM precisely and thus to provide an accurate clock skew estimation. The function LocateROM() denotes the combined process in the subsequent sections.

Table 1 Parameters and important variables in Algorithms 1 and 2

Parameter	Alg.	Description
S	1, 2	Set of all points
$\theta_{min}, \theta_{max}$	1, 2	Lower and upper bounds of the angle range
θ_{inc}	1, 2	Angle resolution
ω_{min}	1, 2	Lower bound of the region thickness
ω_{inc}	1, 2	Step size by which the thickness increases
k	1, 2	Threshold of the number of points inside the ROM
R	1, 2	Structure storing the information of the ROM
D	1	Array of the distances of all points in some angle
seg_{min}	2	Number of points for a segment base
inc	2	Increment of sampling range and segment extension
ω_{max}	2	Maximum allowed thickness of the ROM
k_{strict}	2	Threshold bigger than k, used only for the segment base
idx	2	Starting point to search for a segment base
idx_l, idx_r	2	Beginning and ending indexes of a segment
$Segments$	2	Array of found segments

3. Dynamic Region of Offset Majority Locating (DROML) Method for Multi-segment Cases

3.1 The Segmentation of Offsets

Based on the assumption that the offsets are distributed in one cluster, the HT-based method bounds the majority of offsets by one ROM. However, some cases, as shown in Fig. 2(a) and Fig. 2(b) involve baselines that shift during the timestamp collection process. Figure 2(a) shows a typical case when NTP time synchronization occurs during the skew measurement. As the clock time of the measurer jumps ahead by roughly 60 ms at around the measurer's time of 700 s, the offset jumps down accordingly, dividing the offsets into 2 segments. The offsets in Fig. 2(b) can be easily divided into 4 segments. The 1^{st} and 3^{rd} segments show almost the same slope, while the 2^{nd} and 4^{th} segments exhibit very similar distributions. This figure results from switching the connecting network interface of the measured device from a wired adapter to a wireless one 3 times, in the measurer's time at around 770 s, 11540 s, and 2320 s, respectively.

Although the ROM of every single segment should be of the same gradient, the gradient of the global ROM usually differs. It is not even possible to bound the gradient error because the scale in each jump is usually unpredictable. Therefore, a method to detect the locations of jumps is necessary.

3.2 Detecting the Boundaries of Segments

People can intuitively identify a jump when looking at a scatter diagram. However, designing an effective and efficient computer algorithm for this task is a different matter. Since a jump can only be detected if a segment is present, the algorithm starts by determining the beginning part of the

Algorithm 2 The DROML method

```
Require: S, seg_min, inc, ω_min, ω_max, ω_inc, k_strict, k
 1: var Segments = []
 2: var idx = 1
 3: while (idx ≤ S.length − seg_min) do
 4:     var R = LocateROM(S(idx ··· (idx+seg_min−1)), ω_min, ω_inc, k_strict)
 5:     if (R.ω > ω_max) then
 6:         idx = idx + inc
 7:     else
 8:         var (idx_l, idx_r) = FindSeg(S, idx, idx + seg_min − 1, inc, R, k)
 9:         idx = idx_r + 1
10:         Segments.add((idx_l, idx_r))
11:     end if
12: end while
13: var skew_sum = 0
14: var o_count = 0
15: for all (idx_l, idx_r) in Segments do
16:     var R = LocateROM(S(idx_l ··· idx_r), ω_min, ω_inc, k)
17:     var weight = idx_r − idx_l + 1
18:     skew_sum = skew_sum + R.skew ∗ weight
19:     o_count = o_count + weight
20: end for
21: return skew_sum/o_count
22:
23: function FindSeg(S, idx_l, idx_r, inc, R, k)
24:     loop
25:         if (idx_r + inc > S.length) then
26:             return (idx_l, idx_r)
27:         end if
28:         var bounded = 0
29:         for (i = idx_r + 1; i ≤ idx_r + inc; i++) do
30:             var (t, o) = S(i)
31:             var ρ = t ∗ cos θ + o ∗ sin θ
32:             if (ρ ≥ R.lb and ρ ≤ R.lb + R.ω) then
33:                 bounded = bounded + 1
34:             end if
35:         end for
36:         if (bounded ≥ k) then
37:             idx_r = idx_r + inc
38:         else
39:             return (idx_l, idx_r)
40:         end if
41:     end loop
42: end function
```

first segment, the *segment base*. Although a jump always happens at some specific location, a point-by-point scanning approach does not work with the interference of outliers. Instead, a jump can always be detected when the ROM of later offsets differs from that of earlier offsets. Therefore, an effective method is to test per batch of later offsets if the ROM of the segment base covers the majority. After a jump is detected, the algorithm repeats the first step by determining the next segment.

