单向延迟测量中时钟动态性检测算法*

王俊峰[1+], 杨建华[2], 周虹霞[3], 谢高岗[2], 周明天[1]

[1](电子科技大学 计算机科学与工程学院,四川 成都 610054)

[2](中国科学院 计算技术研究所 信息网络研究室,北京 100080)

[3](电子科技大学 电子工程学院,四川 成都 610054)

Detecting Clock Dynamics in One-Way Delay Measurement

WANG Jun-Feng[1+], YANG Jian-Hua[2], ZHOU Hong-Xia[3], XIE Gao-Gang[2], ZHOU Ming-Tian[1]

[1](College of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China)

[2](Network Research Division, Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100080, China)

[3](College of Electronic Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China)

+ Corresponding author: Phn: +86-28-83203300, E-mail: cswangjf@std.uestc.edu.cn

Received 2003-02-24; Accepted 2003-06-18

Wang JF, Yang JH, Zhou HX, Xie GG, Zhou MT. Detecting clock dynamics in one-way delay measurement. *Journal of Software*, 2004,15(4):584~593.

http://www.jos.org.cn/1000-9825/15/584.htm

Abstract: A key issue in one-way delay measurement is the removal of relative clock offset in the situation of without external clock synchronization mechanisms for the end-to-end hosts. Most researches are based on the assumption that the clock skew retains constant and without clock adjustments and drifts during measurement. But in fact, it is found that end system clock might be subject to gradual or instantaneous clock adjustments and frequency adjustments in operation. In this paper, with the time series segmentation technology, we discuss the detection of clock dynamics in one-way delay measurement. Two algorithms are proposed to estimate the relative clock offset in post facto and on-line mode respectively, while with only unidirectional probe packets. The

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2002AA121032 (国家高技术研究发展计划(863)); the Institute of Computing Technology Youth Fund of China under Grant No.20026180-14 (计算技术研究所青年基金)

WANG Jun-Feng was born in 1976. He is a Ph.D. candidate at the College of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include high speed network measurement, protocol test and analysis. **YANG Jian-Hua** was born in 1978. She is a Ph.D. candidate at the Institute of Computing Technology, the Chinese Academy of Sciences. Her research interests include high speed network measurement and monitoring. **ZHOU Hong-Xia** was born in 1976. She is a postgraduate student of College of Electronic Engineering, University of Electronic Science and Technology of China. Her current research is focused on digital signal processing. **XIE Gao-Gang** was born in 1974. He is an associate professor at the Institute of Computing Technology, the Chinese Academy of Sciences. His research is focused on high speed network technology, network measurement and monitoring, QoS. **ZHOU Ming-Tian** was born in 1939. He is a professor and doctoral supervisor at the College of Computer Science and Engineering, University of Electronic Science and Technology of China. His current research interests include computer networking information system, open distributed processing system, computer system and software, and computer supported collaboration work.

computational complexity of the post facto algorithm is of order $O(N^2)$. Experiments show that these algorithms can provide reasonable clock dynamics detection and informative one-way delay estimation.

Key words: one-way metrics; clock dynamics; network measurement; time series segmentation

摘 要: 延迟是评价网络性能的重要指标,也是进行其他网络性能指标测量的基础.基于全球定位系统(GPS)的端到端(end-to-end)时钟同步是测量网络单向指标的常用方法,但是其代价昂贵且缺乏灵活性.在无端到端时钟同步机制下进行网络单向延迟指标测量的关键是消除时钟偏差效应的影响.基于对时间序列分段技术的分析,提出了一种新的时间序列分段标准与改进的分段算法,实现序列的自动聚类,其时间复杂度为 $O(N^2)$.将该算法应用于检测端到端时钟的动态性,识别测量过程中时钟跳变和时钟频率调整位置,实现对网络单向延迟的测量,弱化了同类工作中对时钟动态性的严格假设.同时提出了基于滑动窗的在线实时时钟动态性检测算法.实际测试实验表明,该算法是行之有效的.

关键词: 单向指标;时钟动态性;网络测量;时间序列分段

中图法分类号: TP393 **文献标识码**: A

Delay metrics are important indicators in evaluating network performance and also the basis for many other metrics measurements. As more and more quality of service (QoS) sensitive applications have been emerging, which presses heavy demands on the performance of the underlying network infrastructure, Internet Service Providers (ISPs) must hold an instant insight view of networks to provide customers with Service Level Agreement (SLA). One-Way delay (OWD) metric has been attracting much attention by international organizations and researchers in recent years. IETF IP Performance Metrics Working Group (IPPM) has produced several RFCs that are related to OWD, and more drafts are also proposed[1~4]. When there is no external clock synchronization mechanism adopted, a key issue in OWD measurement is how to remove the relative clock offset arising from the different clock frequencies in different systems.

