A Brief Study of Time

By

Florian Buchholz and Brett Tjaden

From the proceedings of

The Digital Forensic Research Conference

DFRWS 2007 USA

Pittsburgh, PA (Aug 13th - 15th)

available at www.sciencedirect.com

journal homepage: www.elsevier.com/locate/diin

A brief study of time

Florian Buchholz[*], *Brett Tjaden*

Department of Computer Science, James Madison University, Harrisonburg, VA 22807, United States

Keywords:
Computer forensics
Computer clocks
Time synchronization
Time measurement
Timestamp correlation

ABSTRACT

In this paper we describe the first large-scale, long-term study of how hosts connected to the Internet manage their clocks. This is important for forensic investigations when there is a need for correlation of events collected from disparate sources, as well as for the correlation of computer events to "real" time. We have sampled over 8000 web servers on the Internet on a regular basis for a period of over six months. We have found that only about 74% of the hosts we observed were within 10 s of our reference time (UTC). The other hosts exhibited a large variety of different clock behaviors, some of which are explainable by existing clock models, some not, warranting further research in the area of forensic time and clock analysis.

1. Introduction

The reconstruction of events as part of computer forensics and incident response can involve events from only a single system, as well as events obtained from multiple, geographically separate sources, each with its own clock. An especially useful technique for event reconstruction is "time-lining". Here, discrete events that have a timestamp associated with them are ordered into a timeline. Timestamps can be obtained from file system metadata, system logs, or application data. Depending on the source of the events, this can provide a detailed sequence of the events that took place on a system (or multiple ones), allowing an investigator to reconstruct the sequence of events that took place.

When considering timestamps that were recorded by a computing system as evidence in an investigation, several factors need to be considered: the time on a computing system is kept by the system hardware clock, and in some cases by an additional software system clock. Depending on the accuracy of these clocks, how they were initialized, and whether they are synchronized, the clock(s) may differ quite considerably from the "real" time. Furthermore, clocks may be misconfigured to be in the wrong time zone or be set to the wrong time, clocks may be manipulated arbitrarily, and a clock may run fast or slow (*clock skew*).

The goal in forensic event reconstruction is to sequence all events that are important so that cause–time relationships may be established for an investigation. If all events we need to reconstruct are timestamped by the same system clock, some of the factors discussed above may not matter as the times will differ from "real" time in a consistent manner and a sequence of events can be established. The possibility of clock manipulation, however, is a factor that always needs to be considered. In particular, when clocks are set back in time, an identical time may be recorded for events that occurred before and after the clock change. Furthermore, few computing systems are completely isolated, and external timestamps are introduced in many ways. For example, information from e-mail headers or HTTP cookie data may play an important role in an investigation, at which point the external and internal timestamps need to be correlated. Also, whenever digital evidence is used to establish or support the point in time when events in the physical world occurred, the timestamps of the systems that are involved need to be translated to this reference time.

[*] *Corresponding author.*
E-mail addresses: buchhofp@jmu.edu (F. Buchholz), tjadenbc@jmu.edu (B. Tjaden).

1742-2876/$ – see front matter © 2007 DFRWS. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.diin.2007.06.004

In this paper we present a study that aims to give us a better understanding of the problems that need to be considered when dealing with computer clocks. Over the course of six months we remotely measured the clocks of over 8000 servers on the Internet. For this purpose, we used two different methods to "measure" time: ICMP and IP timestamp requests and HTTP. The goal of the study is to establish how computers connected to the Internet actually manage their clocks. The study we present here is the first large-scale study of how computers' clocks behave over a long period of time. By finding out how different computer clocks behave we can quantify whether the hypothetical time correlation problem described above is likely to be of practical concern. Another reason for the study is to establish ways to measure the clocks of remote hosts so that any techniques we develop from that may be used in forensic investigations.

The following section will give an overview of the problems we are trying to address with our research. Section 3 discusses related work, Section 4 describes how we conducted our study, in Section 5 we present our results, and we give conclusions and discuss future work in Section 6.