In this work, a method called DROML is developed to measure the skew of multi-segment offset-sets. DROML is designed based on the above idea, and is able to work in real-time. The pseudocode of DROML is shown in Algorithm 2, and its parameters along with important variables are listed in Table 1, with short explanations. Starting from the first few hundred points, DROML calls LocateROM() function and tests if the thickness of the returned ROM exceeds the threshold ω_{max}. If this is the case, then the current sampled seg_{min} points are considered not stable enough to serve as a segment base. DROML then slides the sampling range by a number *inc*, i.e., from $1 \cdots seg_{min}$ to $(1 + inc) \cdots (seg_{min} + inc)$, and calls LocateROM() again. This step repeats until a segment base is located.

The next step is to extend the right-hand side of the segment base until a jump is encountered. This is performed by function FindSeg() in lines 23–42. It receives the segment base in the form of the beginning index *idx_l* and the ending index *idx_r* in S, and tries to increase *idx_r* in the *inc* unit if the later offsets pass the continuity test. A fast way to do this is to test if there are enough points within the same distance range of the ROM for every *inc* points. A smaller *inc* gives higher resolution boundary detection, but the number should be kept to at least a few dozen, or a burst of outliers could easily cause a false positive. If the test is passed, *idx_r* is increased by *inc* and FindSeg() continues to test the next *inc* points. Otherwise, it is assumed that the boundary of this segment has been reached, and FindSeg() returns the current (*idx_l*, *idx_r*) pair. DROML then starts locating the next segment base from where a jump occurs.

The whole algorithm stops when there is no more *inc* offset to slide to, or to extend.

In addition to k, another new threshold k_{strict} introduced in DROML. As a larger threshold than k, it is passed to LocateROM() for finding the ROM of a segment base. This design results from considerations for practical use. Since LocateROM() always returns the thinnest ROM, the number of bounded points might only marginally exceed the threshold. Therefore, it is possible that some *inc* points, though belonging to the same segment theoretically, marginally fail to pass the continuity test in FindSeg() if the same threshold is used. To compensate for this kind of error, a larger threshold k_{strict} is passed to LocateROM() to ensure a larger thickness, while the smaller threshold k is used in the continuity test.

As a ROM oriented approach, DROML is able to detect a jump on-the-fly and adaptively relocate the ROM. Furthermore, since LocateROM() is able to output a stable ROM with only a few hundred offsets [8], DROML is able to give a reasonable skew estimation soon after the measurement begins. Finally, the *inc* batch unit used in DROML might cause the segment boundary to be located imprecisely, and one segment might contain a few points which should belong to the adjacent segment. However, these *noise* points are treated as outliers of this segment, and thus have no effect on the ROM locating.

3.3 Estimating the Clock Skew

The pseudocode in lines 13–21 of DROML estimates the clock skew of the whole offsets-set by calculating the weighted mean of the skews of all segments. The weight of a segment here is the number of points covered in that segment. Note that instead of using the ROM of the segment base, the ROM of the whole segment is acquired by invoking LocateROM() again at line 16 for more precise estimation. Most often, the weighted mean provides convincing estimates. However, it should not be used when there is at least one segment whose skew differs from others over the tolerable range, e.g. [-1, 1] ppm. Instead, the measurement is considered invalid and a remeasurement is required.

4. Evaluation Results

Two datasets: Dataset-1 in Fig. 1 for the normal case, and Dataset-2 in Fig. 2(b) for the multi-segment case, were used to evaluate the performance and robustness of the proposed methods. These datasets were timestamps sent from a notebook of Microsoft Windows 7 OS, along with timestamps of receiving time recorded by another PC of Ubuntu 14.04 OS. Both computers were connected to the LAN in our laboratory. The 6000 timestamps in each dataset were sent at 500-ms interval. While Dataset-1 is taken from a constantly-connected wireless measurement, Dataset-2 is recorded in an environment with both wired and wireless network connections.