Although many skew estimation and removal algorithms have been proposed, nearly all of them are based on the same assumption that the clocks retain constant skew and with no clock adjustments in measurement[5~8]. Paxson pointed out that computer clocks are sometimes subject to gradual or instantaneous adjustments[5]. In our experiments, it is found that the clock frequency might undergo abrupt adjustment in minutes to day's time scale and the frequency retains constant between two consecutive clock/frequency adjustments in addition to clock adjustments (e.g. clock notation adjustments). Thus, those previously proposed algorithms are not suitable for the removal of relative clock offset and OWD estimation.

Paxson developed an algorithm for detecting the clock adjustments and estimating the clock skew through bidirectional measurements[9]. The central notion of the method is that the signature of the one-way transit time in each direction shows equal but opposite level of shifts. The drawbacks of the algorithm are also discussed in Ref.[9]. The skew estimation algorithm supposes that the forward and reverse one-way delays experienced by the probe packets appear equal but opposite trends. As the variations of bandwidth resource allocation algorithms, queuing policies in routers and switches, and different traffic characterizations, this assumption seldom holds in current Internet. In this paper, we divide the clock dynamics into three types: clock adjustments, frequency adjustments and clock drift, then propose a segmentation-based clock dynamics detection algorithm. It can remove the relative clock offset automatically with only unidirectional measurement. We also prove that the algorithm is suitable for online OWD estimation.

The rest of paper is organized as follows. Section 1 presents the experimental design in our practical measurements. Section 2 introduces the time series segmentation algorithm. In Section 3, we present the segmentation-based clock adjustments detection and skew estimation. Section 4 proposes an online version of this

algorithm. We conclude the paper in Section 5.

1 Data Acquisition

The OWD trace is obtained using the developed tool under Linux OS called *capOWD*, which consists of two components. One resides on the source end system sending UDP probe packets periodically while the other on the destination receiving these probes. The detailed structure of probe packet is outlined in Ref.[10].

Let $t^s(i)$ and $t^r(i)$ $(i=0,1,2,...)$ denote the timestamp that the *i-th* probe packet is sent and received according to its local end system clock. The Raw One-way Delay (ROWD) is defined as $\Delta t(i)=t^r(i)-t^s(i)$. Figure 1 shows the evolutions of ROWD with respect to the probe sequence in a trace. The interval between successive probes is one second, the packet length is 1 500 bytes, including IP header, and the lost probes are not shown.

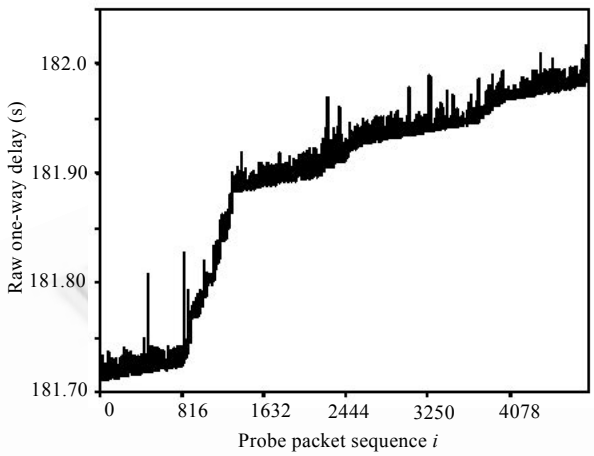

Fig.1 Evolutions of raw one-way delay

From Fig.1, it is clear that not only the two end systems exist initial clock offset in measurement, but also at least one clock frequency experiences several times of variations in merely less than 1.5 hours. The time series plot reveals that the clock skew remains nearly stable, e.g. the clock drift is neglectable between two successive clock adjustments or frequency adjustments. The piecewise character of the plot discloses the nature of ROWD. The core of the clock dynamics detection algorithm in the paper is to segment the measured ROWD at proper positions by which we could find when clock/frequency adjustments occur. For each segment, different clock skew estimation algorithms could be used to remove the clock skew and calculate the relative OWD variations.

2 Time Series Segmentation Algorithm

Time series segmentation is a simple instance of cluster analysis that is widely used in data mining[11~13]. Essentially, it is a high level representation of time series and an unsupervised classification mechanism to find internally homogeneous segments in a dataset. In the paper, we are concerned on locating the stable period of clock activities and identifying the change points where clock/frequency adjustments occur.

2.1 Time series segmentation definition

Definition 1. Let $T=\{x(i)|1\le i\le N\}$ be a time series with N samples, and $s_T(a,b)=\{x(i)|a\le i\le b\}$ indicate a segment of T consisting of non-empty consecutive samples from $x(a)$ to $x(b)$. A *k-segmentation* S_T^k, $k\le N$ of T is a partition of T to k non-empty and non-overlapping segments

$$S_T^k=\{s_T^i(a_i,b_i)\,|\,1\le i\le k\} \tag{1}$$

where $a_1=1, b_k=N$ and $a_i=b_{i-1}+1, (1<i\le k)$. For simplicity, let s_k^i denote the i-th segment of *k-segmentation* of time series T.

