Available online at www.sciencedirect.com
SCIENCE DIRECT
Computer Communications 29 (2006) 1213–1225
computer communications
www.elsevier.com/locate/comcom

On estimating clock skew for one-way measurements

Jingping Bi*, Qi Wu, Zhongcheng Li

Institute of Computing Technology, Chinese Academy of Sciences, P.O. Box 2704, Beijing 100080, People's Republic of China

Received 19 September 2004; revised 18 July 2005; accepted 25 July 2005
Available online 30 August 2005

Abstract

Owning to the asymmetry of Internet paths, more and more studies have turned to the measurement of one-way metrics. Since the clocks at end systems often behave diversely, the synchronization between end hosts is what we care about all along. In this paper, we firstly propose a general model for clock skew estimation in one-way measurements, which turns the problem of clock skew estimation to the solution of *n*-dimension equation group, and give the equation group needed based on different presumptions. We then present Piece-wise Reliable Clock Skew Estimation Algorithm (PRCSEA), which introduces reliability test to estimation results and eliminates the extra presumptions needed by other algorithms, such as only one clock adjustment in the measurements. PRCSEA solves the skew estimation problem in a heuristic way, and it can handle many special cases affecting the estimation of clock skew, such as routing change, clock hiccup and network congestion. PRCSEA is the only algorithm that can handle clock drift to the best of our knowledge.

Keywords: Network measurement; One-way measurement; Clock synchronization; Clock skew; Drift

1. Introduction

Clock is the basis of network performance measurement. Many measurement techniques need to retrieve the time consumed by network events, such as packet-transit time in the network, server-processing time and so on. Due to the asymmetry of Internet routing [9], more and more studies have focused on the measurement of one-way metrics. Since the clocks at both end-systems are involved in delay measurement, synchronization between the two clocks becomes an important issue.

Before proceeding to detailed discussion of clock synchronization, we would like to introduce some definitions first. We define *offset* as the relative difference of the time reported by two clocks, and *skew* as the relative difference of two clocks' first order differentiation (or frequency). Furthermore, we define *drift* as the difference between two clocks' second order differentiation. Our definition consists with previous definitions in [6–8], which model computer clock as a second order differentiable function. Particularly, when we compare a clock with the *true* clock, we often omit 'relative to true clock', that is, the offset (skew, drift) of a clock is the relative (frequency, second order differentiation) difference between the clock and the true clock. Synchronization can happen at a particular moment or during a period, and the elements synchronized can be the value or frequency of clocks. We define *adjustment* as adjusting the offset to zero or under a certain level at a given moment. Furthermore, we define *time keeping* and *frequency keeping* as keeping the offset and frequency to zero or under a certain level during a period, respectively.

Although time keeping is the ideal case for clock synchronization, it is hard to achieve without the help of hardware devices like GPS or CDMA receiver. The asymmetry of network path, workload and even the bandwidth (e.g. ADSL or GPRS subscribers) makes it difficult to estimate the delay difference in two directions, which is essential for calculating clock offset [11,13]. Fortunately, in most cases frequency keeping is enough for us. For example, in delay measurement the dynamic part-mainly queuing delay-attracts much more attention than the static part composed of propagation delay and transmission delay [1,3,10,18]. Besides, many measurement methods,

* Corresponding author. Tel.: +86 10 62565533x9224; fax: +86 10 62567724.
E-mail addresses: jpingbi@ict.ac.cn (J. Bi), abewu@ict.ac.cn (Q. Wu), zcli@ict.ac.cn (Z. Li).

0140-3664/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2005.07.023

such as available bandwidth estimation, are independent from a constant offset [16].

The key of frequency keeping is to estimate skew between two clocks. To find the essential of clock skew estimation, we propose a general model that turns the skew estimation to the solution of an n-dimension equation group, and use this model to evaluate existing techniques. We then present a Piece-wise Reliable Clock Skew Estimation Algorithm (PRCSEA), which introduces a test mechanism to verify the estimation results. If an estimation result in some interval does not pass the test, PRCSEA will divide the interval into smaller ones and re-estimate until the estimation results pass the test or the interval cannot be partitioned any more. To solve the clock drift problem, we propose multi-scale test (MST), which can validate the skew estimation result in log n scales, where n is the number of received packets. Combining with MST, PRCSEA is effective in the case of a long-term measurement, where the clock drift is usually non-neglectable. As far as we know, PRCSEA is the only algorithm that can handle clock drift at present time.

The rest of the paper is organized as follows. We give related works and develop the clock skew estimation model in Sections 2 and 3, respectively. In Section 4, we present the PRCSEA, together with the single-scale test and MST. We evaluate state-of-the-art algorithms using our model in many aspects and find that PRCSEA provides many good features without increasing time complexity in Section 5. The correctness and adaptability of PRCSEA is validated through measurements in Section 6. The conclusions are given in Section 7.

2. Related works

Related works can be classified by how they synchronize clocks. GPS and CDMA receivers are hardwares targeting at time keeping and the offset of clocks can be reduced to tens of microseconds with the help of them. As a result, many large network measurement projects use them to keep time in measurement probes, such as RIPE [14] and Surveyor [4]. Network Time Protocol (NTP) is a software for time-keeping and it is the most widely used time protocol in the Internet [6]. The accuracy of NTP is affected in part by the path characteristics between NTP clients and servers. The clock offset between the synchronized host and the NTP server can often be maintained on the order of multiple milliseconds [5]. However, the clock with NTP may manifest temporal clock skew or drift and these temporary clock behaviors often greatly affect the accuracy of measurement, which makes NTP not a good choice for network measurement.

Using more stable oscillators is a straightforward approach in frequency keeping and there is such a software clock based on CPU registers [12]. However, for the measurement of network internal delay [1], it is nearly impossible to install synchronization devices or upgrade software in all routers along a path. Another approach in frequency keeping is to estimate and remove clock skew from a sequence of timestamps. Paxson selects 'de-noised' One-way Transit Time (OTT) by partitioning the data into intervals and picking the minimum delay value from each interval [11], and uses them to estimate the clock skew. The approach by Moon et al is to fit a line lying under all delay points and as closely to them as possible, and use the slope of the line as the estimated skew [8]. Zhang et al compute the Convex-hull of delay points and give different skew estimations in terms of distinct objective functions [15].

There are several limitations in the above clock skew estimation algorithms. First, none of them provides verification of the estimation results. (Paxson's cumulative minima test verifies whether a continuous trend exists, not whether the estimation is correct or not.) Even though the algorithms rarely fail, the effect of the incorrect estimation may be large in many cases. Accordingly, we always need to observe and judge whether the estimations are right, which further restricts their applications. Second, the functioning of the above algorithms requires strict conditions, so Paxson and Zhang separately present techniques to detect clock adjustment, and then estimate skew to the parts without adjustments. Unfortunately, their adjustment detection algorithms also have many restrictions. Third, all of them do not take clock drift into account. Our experiments indicate that looking on the clock skew as a constant in a long-term measurement often leads to absurd conclusions.

3. Modeling clock skew estimation

To find the key of clock skew estimation and give an insight view of the existing algorithms, we develop a theoretical model with which all skew estimation techniques are comparable. Suppose node r receives a sequence of packets numbered from 1 to n from node s. Let the clocks of s and r are C_s and C_r, respectively. As in previous definitions, C_s and C_r are second order differentiable functions. We use C to denote the true clock, that is, $C(t)=t$. Let l_i^* be the moment that the i-th packet leaves s according to C_* and a_i^* be the moment that the i-th packet reaches r according to C_*, where * is the wildcard. Let d_i and D_i be the *true* delay and the measured delay of the i-th packet. Evidently, we have $d_i = a_i - l_i$, and $D_i = a_i^r - l_i^s$. Fig. 1 shows the relation between the above variables, where the horizontal lines represent the time axes of C, C_r and C_s from the bottom upwards. The two directed diagonals from the C_s axis to the C_r axis represent the sending process of the i-th packet and the $(i+k)$-th packet, respectively. The point of intersection of the diagonal and C_s indicates the sending moment of a packet, and that of the diagonal and C_r indicates the receiving moment of a packet. D_i indicates the difference in time between the two endpoints of the diagonal.