It is worth nothing that offsets, when recorded in a connection with heavy network traffic or a long relay route, tend to be scattered due to higher delay and jitter. However, the HT-based method is designed to trade off higher computation time for improved robustness in these situations [8], and so be its extension DROML.

The first evaluation demonstrates how stable the ROMs obtained by the modified HT-based method are. Lower skew fluctuation means greater stability and accuracy in this evaluation. Here, Dataset-1 was explored in two ways: 1) accumulated-offset scheme, where estimations are gradually conducted from 500 offsets to 6000 offsets with a 500 offset increment; 2) separated-offset scheme, where estimations are all conducted on small segments of 500 offsets. The results of the modified HT-based method are then compared with those obtained by LPA and by the original HT-based method. For each estimation, ω_{min} is set to 100 μs, ω_{inc} to 50 μs, and k to 50% of the offset count.

The next evaluation demonstrates how robust DROML is when estimating the multi-segment case. Here, seg_{min} is set to 500, k_{strict} is 60% of the offset count, k is 50% of the offset count, ω_{max} is 3 ms, and inc is 100. Even with totally different distributions, both datasets are taken from the same two devices, and thus the clock skews of both datasets should be very similar, if not of the same value. Therefore, the closer the estimation for Dataset-2 is to the estimation for Dataset-1, the higher the accuracy of the DROML method is.

4.1 Evaluating the Stability of ROMs

Table 2 summarizes the results of the accumulated-offset scheme for Dataset-1. The full-size estimations (6000 offsets) of the three methods, 53.07 ppm by LPA, 53.21 by the original HT-based method, and 53.1 ppm by the modified version were very close to each another, and were used as the references. Incidentally, most measured skews range from -200 to 200 ppm [8]. The skew stability of the three methods can be observed in the "Max − Min" row of Table 2. The maximal and minimal estimation values of the three methods occurred at the 500 and the 6000 rows in this evaluation. The fluctuation scale of the modified HT-based method was only 0.6 ppm, much smaller than the 1.07 ppm of the LPA method, and the 0.91 ppm of the original HT-based method with linear regression.

Table 2 Results of Accumulated Estimation

Offset	LPA (ppm)	Origin HT-based method			Modified method	
		ω (μs)	$\theta - \pi/2$ (ppm)	Skew by LR (ppm)	ω (μs)	$\theta - \pi/2$ (ppm)
500	54.14	500	54.2	54.02	300	53.7
1000	53.67	500	53.1	53.12	300	53.5
1500	53.56	500	53.7	53.51	300	53.4
2000	53.42	400	53.7	53.52	300	53.7
2500	53.34	300	53.3	53.11	300	53.3
3000	53.30	300	53.3	53.12	300	53.3
3500	53.23	300	53.2	53.18	300	53.2
4000	53.22	300	53.2	53.26	250	53.2
4500	53.18	300	53.1	53.12	300	53.1
5000	53.13	300	53.3	53.22	250	53.1
5500	53.09	300	53.1	53.21	300	53.1
6000	53.07	400	53.2	53.21	300	53.1
Max	54.11		54.2	54.02		53.7
Min	53.07		53.1	53.11		53.1
Average	53.36		53.37	53.3		53.3
Max − Min	1.07		1.1	0.91		0.6

Table 3 Results of Separate Skew Estimation

Range of offset	LPA (ppm)	Origin HT-based method			Modified method	
		ω (μs)	$\theta - \pi/2$ (ppm)	Skew by LR (ppm)	ω (μs)	$\theta - \pi/2$ (ppm)
1–500	54.14	500	54.2	54.02	300	53.7
501–1000	53.49	400	53.1	53.12	300	53.7
1001–1500	53.58	500	54.5	54.21	300	53.5
1501–2000	53.12	400	54.3	54.31	300	53.4
2001–2500	52.79	400	53.1	53.12	300	53.3
2501–3000	52.86	300	53.3	53.29	250	53.3
3001–3500	52.71	300	53.2	53.16	300	53.2
3501–4000	52.98	300	53.2	53.21	300	53.2
4001–4500	53.14	300	53.2	53.18	250	53.2
4501–5000	52.77	300	53.1	53.12	250	53.1
5001–5500	52.69	400	53	53.08	300	53.1
5501–6000	52.77	300	53.1	53.12	300	53.1
Max	54.14		54.5	54.31		53.7
Min	52.69		53	53.08		53.1
Average	53.09		53.4	53.41		53.3
Max − Min	1.45		1.5	1.23		0.6