Two operators, *split* and *merge* are defined as: the *split*($s(a,b)$) procedure splits a given segment $s(a,b)$ into two consecutive segments $s(a,j)$ and $s(j+1,b)$ at point j while the *merge* procedure takes a converse operation as *split*. It concatenates two consecutive segments into one segment.

To obtain an optimized *k-segmentation* of T, a cost function $COST(s_k^i)$ is associated with each possible segment s_k^i and a function $COST(S|k)$ is related to time series T. Commonly, $COST(S|k)$ is defined as the sum of cost of each possible *k-segmentation*:

$$COST(S|k)=\sum_{i=1}^{k}\text{cost}\left(s_k^i\right) \tag{2}$$

then, the optimized *k-segmentation* of T is such that minimizes $COST(S|k)$ in all possible *k-segmentations*.

There have been a number of segment cost functions proposed, and the widely used one is the K-means as in Ref.[13]:

$$\text{cost}_{\text{KM}}(s_k^i)=\frac{1}{b_i-a_i+1}\sum_{j=a_i}^{b_i}[x(j)]^2-\left[\frac{1}{b_i-a_i+1}\sum_{j=a_i}^{b_i}x(j)\right]^2 \tag{3}$$

the cost function (which is to be minimized) is the sum of the squared error between each sample and its assigned segment center.

This segment cost function is not much suitable for our clock dynamics detection purpose, because it does not take any characters of the ROWD time series into consideration. The internal homogeneous segments are those that the samples in the segment present a stable clock skew with no clock adjustment or frequency adjustment occurred. Before developing a new cost function for a segment, we first define a Minimal Expectation Line (MEL) for each possible segment:

$$MEL(s_k^i)=\sigma_i x+\mu_i \tag{4}$$

where σ_i and μ_i are parameters subject to

$$\begin{cases}\min\left\{\sum_{j=a_i}^{b_i}[x(j)-(\sigma_i * j+\mu_i)]\right\}\\ x(j)-(\sigma_i * j+\mu_i)\ge 0,\quad a_i\le j\le b_i\end{cases} \tag{5}$$

Let $MEL(j|s_k^i)$ be the minimal expectation at point j in segment s_k^i. The new cost function is defined as:

$$\text{cost}_{\text{MEL}}\left(s_k^i\right)=\sum_{j=a_i}^{b_i}\left[x(j)-MEL\left(j|s_k^i\right)\right] \tag{6}$$

Thus, if we get an optimized *k-segmentation* of T, then it is easy to estimate the clock skew backward from Eq.(5) as described in next section.

2.2 Segmentation algorithm

Keogh in Ref.[14] reviews the three major time series segmentation approaches in various application fields. The Top-Down Algorithm is straightforward. It splits T optimally into two segments, then every segment is further segmented at optimal points as candidate segments, the partition that has minimal *COST* is accepted for next round of splitting and this procedure continues until k segments are generated. The limitation of the method is that once a break point is determined, it will not be changed or replaced in latter round of splitting even if it is proven that it is not an optimal point (weak point) from a global point of view.

We introduce a *k-segmentation* Optimized Top-Down Algorithm (K-OTDA) to address this drawback. In each round, one new segment is generated, thus it must take k–1 rounds to split the *k-segmentation* time series. Assuming in round $m, m<k$, splitting segment $s_m^i(a_i,b_i)$ at position j into $s(a_i,j)$ and $s(j+1,b_i)$ gets the minimized *COST*, we assume that the partition points that are around s_m^i have higher probability of weak optimization. Then the following two procedures are taken to relocate these weak partition points.

$split(merge(s(a_{i-1},b_{i-1}),s(a_i,j)))$, $i-1\ge 1$ and $split(merge(s(j+1,b_i),s(a_{i+1},b_{i+1})))$, $i+1\le k$

1. PROC K-OTDA (T,k)
2. FOR $i=2$ to k
3. FOR $j=1$ to $j-1$
4. split $s_{i-1}^{j}(a_j,b_j)$ at point p_j that minimizes: $C_j=\text{cost}_{\text{MEL}}(s_i^j(a_j,p_j))+\text{cost}_{\text{MEL}}(s_i^{j+1}(p_j+1,b_j))$
5. END FOR // end FOR for j
6. choose the segment $s_{i-1}^{l}, l\le i-1$ that has the minimal C_l, set $p=p_l$ as new partition point
7. invoke $split(merge(s_{i-1}^{l-1}(a_{l-1},b_{l-1}),s_{i-1}^{l}(a_l,p)))$ and $split(merge(s_{i-1}^{l}(p,b_l),s_{i-1}^{l+1}(a_{l+1},b_{l+1})))$
8. END FOR // end FOR for i
9. END PROC