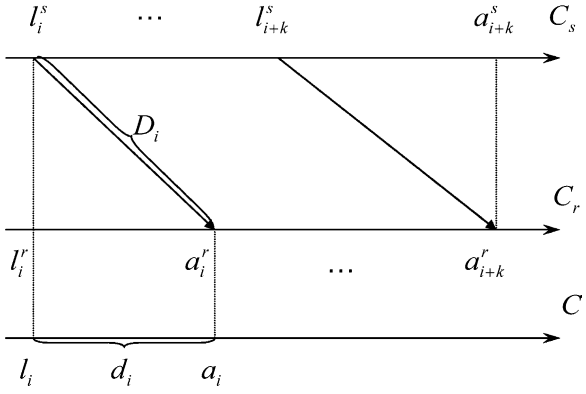

Fig. 1. Relationship between time variables.

In fact, what we care more about is the relative skew between C_s and C_r, rather than the absolute skew. For simplicity and without loss of generality, we assume the clock at source host to be the true clock in the rest of the paper, i.e. $C_s(t) \equiv C(t) = t$, and we have $l_i^s = l_i$, $a_i^s = a_i$.

After collecting n packets at the receiver, we can get the following set of equations:

$$\begin{cases} a_1^r = C_r(a_1) = C_r(l_1 + d_1) \\ \qquad\vdots \\ a_n^r = C_r(l_n + d_n) \end{cases} \tag{1}$$

It is impossible to solve (1) without any knowledge of C_r, and what we have to do first is to model C_r. Firstly, let us consider the simplest case, when the measurement duration is short, we can take the skew of receiver r's clock as a constant. If no clock adjustment occurs in the measurement, then the equation

$$\begin{aligned} C_{\mathrm{r}}(t) &= C_{\mathrm{r}}(l_1) + C'_{\mathrm{r}}(l_1)\cdot(t - l_1) \\ &= C'_{\mathrm{r}}(l_1)\cdot t + C_{\mathrm{r}}(l_1) - C'_{\mathrm{r}}(l_1)\cdot l_1 \end{aligned} \tag{2}$$

stands for $l_1 \le t \le a_n$. Let $\alpha = \dot{C}_{\mathrm{r}}(l_1)$, $\beta = C_{\mathrm{r}}(l_1) - \dot{C}_{\mathrm{r}}(l_1)\cdot l_1$, then Eq. (2) can be written as:

$$C_{\mathrm{r}}(t) = \alpha\cdot t + \beta \qquad l_1 \le t \le a_n \tag{3}$$

where $\alpha - 1$ is the skew we pursue. Substituting Eq. (3) into equation group (1), we get the following equation group:

$$\begin{cases} a_1^r = C_r(a_1) = \alpha\cdot a_1 + \beta = \alpha\cdot(l_1 + d_1) + \beta \\ \qquad\vdots \\ a_n^r = \alpha\cdot(l_n + d_n) + \beta \end{cases} \tag{4}$$

In the following, let us consider the case that some adjustments occur in receiver r's clock. Clock adjustment may happen instantaneously (such as clock is set manually) or for a period of time (such as being adjusted by NTP). We define two classes of adjustment functions separately according to the above cases:

$$\text{instantaneous jump function} \quad \Delta(t) = \begin{cases} 0 & t \le t_1 \\ C_1 & t > t_1 \end{cases} \tag{5}$$

and

$$\text{linear adjustment function} \quad V(t) = \begin{cases} 0 & t \le t_s \\ h\cdot(t - t_s) & t_s \le t < t_e \\ h\cdot(t_e - t_s) & t \ge t_e \end{cases} \tag{6}$$

Then the clock being adjusted j_1 (instantaneously) plus j_2 (linearly) times can be expressed as:

$$\begin{aligned} C_r(t) = {} & \alpha\cdot t + \beta + \Delta_1(t) + \ldots + \Delta_{j_1}(t) + V_1(t) + \ldots \\ & + V_{j_2}(t) \qquad l_1 \le t \le a_n \end{aligned} \tag{7}$$

Let

$$A(t) = \Delta_1(t) + \ldots + \Delta_{j_1}(t) + V_1(t) + \ldots + V_{j_2}(t) \tag{8}$$

Substitute (7) and (8) into (1), we have:

$$\begin{cases} a_1^r = \alpha\cdot(l_1 + d_1) + \beta + A(l_1 + d_1) \\ \qquad\vdots \\ a_n^r = \alpha\cdot(l_n + d_n) + \beta + A(l_n + d_n) \end{cases} \tag{9}$$

If the measurement duration is long, then the presumption that the clock skew in receiver r remains constant will result in unacceptable error (which will be discussed in the next sub-section). Let $U_i(t)$ be the segment from $(t_i, C_r(t_i))$ to $(t_{i+1}, C_r(t_{i+1}))$, i.e.

$$U_i(t) = \begin{cases} \dfrac{C_r(t_{i+1}) - C_r(t_i)}{t_{i+1} - t_i}\cdot(t - t_1) + C_r(t_i) & t_i \le t < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

When $t_i \le t < t_{i+1}$, we can use $U_{\mathrm{i}}(t)$ to approximate $C_r(t)$. Let $U(t) = U_1(t) + \ldots + U_m(t)$, we have

$$C_r(t) = U(t) + A(t) \tag{11}$$

By (1) and (11), we will have a more complicated equation group (12).

$$\begin{cases} a_1^r = U(l_1 + d_1) + A(l_1 + d_1) \\ \qquad\vdots \\ a_n^r = U(l_n + d_n) + A(l_n + d_n) \end{cases} \tag{12}$$

Table 1
Equation groups versus presumptions ($1 \le i \le n$)

Presumptions		Equation groups
Adjustment	Drift	
No	Zero	$a_i^r = \alpha\cdot(l_i + d_i) + \beta$
Yes	Zero	$a_i^r = \alpha\cdot(l_i + d_i) + \beta + A(l_i + d_i)$
Yes	Non-zero	$a_i^r = U(l_i + d_i) + A(l_i + d_i)$

Table 1 summarizes the relationship between different presumptions and the needed equation groups.

According to the model, Paxson and Zhang solve equation group (9) by detecting clock adjustments and estimating clock skew in adjustment-free intervals. Their detection and estimation approaches stop at equation group (12) because of the following three reasons. First, piece-wise linear function $U(t)$ is only an approximation of the $C_r(t)$. In the real world the skew changes continuously and it is impossible for us to determine the change points. Second, without knowledge of a clock's characteristics, it will be very difficult to give a time scale under which the clock skew can be taken as constant. Third, it is always hard to tolerate network noise by simply partitioning the measurement results into segments. If the clock skew estimation results are different in the neighboring intervals, it is difficult to distinguish whether the clock skew really changes or just the network delay varies. These difficulties require us to solve equation group (12) from a different point of view.

4. Piece-wise reliable clock skew estimation algorithm (PRCSEA)

4.1. The algorithm

Existing techniques do not test the reliability of estimation results, disqualifying them from further investigation. At the same time, it is hard to decide whether the network noise or the clock itself results in the current estimation. We present PRCSEA, which can test the clock skew estimation results in each time interval. If the estimation results in an interval does not pass the test, the interval will be partitioned into smaller ones and re-estimated until the test passes. Due to the introduction of verification to estimation results, PRCSEA bypasses two obstacles (identifying skew changing point and determining appropriate time scale) in the way of solving equation group (12). In addition, the algorithm is as follows.

Algorithm 1. Piece-wise reliable skew estimation algorithm

```
PROC prcsea(data_array ma, int start, int end)
calculate_fit_line(ma, start, end, α, β);
   if (reliablility_test(ma, start, end, α, β))
      print(start, end, PASS, α);
   else
      if (end-start > P_min and l_end-l_start > T_min)
         prcsea(ma, start, (end-start)/2);
         prcsea(ma, (end-start)/2 + 1, end);
      else
         print(start, end, FAIL, α);
      endif
   endif
endproc
```

This is a recursive algorithm, and it resembles the classic *binary tree inorder traversal* algorithm. PRCSEA does not provide details about how to resolve α and β in equation group (4) (*calculate_fit_line*) and test the estimation results (*reliablility_test*), which makes it a framework algorithm in skew estimation and independent from these two sub-algorithms. We can use the existing algorithms to resolve α and β. The test algorithm will be demonstrated in next two sub-sections.

The two parameters in PRCSEA, $P_{\min}$ and $T_{\min}$, represent the minimal number of packets an interval must contain and the minimal interval length, respectively. If an interval is too short, or contains too few points, the estimation result might be misleaded by network noises and become useless. In this paper, we set $P_{\min}$ to 100 packets and $T_{\min}$ to 20 s. The existence of these two parameters does not mean that PRCSEA cannot deal with adjacent adjustments (or other events) in the measurement. In Section 6, we will demonstrate the output of PRCSEA is still correct when the clock being adjusted twice in very short time.