Another comparison of separate skew estimations of 500-offset batches is shown in Table 3. As expected, the skew fluctuation of the LPA increased from 1.07 ppm to 1.45 ppm, and similarly the precision of the original HT-based method decreased from 0.91 ppm to 1.23 ppm. Surprisingly, the modified HT-based method maintained its precision at 0.6 ppm. It was therefore concluded that the ROM of segment base returned by function LocateROM() in Algorithm 2 is stable enough to bound the whole segment.

Finally, Tables 2 and 3 also showed that the thickness ω of the ROM obtained by the modified HT-based method was less than that of the original version in most cases. Given the same threshold k, a smaller ROM has higher density, and thus results in higher precision.

Table 4 Result of Estimation on a Multi-segment Offset-set

Segment number	Segment base			Whole segment		
	Range	$\theta - \pi/2$ (ppm)	ω (μs)	Range	$\theta - \pi/2$ (ppm)	ω (μs)
1	1–500	53.3	100	1–1500	53.2	100
2	1501–2000	53.8	900	1501–3000	53.4	300
3	3001–3500	53.2	100	3001–4500	53.2	100
4	4501–5000	53.7	1000	4501–5900	53.5	300
Total				1–5900	53.32	

Table 5 Comparison Between LPA, HT-based Method, and DROML

Measured segment	LPA		HT-based method			DROML	
	skew (ppm)	time (s)	skew (ppm)	time (s)	ω (μs)	skew (ppm)	time (s)
1	53.59	0.9	53.4	2.1	300	53.2	1.24
1 and 2	62.98	1.02	60.27	44	3200	53.3	4.58
1, 2, and 3	56.37	1.75	50.67	45	1600	53.27	5.85
1, 2, 3, and 4	56.37	2.6	56.37	60	4000	53.32	9.15
Max - Min	9.39		9.6			0.12	

Table 6 Value of ω as a Function of Jump Scale

Jump scale (ms)	$\theta - \pi/2$ (ppm)	ω (μs)	Distance (μs)	Count
6	94.5	4100	-31612	306
5.5	91.9	3700	-28992	303
5	88.7	3200	-26122	305
4.5	86.1	3050	-23518	303
4	82.6	2700	-20101	305
3.5	79.1	2550	-18165	302
3	75.4	2200	-15611	303
2.5	71.3	1950	-13281	310
2	69.6	1700	-10662	311
1.5	65.7	1350	-8142	312
1	60.2	1000	-5424	320

4.2 Evaluating DROML on the Multi-segment Case

Table 4 details the results when DROML was run on Dataset-2. DROML was able to correctly detect the four segments in the dataset. Since 25 packets were lost in Dataset-2, the offset count was 5975. As DROML was set to proceed per 100 offsets, the last 75 offsets were not used in the estimation. Thus, the 4^{th} segment ended at index 5900. Also note that the thicknesses $R.\omega$ of the ROMs in the 1^{st} and 3^{rd} segments are both 100 μs, while the thicknesses in the 2^{nd} and 4^{th} segments are both 300 μs. The pattern of thickness reflects how the offset distribution in each segment changes in order.

The last row of Table 4 gives a global skew estimation of 53.25 ppm. This value, compared with the 53.1 ppm in a normal case as a reference, contains only a 0.22 ppm error. These results show that DROML is robust in handling multi-segment cases.

4.3 Discussion

This subsection will discuss a few subordinate issues of DROML, completing the scope of this study.