Fig.2 Pseudo code for optimized top-down segmentation algorithm

The pseudo code for K-OTDA is illustrated in Fig.2. Step 6 selects the segment s_{i-1}^{l} that has minimized C_l to ensure that $COST(T\mid i)$ is minimal. Himberg in Ref.[13] proposed two Top-Down based greedy algorithms LIR and GIR to deal with the limitation of the original Top-Down algorithm. Our K-OTDA algorithm acts similar as GIR, but because no random factor is introduced as done in GIR or LIR, K-OTDA is more deterministic. A Linear Programming Algorithm (LPA) can calculate Eq.(4) in linear time[6]. Commonly $k<<N$ the computational complexity of K-OTDA is $O(N^2)$.

2.3 Algorithm performance evaluation

In practice, it is sometimes hard to know the true number of segments and its partitions. This subsection evaluates K-OTDA's performance against Top-Down Algorithm's with different cost functions cost_{MEL} and cost_{KM} when the number of segments is known. In next subsection, K-OTDA is extended to estimate the number of segments. A 4000-sample data set with 11-segmentation is generated, as shown in Fig.3.

The start point of the *i*-th segment is shown in the "TRUE" column in Table 1. It shows that the Optimized Top-Down *k*-segmentations Algorithm with the MEL cost function (KOTD-MEL) performs partition perfectly well while others might split the time series at unreasonable positions if the time series is noisy.

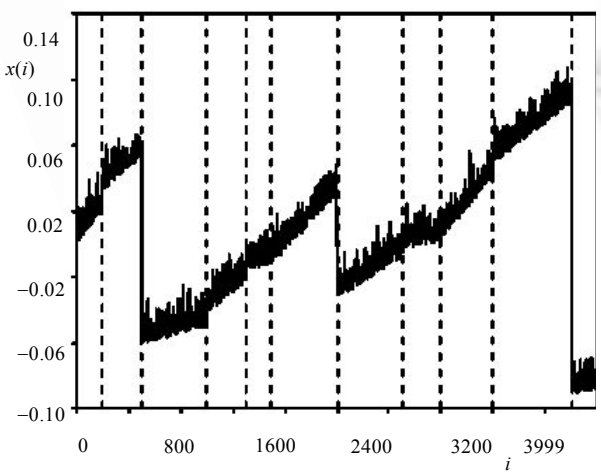

Fig.3 Generated 11-segmentation time series

Table 1 Performance comparisons with different cost functions and different segmentation algorithms

Seg	TRUE	KOTD-MEL	KOTD-KM	TD-MEL	TD-KM
1	0	0	0	0	0
2	200	200	200	200	200
3	500	500	500	500	500
4	1 000	1 000	1 109	997	1 000
5	1 300	1 300	1 637	1 054	1 247
6	1 500	1 489	2 000	1 300	1 672
7	2 000	2 000	2 410	1 491	2 000
8	2 500	2 500	2 929	2 000	2 800
9	2 800	2 800	3 198	2 724	3 024
10	3 200	3 200	3 480	3 200	3 205
11	3 800	3 800	3 800	3 800	3 800

2.4 Estimation of the number of segments

In most situations, we often do not know how many segments that underlie a time series. Estimating the true

number of segments is still an open problem. Tibshirani proposed a method called Gap statistic to estimate the number of clusters in a dataset[15]. It requires multiple reference datasets to compute the Gap statistic. In Ref.[16], Vasko developed the *Pete* algorithm to infer the number of segments in time series. It generates a large number of permutations based on the original time series dataset to perform permutation tests. These methods do not meet our requirements in detecting the clock dynamics, because only by providing an online detecting mechanism, will it be valuable for ISPs or customers to evaluate network performance. In the paper, we use only one reference dataset to estimate the number of segments.

Let T' be the carefully chosen permutation of time series T. The criterion for $T'$is that it should conceal the homogeneous of T as much as possible. The natural method is to choose samples randomly from T to generate another time series T'. Let ratio $r(T\,|\,k)$ be:

$$r(T\,|\,k)=\frac{COST(T\,|\,k)}{COST(T\,|\,k-1)},\ k>1$$

and $r(T\,|\,1)=1$. Vasko analyzed the evolutions of $COST(T\,|\,k-1)-COST(T\,|\,k)$ with the increase of segments k and pointed out that the decrease of cost is mainly due to the effect of over-segmented if k grows larger than the true segments.

$COST(T'\,|\,k-1)-COST(T'\,|\,k)$ could provide a reference for the cost decrease trend of T. The simple algorithm for estimating the segments is described in Fig.4. It will stop splitting once the current $r(T\,|\,k)$ is larger than the minimal $r(T'\,|\,)$ and output $k-1$ as the estimated number of segments.