4.2. Single-scale test (SST)

Previous works show that, the minimum value of delay indicates the path was lightly loaded when the packet experiencing that delay was traversing in the network [2]. If the results given in the clock skew estimation algorithm are right, then the points with the minimal delay will be on the fitting line provided by the algorithm. Let q be the probability that a packet experiences the minimum delay (and the corresponding point is on the line), then the number of points on the line has a binomial distribution. Let

$$R(k) = \sum_{i=0}^{k} C_n^i \cdot q^i \cdot (1-q)^{n-i} \tag{13}$$

and $R(k)$ means the probability that the number of points on the line is smaller than or equal to k. Let us take the *null hypothesis* that the estimation result is correct, and let m be the number of points on the line. If $R(m)$ is small, then we reject the null hypothesis, and accept the *alternative hypothesis* that skew estimation algorithm gives the incorrect answer.

There are still things to be done before the execution of the single-scale test. The first is how to get the value of q. The probability that packets experience the minimal delay may be diverse in different paths or even at distinct moments in the same path, which makes it difficult to determine q. Given n and k, $R(M)$ decreases with the increase of q, meaning that we are more likely to reject the null hypothesis. If we make the first-class mistake, that is, we reject an estimation result that is correct, we will partition the interval into smaller ones and re-estimate the skew. Making second-class mistakes results in accepting incorrect estimations. Evidently, the effect of the second-class mistake is much worse than that of the first one, so we

apt to a more prudent way and the value of q should be chosen to be larger. On the other hand, if the value of q is too large, the test may be so strict that it always refuses the estimation of some measurement results. In this paper, we choose $q=0.05$ and we will demonstrate in the next section that our algorithm is robust, which gives the correct answer when the probability is much larger than the preset value.

Another is the threshold of *p-value* beneath which we reject the null hypothesis. Similar to the previous discussion of q, according to our policy being prudent, we let the threshold $p_o=0.05$.

4.3. Clock drift and multi-scale test (MST)

Clock drift issues its challenges to the single-scale test. If the estimated skew value happens to meet the true skew value at some intervals, and the path is light-loaded, then most of the points in these intervals might be on the line, which impels the SST to accept the null hypothesis. We present MST to make up for the limitations. Specifically, let q be the probability of a packet experiencing a minimal delay, then the probability of one or more packets experiencing a minimal delay in w packets is:

$$q_w = 1-(1-q)^w$$

Divide all the data into $\lceil n/w \rceil$ segments with each of the first $\lfloor n/w \rfloor$ segments containing exactly w packets. In the following, we only talk about the $\lfloor n/w \rfloor$ w-width segments. If one or more packets in a segment experience the minimal delay, then we term the interval as a *good* segment, otherwise we call it a *bad* segment. Let

$$R_w(k) = \sum_{i=0}^{k} C^i_{\lfloor n/w \rfloor} \cdot q_w^i \cdot (1-q_w)^{\lfloor n/w \rfloor - i} \tag{14}$$

and $R_w(k)$ is the probability that the number of good segments is smaller than or equal to k. Let m_w be the number of good segments. Similarly, based on the value of $R_w(m_w)$, we can accept or reject the null hypothesis that the estimation is correct at scale w. MST checks for all w in $\{2^0, 2^1, \ldots, 2^{\lfloor \log_2 n \rfloor - 2}\}$. Only when the null hypothesis is accepted at all scales, we say that the estimation passes the MST. We can see that the former test is a degenerate special case of $w=1$.

5. Evaluation

In this section we will evaluate PRCSEA and state-of-the-art algorithms using the model we develop in Section 3. We calculate the error of these algorithms first and then give a comparison of their time complexity. Finally, we compare the flexibility, robustness, and reliability of these algorithms. Moon et al answer the question whether the error has relationship with the skew itself [8], and here we give the error upper bound.

5.1. Error

All above algorithms estimate clock skew in adjustment-free intervals, so errors of these algorithms are determined by the methods they used to solve equation group (4) where $a_1,\ldots,a_n$ and $l_1,\ldots,l_n$ are known and $\alpha,\beta,d_1,\ldots,d_n$ are unknown. Unfortunately, there are n equations in the group while the number of variables is $n+2$, and we will get infinite solutions in theory. Notice that $d_1,\ldots,d_n$ are one-way transit time at different moments in the same path, so the relationship might make them not fully independent variables. All of current algorithms could be transformed to use the relationship in OTTs to solve equation group (4). PRCSEA can use any existing ways to solve the equation group and the error depends on the sub-algorithm it chose, so we exclude it here.

5.1.1. Paxson's algorithm

Paxson uses piece-wise minimal OTT to denoise the measured values, and selects median of the slopes in each minimal points pair as the estimated skew. Suppose there are m intervals and K_i is the subscript of the measured OTT in the i-th interval. Let $s_{i,j}$ be the slope of (l_{K_i}, D_{K_i}) and (l_{K_j}, D_{K_j}), which is

$$s_{i,j} = (D_{K_i} - D_{K_j})/(l_{K_i} - l_{K_j}) \tag{15}$$

Let

$$e_{i,j} = s_{i,j} - (\alpha - 1) \tag{16}$$

which is the error of using $s_{i,j}$ to estimate the skew. The median of $\{e_{ij} | 1 \le i < j \le m\}$ is the error of Paxson's algorithm. After the de-noising process, equation group (4) becomes

$$\begin{cases} a^r_{K_1} = \alpha \cdot a_{K_1} + \beta = \alpha \cdot (l_{K_1} + d_{K_1}) + \beta \\ \vdots \\ a^r_{K_m} = \alpha \cdot a_{K_m} + \beta = \alpha \cdot (l_{K_m} + d_{K_m}) + \beta \end{cases} \tag{17}$$

Subscribe equation i and j in the above group, we will get

$$\alpha = (a^r_{K_i} - a^r_{K_j})/(l_{K_i} - l_{K_j} + d_{K_i} - d_{K_j}) \tag{18}$$

Substitute $s_{i,j}$ and α into (16), we can get

$$\begin{aligned} e_{i,j} &= (a^r_{K_i} - a^r_{K_j})/(l_{K_i} - l_{K_j}) - (a^r_{K_i} - a^r_{K_j})/(l_{K_i} - l_{K_j} \\ &\quad + d_{K_i} - d_{K_j}) \\ &= \alpha \cdot (d_{K_i} - d_{K_j})/(l_{K_i} - l_{K_j}) \end{aligned} \tag{19}$$

In most cases, the absolute value of computer clocks' skew is between $10^{-3} \sim 10^{-6}$ [7], which means that $|\alpha - 1| \ll 1$ and $\alpha \approx 1$. Approximately,

$$e_{i,j} \approx (d_{K_i} - d_{K_j})/(l_{K_i} - l_{K_j}) \tag{20}$$

Let K_i be the subscript of the minimal OTT in the i-th interval, and we have $d_{k_i} \le d_{K_i}$ and $D_{k_i} \ge D_{K_i}$ according to the definition of K_i and K_i. Substitute $D_{k_i} = a^r_{k_i} - l_{k_i} =$

$(\alpha-1)\cdot l_{k_i}+\alpha\cdot \le d_{k_i}+\beta$ into $D_{k_i}\ge D_{K_i}$, we can get

$$d_{K_i}-d_{k_i}\le(\alpha-1)\cdot(l_{k_i}-l_{K_i}) \tag{21}$$

$$\begin{aligned}|d_{K_i}-d_{K_j}| &= |d_{K_i}-d_{k_i}+d_{k_i}-d_{k_j}+d_{k_j}-d_{K_j}|\\ &\le |d_{k_i}-d_{k_j}|+|d_{K_i}-d_{k_i}+d_{k_j}-d_{K_j}|\\ &\le |d_{k_i}-d_{k_j}|+|\alpha-1|\cdot\max\{|l_{k_i}-l_{K_i}|,\\ &\quad |l_{k_j}-l_{K_j}|\}\end{aligned} \tag{22}$$

Substitute (22) into (20), we have

$$\begin{aligned}|e_{i,j}| &\approx |d_{K_i}-d_{K_j}|/|l_{K_i}-l_{K_i}|\\ &\le [|d_{k_i}-d_{k_j}|+|\alpha-1|\cdot\max\{|l_{k_i}-l_{K_i}|,|l_{k_j}-l_{K_j}|\}]\\ &\quad /|l_{K_i}-l_{K_i}|\end{aligned} \tag{23}$$

$|l_{K_i}-l_{K_j}|$ is the absolute difference between the moment of packet K_i and K_j leaves s. Approximately, we have

$$|l_{K_i}-l_{K_j}|\ge(|j-i|-1)\cdot w \tag{24}$$

$$\max\{|l_{k_i}-l_{K_i}|,|l_{k_j}-l_{K_j}|\}\le w \tag{25}$$

where w is the average width of each interval. Substitute (24) and (25) into (22), we have

$$|e_{i,j}|\le\frac{|d_{k_i}-d_{k_j}|}{(|j-i|-1)\cdot w}+\frac{|\alpha-1|}{|j-i|-1} \tag{26}$$

Given the distribution of end-to-end delay, we can easily calculate the upper bound error of Paxson's algorithm using formula (26).