4.3.1 Computation Time and Time Complexity

In order to compare DROML with LPA and the HT-based methods when estimating the multi-segment case, Table 5 shows the computation times and estimation results of these methods. The computation was executed by a PC with an Intel Core-i7 processor and 2 GHz RAM. It is not surprising that DROML, when handling wireless segments, took more time than LPA. However, the Max-Min row in Table 5 shows that DROML is far more stable when compared with LPA. Meanwhile, due to the extra-large thicknesses of the ROMs, the HT-based method required a much greater computation time, with a 9.6 ppm fluctuation.

Since DROML can save the clock skew of any segment it has collected, the computation time to generate the estimation depends mainly on the offset count of the current segment when DROML runs in online mode. The most time consuming step in Algorithm 2 is the LocateROM() function, and its time complexity can be easily derived from Algorithm 1. Let $n = |S|, c_\omega = \frac{R.\omega - \omega_{min} + 1}{\omega_{inc}}, c_\theta = \frac{\theta_{max} - \theta_{min} + 1}{\theta_{inc}}$, then the time complexity is:

$$O(c_\omega c_\theta(an + bn\log n + cn)) = O(c_\omega n \log n) \quad (3)$$

where a, b and c are constants; an denotes the computation time at lines 6–9, $bn \log n$ is the sorting method time, cn is the time required by the **for** loop at line 11–20. c_ω may increase when the delay jitter is high, but the value of n can be bounded in a few thousand, which is sufficient for the modified HT-based method to give a stable and precise estimation.

4.3.2 Online DROML

The pseudocode in Algorithm 2 is an off-line version of DROML. An online DROML measurer should behave as follows. First, it is idle, waiting for sufficient timestamps of the first segment base. When there are enough timestamps, or offsets, DROML automatically activates to locate the ROM of the segment base. Simultaneously, the measurer keeps running the timestamp collection process. When a segment base occurs, DROML is triggered every time inc timestamps are received. Finally, The measurer can invoke DROML to assess global skew estimation at any time. DROML, when called, uses the currently found segments to calculate the weighted mean.

As every offset contains an error of variant value, it is the nature of clock skew measurement to process offsets in batches. However, the online DROML is able to feedback the estimated skew during the measuring process with a reasonably short response time, we thus argue that it works in real-time.

4.3.3 The ω_{max} Threshold

An appropriate value of ω_{max} prevents DROML from incorrectly recognizing an invalid range as a segment base. Outside of high-jitter cases, it is interesting to note how a jump may affect the ROM thickness, as shown in Fig. 2(a).

Since LocateROM() always finds the thinnest region with at least k_{strict} points inside, the longer half segment will become the majority. However, LocateROM() fails to output a valid ROM if the jump occurs around the middle of the range.

Now consider a half-and-half case, with its first half from the last 250 offsets of the 1^{st} segment in Fig. 2(b), and its second half from the beginning 250 offsets of the 2^{nd} segment. The ROM of these 500 offsets by LocateROM() is shown in the first row of Table 6. In this case, ω_{max} has to be set as smaller than its thickness, 4100 μs, to prevent a false negative. According to the computation results in Table 4, the distance of the lower boundary line $R.\rho$ of the 1^{st} segment is -60 μs, while $R.\rho$ of the 2^{nd} segment is 6021 μs. Thus, the jump scale between these two segments is about 6 ms. Therefore, 4100 μs serves as a reference for DROML to abandon any range containing a jump of at least 6 ms in the middle.

By manually lowering every offset in the 2^{nd} segment, more ω_{max} references may be derived for smaller jump scales. Table 6 gives the computation results of jump scales from 6 ms down to 1 ms. The thickness in the 1 ms case is the same as the thickness in the 4^{th} segment base in Table 4, and thus a jump of scale 1 ms or smaller is not detectable by DROML.

5. Conclusions

This paper proposed the DROML method to extend skew measurement to multi-segment offset-sets. DROML is realized by an improved HT-based method, and can thus provide stable estimation, even for sets as small as 500 timestamps. As the core concept of DROML, ROM is utilized not only for estimating the skew, but also for detecting jumps, which are responsible for the multi-segment phenomenon. As the robustness and accuracy of DROML have been proven through experiments on both normal and multi-segment cases, it can be concluded that DROML, an HT-based clock skew method with abilities to provide stable ROMs and to handle dynamically the multi-segment problem, is a complete, robust method for estimating the clock skew of devices over network connections.