As the number of samples in T is finite, algorithm *OTDA* will stop in limited steps. Figure 5 shows the evolutions of T and T' as a function of the number of segments k ($r(T\,|\,k), k\le 11$, is outside the visible area) using the artificial data set generated in Section 2.3. The estimated number of segments is 11.

1. PROC OTDA: IN (T) ,OUT (k)
2. generate T' from $T, k=1, \min\_r(T'\,|\,)=\infty$
3. WHILE $(r(T\,|\,k)\le \min\_r(T'\,|\,))$
4. K-OTDA$(T,k+1)$; K-OTDA$(T',k+1)$
5. k++; $\min\_r(T'\,|\,)=\min(\min\_r(T'\,|\,), r(T'\,|\,k))$
6. END WHILE
7. $k=k-1$
8. END PROC

Fig.4 Estimation of the number of segments with one permutation as reference

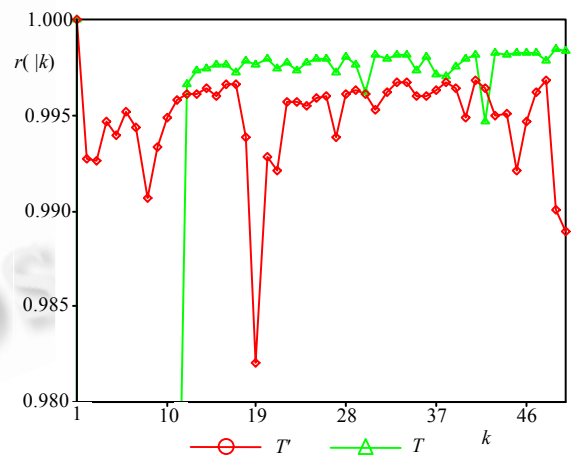

Fig.5 $r(\ |k)$ as a function of the number of segments

3 Detecting Clock Dynamics

Section 2 details the segmentation algorithm of time series. In this section, we apply it to real measured dataset from network OWD measurements. The measured dataset is shown in Fig.1 of Section 2.

Reference [17] proposes three properties: clock offset, skew and drift to describe clock dynamics. Moon defines a clock as a piecewise continuous function that is twice differentiable except on a finite set of points[6]. In fact, a clock is not a piecewise continuous function in most situations because clock adjustments will occur if a *cron*-like task is executed periodically or a Global Positioning System (GPS) is installed to synchronize clock

notation with an external reference clock for some tasks. Thus, those de-skew algorithms based on the assumption that there are no clock adjustments in OWD measurement are not practical. This paper deals with the following two aspects of clock dynamics: detecting clock adjustments that cause the uncontinuity of a clock notation, estimating and removing the skew between two clocks. We here use the same assumption as that of other literatures that no clock drift exists.

To be consistent with the previous work, the nomenclature used in the paper is the same as in Ref.[6]. Figure 1 is transformed to Fig.6 for analyzing without loss of generality.

A two-step procedure is employed to detect clock dynamics: firstly, the segmentation algorithm proposed in section 2 is used to fragment the relative OWD time series into an optimal number of segments by which the clock dynamics decides whether clock adjustments or frequency adjustments can be detected. Then, in each segment, we assume there are no clock adjustments, the increase or decease quantity of offset is only the results of the clock skew between two end hosts. Therefore, any previously developed skew estimation algorithms can be taken to remove this skew effect and obtain the true relative OWD.

Moon provided a novel method referred to as Linear Programming Algorithm (LPA) to estimate and remove the clock skew in the OWD measurement. We also adopt this LP algorithm in this paper. The objective function of the Linear Programming (LP) problem is to minimize the sum of the distances between the reference line and the relative OWD:

$$\min\{\sum_{j=b_i-N_i+1}^{b_i}[d_j-\alpha_i t_j^r+\beta_i]\} \quad \text{subject to} \quad d_j-\alpha_i t_j^r+\beta_i\geq 0, 0\leq j\leq N_i-1 \tag{7}$$

where d_j is the measured OWD relative to the 0-th probe packet, N_i represents the number of samples in segment s_k^i and b_i is the end point of the segment. From Eqs.(7) and (5), we can conclude that if the measured OWD time series has been spitted into k optimal segments, then $\alpha_i=\sigma_i, \beta_i=\mu_i$, thereby, the estimation of clock skew of the i-th segment α_i can be obtained directly from σ_i.

Figure 6 shows the 14-segmentation of the relative OWD time series through OTDA. We infer that there are 13 times of clock dynamics happened during measurement. Table 2 illustrates the start point and the end point for each segment. The uncontinuity of segment 1, 2 and 5, 6 are caused by probe packets' loss in measurement. Though the true segments are not known, Fig. 6 still shows that OTDA performs reasonable segmentation in measurement.