5.1.2. Line-fitting algorithm (LFA)

Moon et al try to fit a line lying under all delay points and as closely to them as possible, and use the slope of the line as the estimated skew [8]. The line satisfies that all the delay points are above or on the line, and the sum of vertical distance between each point and the line reaches the minimum. Using linear algorithm by N. Megiddo, the time complexity of the algorithm is O(n). Zhang et al calculate the convex hull of delay points and give different estimated skew value according to different optimal requirements. The requirements include: (1) minimizing the sum of vertical distance between the point and the line, (2) minimizing the area between the segments made up by the points and the line, and (3) maximizing the number of points on the line. The skew value got by the first optimal requirement is equal to the value got by Moon et al.

We call an algorithm Line Fitting Algorithm if it uses a line to fit a line to the bottom of the delay points $\{(l_i,D_i)|1\le i\le n\}$ and reaches some optimal requirements on the condition that no point is underneath the line. The algorithms of Moon and Zhang et al belong to LFA. Zhang et al show that the line must coincide with one segment of the convex hull, which impels us to study the property of the convex hull. Theorem 1 gives the error upper bound of the segments that makes up the convex hull.

Theorem 1. If no point is beneath the line determined by $p_i=(l_i,D_i)$ and $p_j=(l_j,D_j)$ $(1\le i<j\le n)$ and the skew can be taken as constant, we have

(1) $\min\{d_i,d_j\}\le\min\{d_k|i<k<j\}$
(2) At least one of the two formulas $\max\{d_i,d_j\}\le\min\{d_k|1\le k<i\}$ and $\max\{d_i,d_j\}\le\min\{d_k|j<k\le n\}$ must be true.

Proof. : The line determined by p_i and p_j is

$$y(t)=D_i+(t-l_i)\cdot s_{i,j} \tag{27}$$

where $s_{i,j}=(D_j-D_i)/(l_j-l_i)=(\alpha-1)+\alpha(d_j-d_i)/(l_j-l_i)$ is the slope of the line.

$y(l_k)$-D_k determines the relationship between p_k and the line. If it is positive, p_k is below the line. If it is negative, p_k is above the line. $y(l_k)-D_k$ equals to zero means the line is just across p_k. Substitute the expression of $y(l_k)$ and D_k, we can get

$$\begin{aligned}y(l_k)-D_k &= D_i+(l_k-l_i)\cdot s_{i,j}-D_k\\ &= \alpha\cdot[l_j\cdot(d_i-d_k)+l_k\cdot(d_j-d_i)\\ &\quad +l_i\cdot(d_k-d_j)]/(l_j-l_i)\end{aligned} \tag{28}$$

Let $\Phi=l_j(d_i-d_k)+l_k(d_j-d_i)+l_i(d_k-d_j)$. Because $l_i<l_i$ and $\alpha>0$, the symbol of $y(l_k)-D_k$ is the same as Φ. If $d_i\le d_j$ and $k>j$, obviously we have $d_j-d_i\ge0$ and $l_k>l_j$. Furthermore,

$$\begin{aligned}\Phi &\ge l_j\cdot(d_i-d_k)+l_j\cdot(d_j-d_i)+l_i\cdot(d_k-d_j)\\ &= (l_j-l_i)\cdot(d_j-d_k)\end{aligned} \tag{29}$$

If $d_j-d_k>0$, $\Phi>0$ must be true since $l_j-l_i>0$, which means that p_k is under the line and obeys the presumption of the theorem. Therefore, d_j-d_k must be smaller than or equal to zero, that is:

$$\max\{d_i,d_j\}=d_j\le\min\{d_k|j<k\le n\} \tag{30}$$

When $d_i\le d_j$ and $i<K<j$, we can get

$$\begin{aligned}\Phi &\ge l_j\cdot(d_i-d_k)+l_i\cdot(d_j-d_i)+l_i\cdot(d_k-d_j)\\ &= (l_j-l_i)\cdot(d_i-d_k)\end{aligned} \tag{31}$$

Similarly, $d_i-d_k\le0$ must be true, and

$$\min\{d_i,d_j\}=d_i\le\min\{d_k|i<k<j\} \tag{32}$$

When $d_i\ge d_j$, we can get the similar formula

$$\max\{d_i,d_j\}=d_i\le\min\{d_k|1\le k<i\} \tag{33}$$

$$\min\{d_i,d_j\}=d_j\le\min\{d_k|i<k<j\} \tag{34}$$

From Theorem 1 we can get that the error of line-fitting algorithm depends on the minimum of end-to-end one-way delay, which is determined by the end-to-end delay distribution.

□

5.2. Time complexity

We always want an algorithm's time complexity as low as possible under the precondition that it has as many above properties as possible. A large-scale measurement always contains up to millions of samplings, so the time complexity of an algorithm determines how fast and efficient it can eliminate the clock effects from data.

PRCSEA resembles the classic inorder traversing binary tree algorithm. And the steps except for *calculate_fit_line*, *reliablility_test* and the recursive call to *prcsea* obviously have $O(1)$ complexity, so what we need is to calculate time complexity of the MST. After that, the complexity of PRCSEA is:

$$O(\log n)\cdot\max\{O(\text{calculate\_fit\_line}), O(\text{reliability\_test})\} \tag{35}$$

First, we calculate the time needed for testing at a given scale w in MST. If we keep the states of each segment (*good* or *bad*) when the scale is $w/2$, we can get the state of a w-width segment by using the or operator between the states of two $w/2$-width segments that make up the w-width segment. (Suppose $good=1$ and $bad=0$.) So the time used in calculating m_w is $o(\lfloor n/w\rfloor)$. Let

$$P_w(i) = C^i_{\lfloor n/w\rfloor}\cdot q_w^i\cdot(1-q_w)^{\lfloor n/w\rfloor - i} \tag{36}$$

and $P_w(i+1)$ can be deduced by $P_w(i)$ using:

$$P_w(i+1) = P_w(i)\cdot\frac{\lfloor n/w\rfloor - i + 1}{i}\cdot\frac{q_w}{1-q_w} \tag{37}$$

which means we can get $P_w(i+1)$ within $O(1)$ time if we know $P_w(i)$. As for $P_w(0)=(1-q_w)^{\lfloor n/w\rfloor}$, the complexity is also $O(1)$ if we use the combination of logarithmic and exponential functions. By (14) and (36), we have

$$R_w(m_w) = \sum_{i=0}^{m_w} P_w(i). \tag{38}$$

So the time used in calculating $R_w(m_w)$ is $O(m_w)=O(\lfloor n/w\rfloor)$. Until now we can get the test time at a given scale w to be $O(\lfloor n/w\rfloor)+O(\lfloor n/w\rfloor)=O(\lfloor n/w\rfloor)$, and the time complexity of MST is

$$O(n) + O(\lfloor n/2^1\rfloor) + \ldots + O(\lfloor n/2^{\lfloor \log_2 n\rfloor - 2}\rfloor) = O(n)$$

The time complexity of *calculate_fit_line* in both Moon's and Zhang's algorithms is $O(n)$, and Paxson's is $O(n\log n)$. We use Zhang's algorithm in *calculate_fit_line*. By (15), the time complexity of PRCSEA is

$$O(\log n)\cdot\max\{O(n), O(n)\} = O(n\log n)$$

Table 2 summarizes the time complexity of different clock skew estimation algorithms. In Zhang's algorithm, R is the number of clock adjustments, and w is the number of packets in an interval. In the case of many hosts or routers synchronizing periodically, the time complexity of Zhang's algorithm can also be written as (or nearly) $O(n\cdot(n/p+w))$, where p is the period of the synchronization. We can see, even with enhanced functions, the overall time cost of PRCSEA is at the same level or better than those of current algorithms when there is periodic synchronization. And the time cost remains the same when clock drift is counted in.