Acknowledgments

This work was supported by grants from the Ministry of Science and Technology, Taiwan (MOST 104-2923-E-011-005-MY3 and MOST 104-2221-E-011-070).

References

[1] S.B. Moon, P. Skelly, and D. Towsley, "Estimation and removal of clock skew from network delay measurements," Proc. INFOCOM Conf., pp.227–234, 1999. DOI: 10.1109/INFCOM.1999.749287.
[2] V. Paxson, "On calibrating measurements of packet transit times," Proc. ACM SIGMETRICS Conf., pp.11–21, 1998. DOI: 10.1145/277851.277865.
[3] 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/TDSC.2005.26.
[4] S. Jana and S. Kasera, "On fast and fccurate detection of unauthorized wireless access points using clock skews," IEEE Trans. Mobile Comput., vol.9, no.3, pp.449–462, March 2010. DOI: 10.1109/TMC.2009.145.
[5] S. Sharma, A. Hussain, and H. Saran, "Experience with heterogenous clock-skew based device fingerprinting," Proc. LASER Workshop, pp.9–18, 2012. DOI: 10.1145/2379616.2379618.
[6] C. Arackaparambil and S. Bratus, "On the reliability of wireless fingerprinting using clock skews," Proc. 3rd ACM Conf. Wireless Network Security, pp.169–174, 2010. DOI: 10.1145/1741866.1741894.
[7] M. Cristea and B. Groza, "Fingerprinting smartphones remotely via ICMP timestamps," IEEE Commun. Lett., vol.17, no.6, pp.1081–1083, June 2013. DOI: 10.1109/LCOMM.2013.040913.130419.
[8] K. Oka 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.
[9] 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," Proc. 26th IEEE Int. Conf. Advanced Inform. Networking and Applicat. (AINA), pp.526–533, March 2012. DOI: 10.1109/AINA.2012.51.
[10] L. Polcák, J. Jirásek, and P. Matousek, "Comment on "Remote Physical Device Fingerprinting"," IEEE Trans. Dependable and Secure Computing, vol.11, pp.494–496, 2014. DOI: 10.1109/TDSC.2013.26.
[11] K. Zeng, K. Govindan, and P. Mohapatra, "Non-cryptographic authentication and identification in wireless networks [Security and Privacy in Emerging Wireless Networks]," IEEE Wireless Commun., vol.17, no.5, pp.56–62, 2010. DOI: 10.1109/MWC.2010.5601959.
[12] 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: 10.4218/etrij.10.0109.0611.
[13] 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.
[14] P.E. 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.
[15] D.-J. Huang, W.-C. Teng, and K.-T. Yang, "Secured flooding time synchronization protocol with moderator," Int. J. Commun. Syst., vol.26, no.9, pp.1092–1115, 2013. DOI: 10.1002/dac.
[16] 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, March 2014. DOI: 10.1016/j.jnca.2013.04.003.
[17] P. Tipwai and S. Madarasmi, "A modified generalized Hough transform for image search," IEICE Trans. Inf. & Syst., vol.E90-D, no.1, pp.165–172, Jan. 2007. DOI: 10.1093/ietisy/e90-d.1.165.
[18] S. Li, "Edge point grouping for line detection," IEICE Trans. Inf. & Syst., vol.E95-D, no.6, pp.1713–1716, June 2012. DOI: 10.1587/transif.E95.D.1713.
[19] S. Fujikawa, "Network clock system that ensures a high level of frequency accuracy," IEICE Trans. Commun., vol.E98-B, no.11, pp.2212–2226, Nov. 2015. DOI: 10.158/transcom.E98.B.2212.

Komang Oka Saputra received his B.Eng. degree from the Brawijaya University, Malang, Indonesia, in 2004, and his M.Eng. degree from the University of Indonesia, Depok, 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, 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 communication network and security.

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.

Takaaki Nara received his Doctor of Engineering degree in 2001 from the University of Tokyo. He is Associate Professor of Graduate School of Information Science and Technology at the University of Tokyo. His research interests include algorithms and sensors for inverse problems.