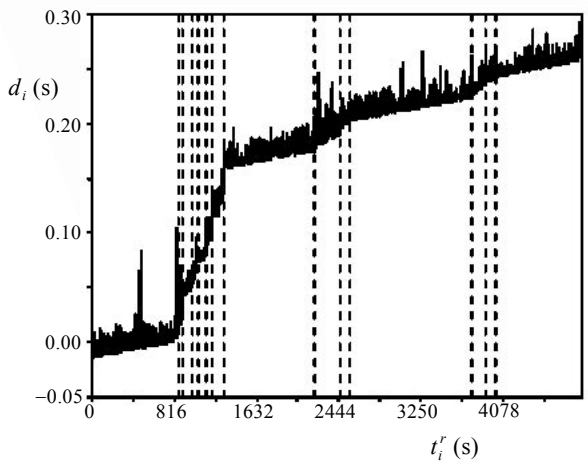

Fig.6 Relative OWD time series with 14 segments

Table 2 The start and end point for each segment

Seg.	Start	End
1	0	851
2	853	902
3	903	1 001
4	1 002	1 053
5	1 054	1 129
6	1 131	1 200
7	1 201	1 307
8	1 308	2 198
9	2 199	2 458
10	2 459	2 557
11	2 558	3 774
12	3 775	3 902
13	3 903	3 994
14	3 995	4 859

4 Online Clock Dynamics Detection Algorithm

Previous section investigates the detection of clock dynamics after a large number of samples have been obtained, but this post facto analysis may not be practical in network monitoring in some situations, because the volumes of the stored data are in the order of gigabytes or terabytes and the management of this vast number of data is a heavy burden. Thus, an online detection algorithm which reflects the constant status of the network is more attractive. Based on the OTDA, this section introduces an online clock dynamics detection algorithm that is referred to as Sliding Window and Optimized Top-Down (SWOTD) to treat the drawbacks of the OTD algorithm.

4.1 The SWOTD segmentation algorithm

The SWOTD employs a sliding window SW with size w and the OTDA algorithm to fulfill the online detection. In fact, SW is a buffer of size w. The defined operator $move(SW,j)$ moves SW to right, producing j data points and incorporating into SW another j new data points.

The SWOTD works as follows: initially, SW loads w data points and the OTDA is applied to the dataset in the window for generating k segments. Then remove the most left j data points where j is the total number of data points of the most left k–1 segments, producing k–1 segments, and move SW to right with j, reading in another j data points. Finally, the OTDA algorithm is performed again to report another k–1 segments. This process of applying OTDA to SW, reporting segments, sliding and loading subsequence data points is repeated as long as the observed samples arrive. Figure 7 outlines this SWOTD algorithm. Obviously, SWOTD addresses the even lengths and redundant computation problems behind the SWAB algorithm proposed in Ref.[14].

1. PROC SWOTD: IN (T), OUT (S)
2. load w number of data points
3. OTDA(SW) and generate k segments
4. report left most $k-1$ segments
5. WHILE observed new data points
6. $move(SW, \sum_{i=1}^{k-1} | s_k^i |$
7. OTDA(SW) and generate k segments
8. report another left most $k-1$ segments
9. END WHILE
10. report the one segment remained in SW
11. END PROC

Fig.7 The sliding window and optimized top-down algorithm

4.2 Experimental results

We conduct another experiment between the same hosts as that in Section 1. Keogh suggests that the sliding window size w should be configured to contain 5 or 6 segments[14], but we find it results in no differences when the window size is set to a little more than the size that could contain the maximal length of a segment.

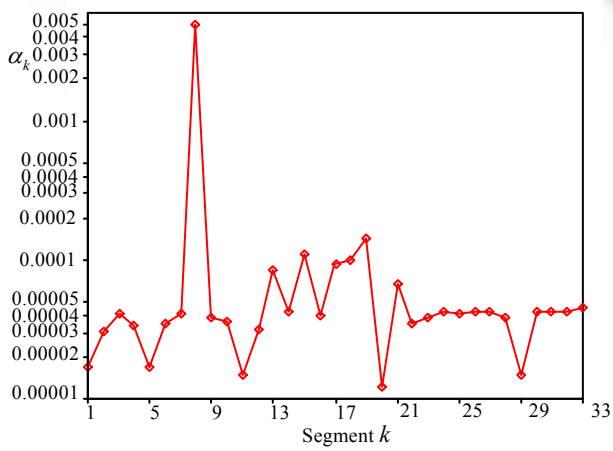

Fig.8 The clock skews variations in measurement

The variation of detected clock skew is shown in Fig.8, where sliding window size w=3000. For clarity, the y-axis is on logarithmic scale. Figure 9 shows the collected 12000 samples and the segmentations of the measured dataset. The SW moves five times to process the 12 000 samples. From the figure, SWOTD could detect not only clock adjustments but also frequency adjustments correctly. The calculated relative OWD after the skew removal is demonstrated in Fig.10.