5.3. Flexibility

As discussed in Section 5.1, all current algorithms give correct estimation only when there is no clock adjustment and the skew is nearly constant, so the *flexibility* is defined as the ability to find an interval as long as possible under these restrictions. However, the stability of clock is affected by many factors such as temperature and pressure and has a wide range, which make it difficult to build a flexible algorithm. Paxson divides the n samplings into about $\sqrt{n}$ intervals where each interval is at least $\sqrt{n}\cdot(l_n-l_1)$ or contains at least $\sqrt{n}$ samplings. Choosing $\sqrt{n}$ is a balance between de-nosing and calculating the median, so Paxson does not take the clock behavior into account when he makes the segmentation. Moon et al do not consider the problem of segmentation at all. The convex-hull approach by Zhang et al does not need to segment the n samplings when calculating the skew, and their segmentation aims at detecting clock adjustments. The interval must be narrow enough so that there is at most one adjustment in any three consecutive intervals with width w. On the other hand, the interval must be wide enough so that the straight line underneath the delay points can be observed within each interval. The two opposite restrictions make it difficult to

Table 2
Complexity of different algorithms

Presumptions		Paxson's	Moon's	Zhang's	PRCSEA
Adjustment	Drift				
No	ZERO	O(n log n)	O(n)	O(n)	O(n)
Yes	ZERO	O(n log n)[a]	N/A	O(n(R+w))[b]	O(n log n)
Yes	NON-ZERO	N/A	N/A	N/A	O(n log n)

[a] At most 1 adjustment can be detected.
[b] At most 1 adjustment within any three consecutive intervals.

choose w. PRCSEA uses post-dividing technology to segment the whole sampling. That is, it divides an interval only when the estimated skew in that interval does not seem to be a valid result. It is the only one that can give variable width intervals. In the next section, we can see that the width of the intervals divided by PRCSEA relates to the speed of skew changes, which validates its flexibility.

5.4. Robustness

Since the main disturber in skew estimation is clock adjustment, we define *robustness* as the ability to tolerate clock adjustments. The simplest solution is proposed by Moon et al, in which they did not consider any adjustment at all. Paxson proposes a simple adjustment detection algorithm that can only bear no more than one instantaneous adjustment. Zhang et al propose algorithms for both instantaneous and linear adjustments, but their approach is based on the assumption that the interval between two adjustments is large enough, which is not always correct. For some abnormal clock behaviors such as hiccup [Pa97] where the clock is adjusted twice in a short period of time, their approach will fail. PRCSEA uses MST to verify whether the estimation is misled by some events such as adjustment, route changes, etc. PRCSEA can give correct skew estimation without eliminating adjustment, and it can bear both kinds of adjustments.

5.5. Reliability

It is almost impossible that an inference-based algorithm can work well in all situations, so it is important for the algorithm to provide verification for the estimation results. Even though the algorithms rarely fail, the effect of the incorrect estimation may be large in many cases. Accordingly, we always need to observe and judge whether the estimations are right, which further restricts their applications. We define *reliability* as the ability to know whether the estimation is correct or not. Paxson's *cumulative minima test* verifies whether a continuous trend exists in the OTT measurements (that is, the skew exists or not), but it does not tell us whether the estimation is believable or not. Both the single-scale test (SST) and MST proposed to verify the correctness of estimated skew value. SST and MST are based on the assumption that the minimal delay will occur several times in a measurement, which is also the assumption of clock skew estimation algorithm.

6. Experiments

6.1. Clock adjustments

In order to validate the algorithm, we perform one-way delay measurements in both wide-area and local-area networks. Traditional one-way measurement tools are composed of the sender part and the receiver part. Both parts must be deployed correctly to perform a measurement, which is rarely possible when the receiver is a third-party router. We developed a tool tsping, which uses ICMP timestamp request message to measure one-way delay. The resolution of ICMP timestamp message is at the millisecond level, and it needs more time for a router to generate an ICMP reply message than just forward a packet [17]. Fortunately, what we care about in this paper is the estimation of clock skew, not the precise one-way delay value. Therefore, the use of ICMP will not affect the validation of the algorithm.

We install the tool tsping in 3 nodes: Institute of Computing Technology in Chinese Academy of Sciences, University of Science and Technology of China and Xi'an Jiaotong University. The destination nodes which cover 8 ASes are randomly chosen. Fig. 2 gives the recursive process of PRCSEA where the clock is adjusted once in a measurement. The parameters used here are: $P_{min}=100$ packets, $T_{min}=20$ seconds, $q=0.05$ and $p_0=0.05$, as discussed in Section 4. We use the simple example here to illustrate the recursive process of PRCSEA, and a study of the impact of these parameters on PRCSEA will be issued latter. The x-axis is the timestamp of the sender, and the y-axis is the one-way delay. The black solid line (the delay line) is the revised delay calculated by the original measured value subtracting the minimal measured value. The vertical light-color line gives the intervals in which PRCSEA estimates and tests the result. The thick line at the bottom of the delay points is the output of PRCSEA, where different colors represent different meanings. Light color means the estimation passing MST, while dark color means failure. PRCSEA first uses *calculate_fit_line* to estimate the skew of the whole data, and the value of estimated skew is the slope of the dark line in Fig. 2a. After this estimation, PRCSEA divides the data from the middle and continues in each interval, as Fig. 2b demonstrates. Only the right-hand interval gives the correct estimation in the second round of computation, so PRCSEA has to partition the left hand interval and goes on. Finally, in Fig. 2d, where PRCSEA gives correct estimation in all intervals, the algorithm stops.

To show the robustness of PRCSEA, we test it in a more complicated scenario where there are multiple clock adjustments during the measurement. The destination device in the scenario is a router, where a daemon wakes up periodically to synchronize its clock. The measurement duration is around one hour, and the daemon synchronizes the router's clock around every 979 s, according to the sender's clock. Fig. 3 gives the output of PRCSEA under the scenario, where the parameters are the same as in Fig. 2 We can see that during the intervals where there are clock adjustments, the estimation of PRCSEA does not pass the MST, and it is marked dark (not trustable). Furthermore, during the non-adjustment intervals, the estimation of PRCSEA is marked light (trustable).

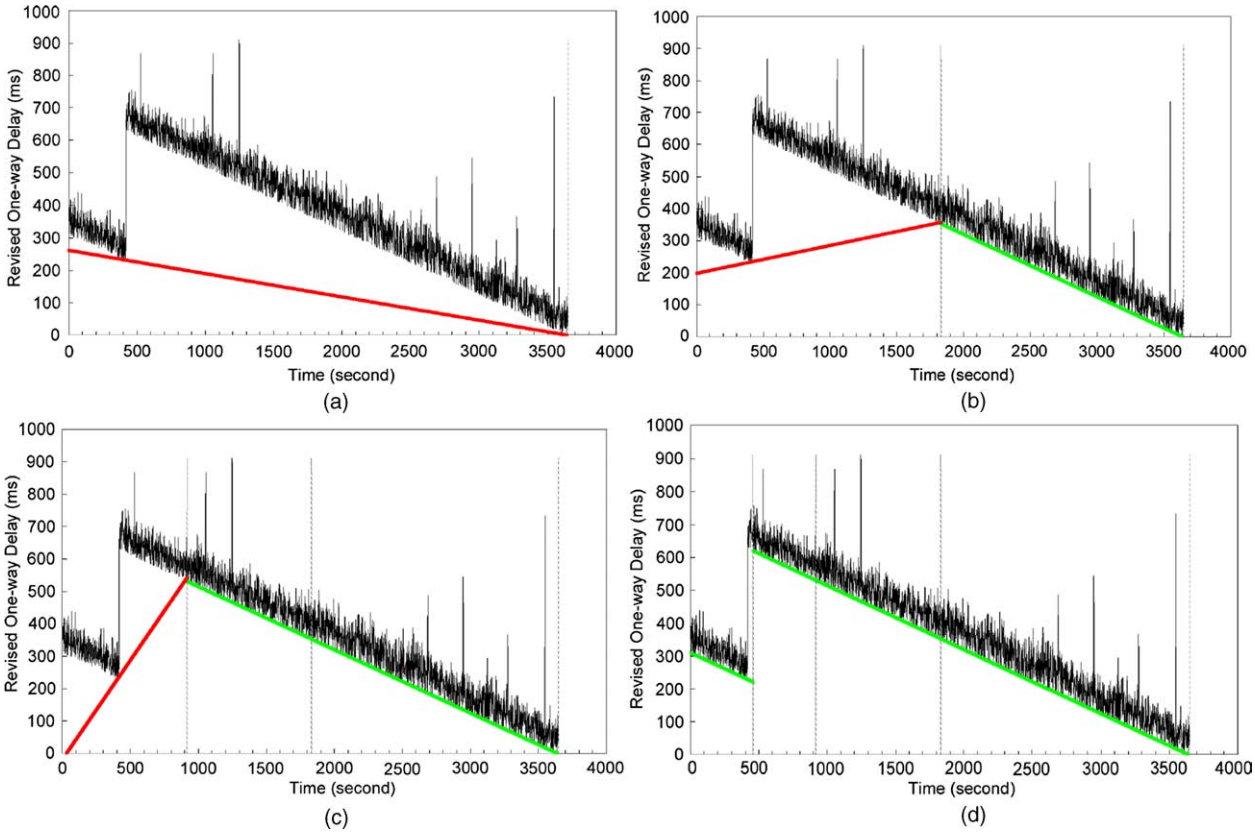

Fig. 2. The recursive process of PRCSEA.

6.2. The impact of parameters

Now we'd like to study the impact of the parameters P_{min}, T_{min}, q and p_0, on the algorithm. First, P_{min} and T_{min} control the minimal interval length, which is measured in the number of packets and seconds, respectively. If the length of an interval is too short, or the interval contains too few packets, the estimation result might be misled by network noises and becomes useless. We use the data in the Fig. 3 to illustrate its impact on the algorithm. The results are illustrated in Fig. 4, where P_{min} and T_{min} are set to (400,80), (200,40), (50,10) and (25,5) in the four charts, respectively, and other variables are set to be constant. The reason we change P_{min} and T_{min} at the same scale is that they both control the interval length, although in different units. From these charts in Fig. 4 and Fig. 3 (where P_{min} and T_{min} are 100 and 20, respectively) we could see that, if P_{min} and T_{min} are too large to be comparable with the clock adjustment interval, the algorithm might stop without any believable estimation, as in Fig. 4a. However, if they are too small and there is no enough data, the estimation might always pass MST, as in Fig. 4d.

The p_0 is the threshold beneath which we reject the null hypothesis, and its selection is rather objective. There are two frequently used values, 0.05 and 0.01, in most of the null hypothesis cases. We have tried the two values, and they do not have much impact on the algorithm.

In previous Section 4.2 we have discussed the impact of q on the algorithm, and select q to be 0.05 based on the discussion. Now we would like to know the behavior of PRCSEA when q is set to be other values. We set q to be 0.01, 0.02, 0.1 and 0.2 and keep other parameters the same as in Section 6.1. The results are shown in Fig. 5. We can see that if q is too small, the very few points at the intersection of light line and the delay points make the estimation pass MST. For example, in the scenario where $q=0.01$ and there

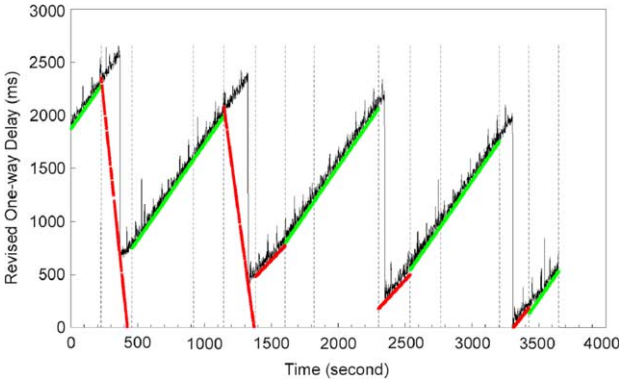

Fig. 3. The output of PRCSEA under multiple clock adjustments.

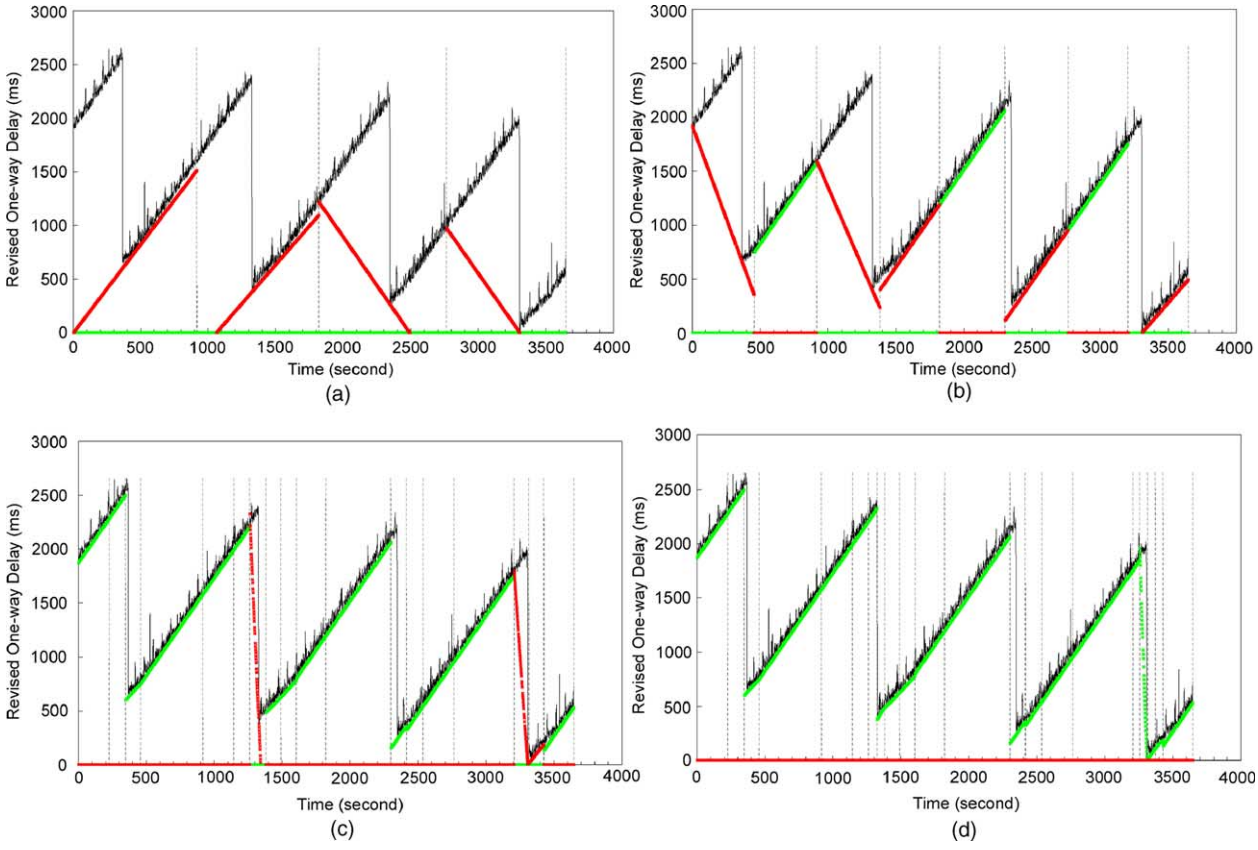

Fig. 4. The impact of P_{min} and T_{min} on the algorithm.