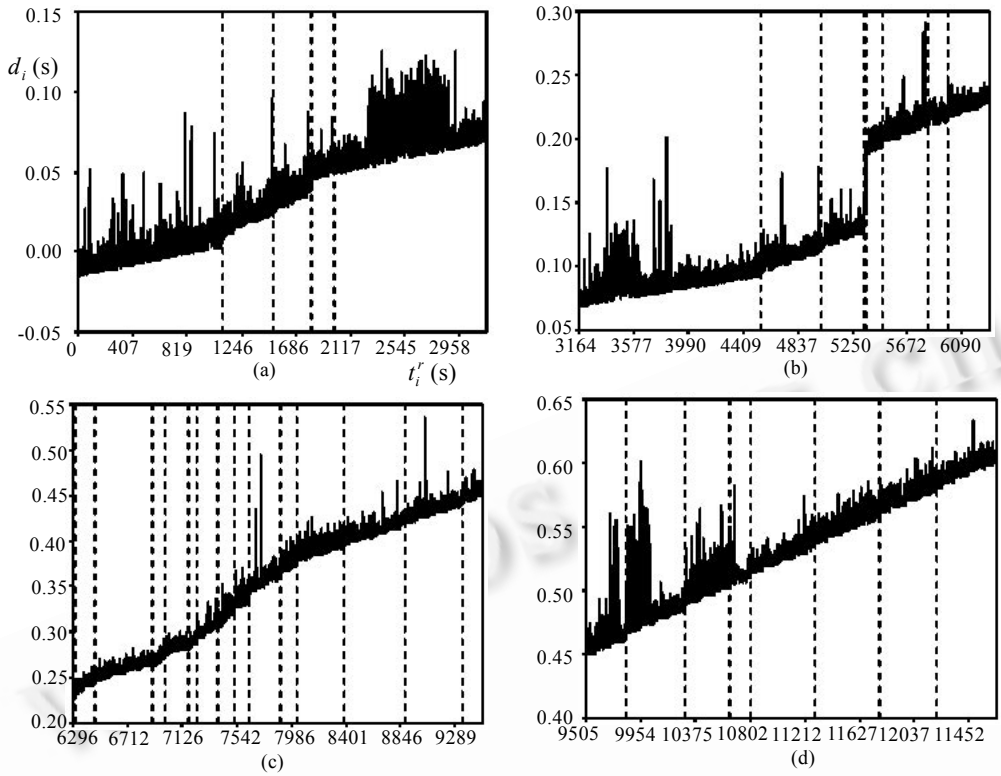

Fig.9 An OWD trace and its 33 segmentations

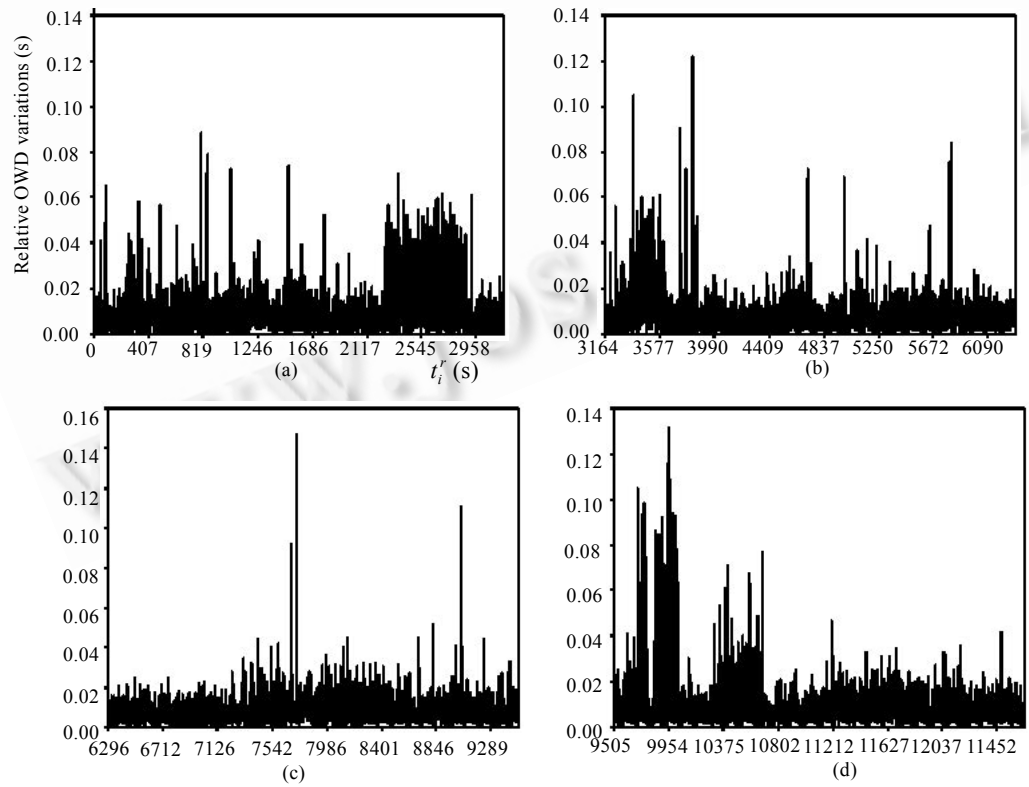

Fig.10 The relative OWD variations after the removal of clock dynamics with SWOTD algorithm