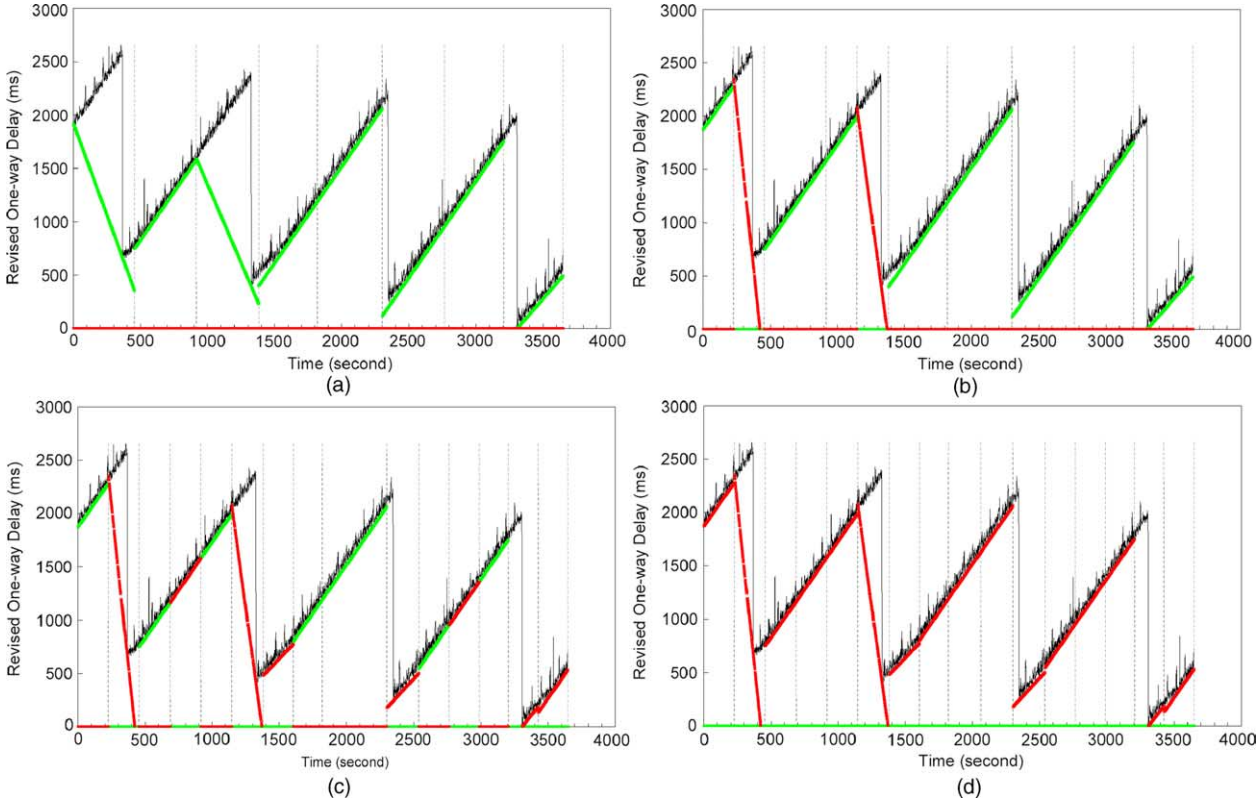

Fig. 5. The impact of q on the algorithm.

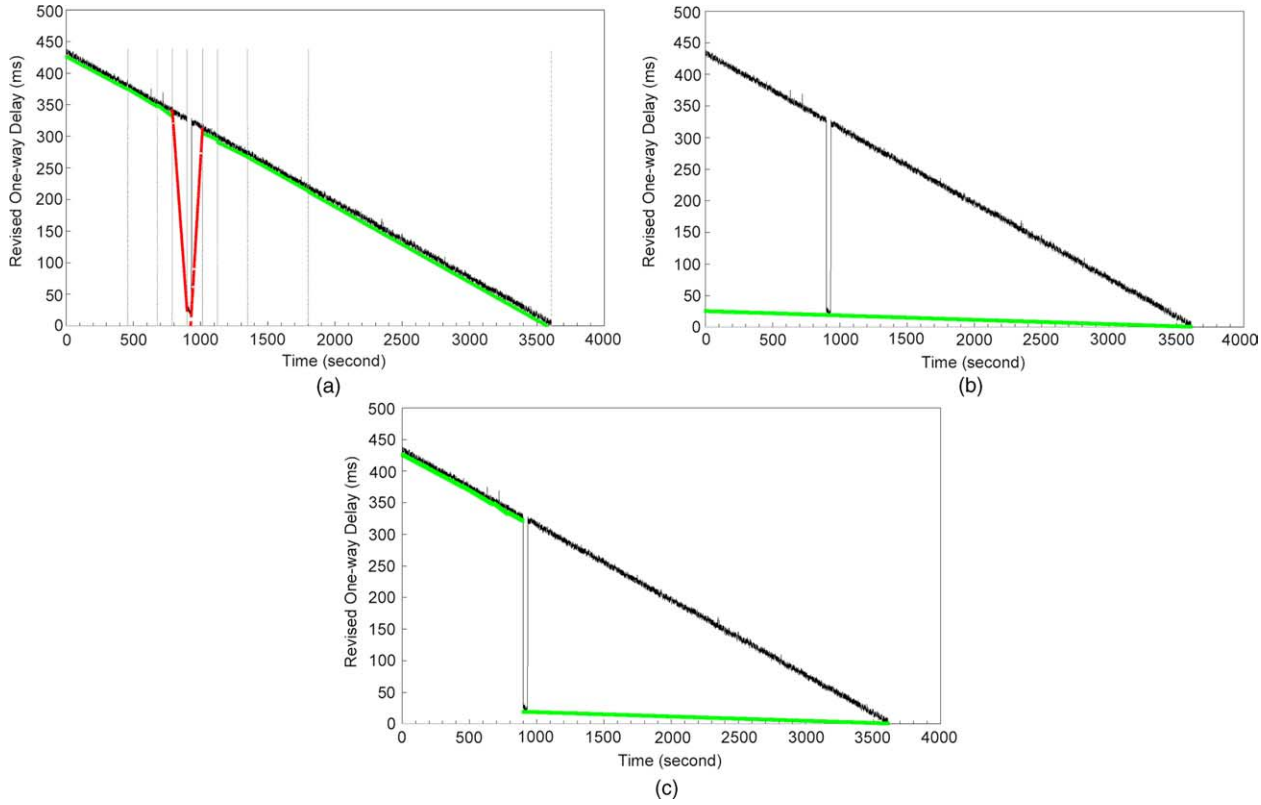

Fig. 6. The output of PRCSEA, Paxson's and Zhang's algorithm with two close clock adjustments.

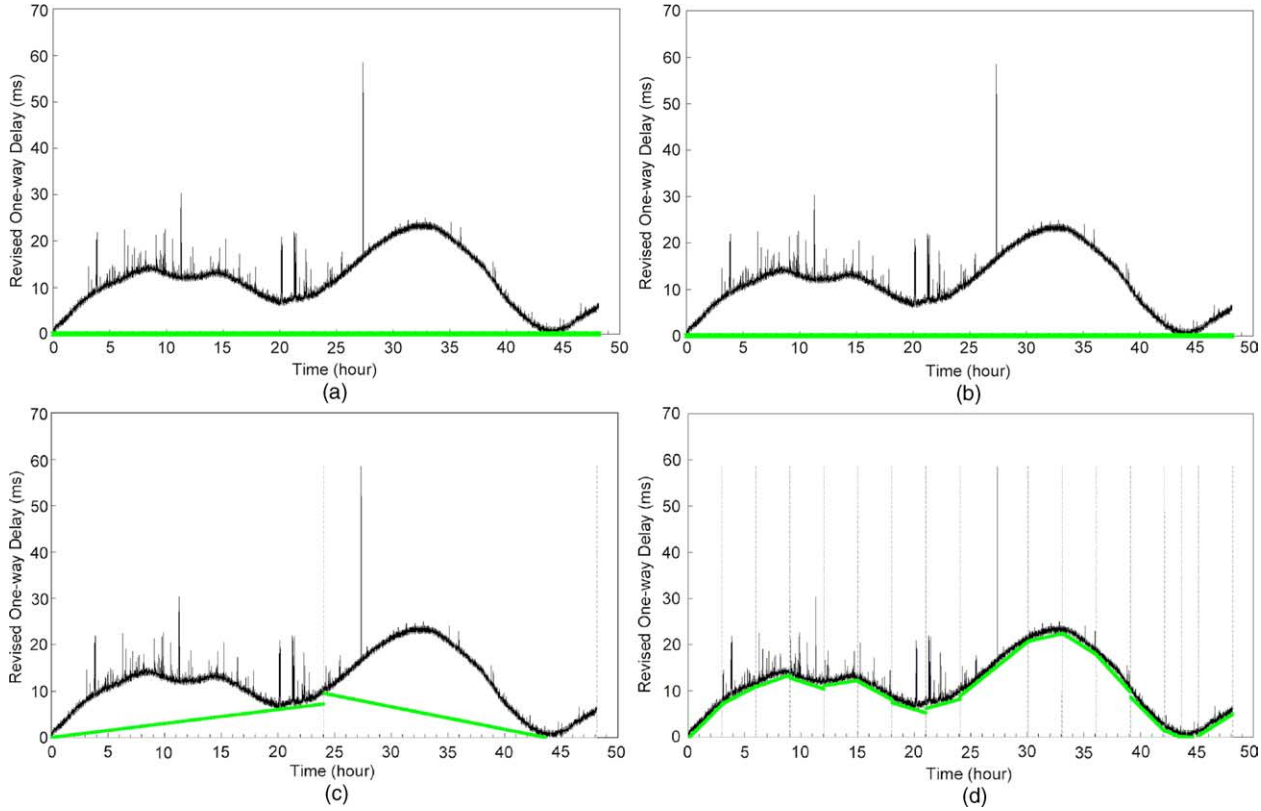

Fig. 7. Comparison of Paxson's algorithm, Zhang's algorithm, PRCSEA-SST, and PRCSEA-MST with non-zero drift.

are 200 points in an interval, since there must be at least two points on the line, the estimation results always pass MST. This is the case shown in Fig. 5a. On the other side, if q is much higher than the inherent delay probability in the measurement result, the estimation will have no way to pass MST, as shown in Fig. 5d. Therefore, we set q to be a smaller value, 0.05. Fortunately, MST helps PRCSEA to handle the scenario where the probability is much higher than 0.05, as shown in the next section.

6.3. Clock hiccup and drift

The three charts in Fig. 6 give the output of PRCSEA, Paxson's and Zhang's algorithms, respectively when there are two close clock adjustments in the measurements, which is called clock hiccup by Paxson [10]. We can see, in the intervals that do not contain the hiccup, PRCSEA gives the correct estimation, and on the contrary, the estimated skew is marked unbelievable (dark color) for failing to pass the MST. In this case, Paxson's algorithm fails to detect the hiccup (Fig. 6b), and Zhang's looks on it as one adjustment (Fig. 6c).

The four charts in Fig. 7 are to verify the robustness of PRCSEA when the drift is non-negligible. The input of PRCSEA in Fig. 7 is a 48 h-long trace in a local area network, so the value of q (more than 0.9) is much higher than the default value (0.05). To show the drift clearly, we remove a constant skew from the data. Other algorithms do not provide the non-constant skew estimation, and they only give non-skew estimation, as Fig. 7a (Paxson's algorithm) and Fig. 7b (Zhang's algorithm) illustrated. However, we can see that SST (Fig. 7c) also does not function normally because of the drift and the extremely high q-value, as we predict in the previous section. The length of the intervals divided by PRCSEA when using MST (Fig. 7d) relates to the speed of skew changes. If the speed is fast, then the length is small and vice versa, which is a desirable property for non-constant skew estimation algorithms. We can also see, even with high q-value, the answer of PRCSEA armed by MST is satisfactory. Moreover, the interval length in the right chart relates to the speed of skew changes. At the time from 24 to 30 in the X-axis where the bottom of delay points almost forms a straight line and the skew is almost constant, the interval length is about 6 h. At the time around 44 in X-axis where drift is large, the interval length is about 1.5 h. These observations validate our assertion in Section 5.3.

7. Conclusions

This paper is the result of an attempt to resolve a Gordian knot in estimating computer clock skew. We present a general model that turns the estimation of clock skew to the solving of n-dimension equation group. We give 3 equation groups corresponding to the following 3 cases: zero clock drift with no clock adjustment (equation group 4), zero clock drift with clock adjustments (equation group 9), and non-zero clock drift with clock adjustments (equation group 12). The existing clock skew estimation algorithms mainly focus on the resolving of equation group 4, Paxson and Zhang et al. resolve equation group 9 via extra presumptions. PRCSEA solves equation group 12 without increasing complexity, and becomes the only skew estimation algorithm that can deal with non-zero clock drift. Furthermore, PRCSEA has many good features compared with related works, such as flexibility, robustness and reliability, while its time complexity is no more than theirs. Finally, measurements show that PRCSEA performs well in realistic scenarios.

Acknowledgements

This work has been supported through funding by National Natural Science Foundation of China Grant No. 90104006.

Special thanks are due to Xing Fang at the School of Engineering at Purdue University for his helpful comments.

References

[1] K.G. Anagnostakis, M. Greenwald, R.S. Ryger, cing: Measuring Network-Internal Delays using only Existing Infrastructure Proceedings of IEEE Infocom'03, San Francisco, U.S. 2003.

[2] G. Almes, s. Kaidindi, M. Zekauskas, A One-way Delay Metric for IPPM IETF Request For Comments 2679, Advanced Network and Services, 1999.

[3] J. Bi, Q. Wu, Z. Li, Packet delay and packet loss in the internet Proceedings of IEEE Symposium on Computers and Communications, Taormina, Italy 2002 pp. 3–8.

[4] S. Kalindidi, M.J. Zekauskas, Surveyor: An Infrastructure for Internet Performance Measurements Proceedings of INET'99 1999.

[5] D.L. Mills, On the accuracy and stability of clocks synchronized by the Network Time Protocol in the Internet system, ACM Computer Communication Review 20 (1) (1990) 65–75.

[6] D.L. Mills, Network time protocol (Version 3) specification, implementation IETF Request for Comments 1305, University of Delaware, 1992.

[7] D. L. Mills, "Modeling and analysis of computer network clocks", Technical Report 92-5-2, Electrical Engineering Department, University of Delaware, May 1992.

[8] S. B. Moon, P. Skelly, D. Towsley, "Estimation and Removal of Clock Skew from Network Delay Measurements", Proceedings of IEEE Infocom'99.

[9] V. Paxson, End-to-End routing behavior in the internet, IEEE/ACM Transactions on Networking 5 (5) (1997) 601–615.

[10] V. Paxson, "Measurements and Analysis of End-to-End Internet Dynamics", Ph.D. thesis, University of California, Berkeley, April 1997.

[11] V. Paxson, On Calibrating measurements of packet transit time Proceedings of ACM SIGMETRICS'98, Madison, Wisconsin 1998 pp. 11–21.

[12] A. Pasztor, D. Veitch, PC Based precision timing without GPS Proceedings of ACM SIGMETRICS'02, California, U.S. 2002 pp. 1–10.

[13] M. Tsuru, T. Takine, Y. Oie, Estimation of clock offset from one-way delay measurement on asymmetric paths 2002 Symposium on Applications and the Internet (SAINT) Workshops, Nara Japan 2002 pp. 126–133.
[14] H. Uijterwaal and O. Kolkman, "Internet Delay Measurements using Test Traffic", Technical Report RIPE-158, RIPE NCC, June 1997.
[15] L. Zhang, Z. Liu, C.H. Xia, Clock synchronization algorithms for network measurements Proceedings of IEEE Infocom'02, New York, U.S. 2002.
[16] R. Prasad, C. Dovrolis, M. Murray, K.C. Claffy, Bandwidth estimation: metrics, measurement techniques, and tools, IEEE Network 17 (6) (2003) 27–35.
[17] Ramesh Govindan, Vern Paxson, Estimating Routing ICMP Generation Delays Proceedings of Passive and Active Measurement Workshop 2002, Fort Collins, Colorado USA 2002.
[18] K. Papagiannki, S. Moon, C. Fraleigh, P. Thiran, F. Tobagi, C. Diot, Analysis of Measured Single-Hop Delay from an Operational Backbone Network IEEE Infocom 2002, NY, USA 2002 pp. 535–544.

Jingping Bi, associate professor, IEEE member. She received the Ph.D. degree in computer application in Institute of Computing Technology, Chinese Academy of Sciences in 2002. Research interests are network measurement, monitoring and management, IPv6 and next-generation Internet, performance evaluation.

Qi Wu, Ph.D. candidate. His research interests include network test and measurement, Internet routing, traffic modeling and prediction, clock synchronization, P2P network.

Zhongcheng Li received the B.S degree in computer science from Peking University in 1983, and M.S and Ph.D. degrees in computer science from Institute of Computing Technology (ICT), Chinese Academy of Sciences in 1986 and 1991, respectively. Since 1986, he has been with the ICT. From 1996 to 1997, Dr. Li worked in EECS Department, University of California at Berkeley as a visiting scientist. He is currently the director and a full professor of Information Network Division, ICT. Dr. Li has authored/co-authored over 70 technical papers and has been awarded twice Second-Level Prize of Natural Science from Chinese Academy of Sciences (1992, 1999). He currently serves as the chair of Technical Committee on Fault-Tolerant Computing, China Computer Federation and serves on the editorial board of Journal of Computer Science and Technology, Chinese Journal of Computers and Journal of Computer-Aided Design and Computer Graphics. His research interests include next-generation Internet, network test, measurement, and dependable systems.