5 Conclusions and Future Work

In this paper, we develop a new cost function and introduce a time series segmentation algorithms OTDA with the complexity of $O(N^2)$ to detect clock dynamics in one-way metrics measurement. With the artificial dataset, comparisons show that the OTDA presents more accurate segmentation than other algorithms. Though we could not get the true relative OWD in practical one-way metrics measurement, the comparison of the observed OWD and the de-skewed relative OWD also indicates that a reasonable clock dynamics detection is performed by OTDA. In addition, SWOTD algorithm is proposed for online clock dynamics detection. We will focus on the following directions to improve the robustness of algorithms for our future work. (1) Find a more accurate robust mechanism to estimate the number of segments underlying a given time series. The method used in the paper is simple and effective, but it only relies on the approximation of experience in experiments. (2) To address the limitation of Top-Down algorithm, OTDA take additional two merging and splitting procedures to relocate the potentially incorrectly splitted boundaries in the previous round of segmentation. We will investigate whether the performance could be improved by further boundaries relocation if suboptimal segmentation is detected. (3) Integrate the improved SWOTD algorithm into *capOWD* to provide an automatic online relative OWD measurement utility.

References:

[1] Almes G, Kalidindi S, Zekauskas M. A one-way delay metric for IPPM. IETF RFC 2679, 1999.

[2] Koodli R, Ravikanth R. One-Way loss pattern sample metrics. IETF RFC 3357, 2002.

[3] Almes G, Kalidindi S, Zekauskas M. A one-way packet loss metric for IPPM. IETF RFC 2680, 1999.

[4] Paxson V, Almes G, Mahdavi J, Mathis M. Framework for IP performance metrics. IETF RFC 2330, 1998.

[5] Paxson V. Measurement and analysis of end-to-end Internet dynamics [Ph.D. Thesis]. Berkeley: University of California, 1997.

[6] Moon SB. Measurement and analysis of end-to-end delay and loss in the Internet [Ph.D. Thesis]. Massachusetts: University of Massachusetts Amherst, 2000.

[7] Ciuffoletti A. Measuring one-way metrics without a GPS. In: Proc. of the PAM 2002. Colorado, 2002. http://www.labs.agilent.com/pam2002/

[8] Tobe Y, Aida H, Tamura Y. Detection of change in one-way delay for analyzing the path status. In: Proc. of the PAM 2000. Hamilton, 2000. http://pam2000.cs.waikato.ac.nz/

[9] Paxson V. On calibrating measurements of packet transit times. In: Proc. of the Int'l Conf. on Measurement and Modeling of Computer Systems 1998 (ACM SIGMETRICS 1998). Madison: ACM Press, 1998. 11~21.

[10] Wang JF, Yang JH, Xie GG, Li ZC, Zhou MT. On-Line estimating skew in one-way delay measurement. In: Proc. of the PDCAT 2003. Chengdu, 2003.

[11] Sugar C, Lenert L, Olshen R. An application of cluster analysis to health services research: Empirically defined health states for depression from the SF-12. Technical Report, Stanford University, 1999.

[12] Tibshirani R, Hastie T, Eisen M, Ross D, Botstein D, Brown P. Clustering methods for the analysis of DNA microarray data. Technical Report, Standord University, 1999.

[13] Himberg J, Korpiaho K, Mannila H, Tikanmaki J. Time series segmentation for context recognition in mobile devices. In: Proc. of the IEEE Int'l Conf. on Data Mining 2001 (ICDM 2001). San Jose: IEEE Computer Science Press, 2001. 203~210.

[14] Keogh E, Chu S, Hart D, Pazzani M. An online algorithm for segmenting time series. In: Proc. of the IEEE Int'l Conf. on Data Mining 2001 (ICDM 2001). San Jose: IEEE Computer Science Press, 2001. 289~296.

[15] Tibshirani R, Walther G, Hastie T. Estimating the number of clusters in a dataset via the Gap statistic. Technical Report 208, Stanford University, 2000.

[16] Vasko KT, Toivonen HTT. Estimating the number of segments in time series data using permutation tests. In: Proc. of the IEEE Int'l Conf. on Data Mining 2002 (ICDM 2002). Maebashi: IEEE Computer Science Press, 2002. 466~473.

[17] Mills DL. Network time protocol (version 3): Specification, implementation and analysis. IETF RFC1305, 1992.