2. Problem description

In digital forensic investigations, knowing the correct time when events occurred can be of great importance. This could be because of the need to correlate events from different systems, establish alibis, or to find out when events occurred with respect to events in the real world. The importance of knowing the correct time has been previously addressed by Boyd and Foster (2004) and Weil (2002), actual investigations (CBS News, 2005), as well as the analysis of the 2003 power blackout in the Northeastern United States (Symmetricom_Power Grid; McAlpin, 2003; Koff, 2003).

Ideally, we would like all computer clocks to be synchronized to some common reference time, such as the Universal Coordinated Time (UTC) (United States Naval Observatory, 2003). However, this may not always be the case, as computer clocks inherently go slightly slower or faster than "real time" based on the quality of their quartz crystals or environmental factors, such as temperature (Symmetricom_Stochastic Model). Because of this and the fact that clocks may easily be set to arbitrary values, it is quite possible that computer clocks do not show values anywhere near real time. Instead they would exhibit different time values at different points in (real) time. When we sample these values over a long period of time, they make up what we call a *clock description* with respect to the reference time (normally UTC).

One method to overcome this deficiency of computer clocks and keep them close to real time is to periodically synchronize them with a computer's clock that is more accurate. The higher accuracy of these *time servers* is achieved through specialized hardware such as GPS receivers or atomic clocks, and the most commonly utilized method of time synchronization over a network is the Network Time Protocol (NTP) (Mills, 1992).

If all computers synchronized their clocks with a suitable time server using NTP, forensic event and time correlation would be easy. It is, however, unlikely that all computers perform this kind of clock synchronization. Furthermore, there is currently no data on how many hosts on the Internet do synchronize their clocks with a time server. To address this shortcoming, this is the first large-scale study of its kind, and it will give an indication of how many computers connected to the Internet have clocks that differ significantly from UTC. There have been previous surveys about the accuracy of the NTP network (Guyton and Schwartz, 1994; Minar, 1999), but these deal with measuring the quality of the information various NTP servers deliver. In our study we investigate web servers, which are generally of more interest to forensic investigations, because timestamps from web servers often can be found on client machines, and HTTP is a widely used protocol in many aspects of computing today. We will show below that there are a large number of servers on the Internet that do not synchronize their clocks. This is a finding that underlines the importance of performing research in time correlation and mapping for the purpose of digital forensic investigations. This is especially important when considering the analysis of disparate sources of evidence, where timestamps were recorded by different clocks.

The second reason for this study lies in the need to understand how computer clocks behave. To translate timestamps encountered on a host back to UTC, both Stevens (2005) and (Buchholz, unpublished data) have developed clock models that "describe" a host's clock with respect to a reference time. We will describe those models in more detail in Section 3. Buchholz predicts that clock descriptions will exhibit a pattern of linear clock skew combined with possible discrete "jumps" in the clock value, but except for a small number of cases, it is unknown how those computer clocks that do not synchronize their time actually evolve with respect to UTC. We look to either confirm the prediction made by Buchholz or to see if there may be additional factors that need to be taken into account when trying to obtain a description of a computer clock.

As another goal of our research we want to establish ways to reliably measure a remote computer's clock over the network. This may be important to a forensic investigation, because frequently timestamps from sources external to the system may be found. Such timestamps may include cached HTTP data or cookies, e-mail headers, or any other data saved from network communication. If clock descriptions of such external clocks can be obtained, it could be a valuable source of independent evidence that can be used to establish or confirm local timestamps. Of course, obtaining a clock description of an external host for a time frame that lies in the past may be impossible. However, if our research should reveal that clocks behave in a predictable manner, it may be possible to measure the external host at a later time and then make predictions as to how that clock behaved in the past. While this will not prove that a clock indeed behaved that way, if the predictions are consistent with the local evidence it can be used as an independent source to increase the overall confidence the investigator has in the evidence.

A second reason one may want to measure the clocks of remote computers is to monitor one's own machines within a network. If a dedicated machine keeps clock information of all other computers on the network, we have additional evidence that can be used in forensic investigations and intrusion detection. First, full clock descriptions of all machines on the network will be available to an investigator. Similar to an external log server, the clock information can be more trustworthy than any information on a compromised machine. Second, the clock monitor can look for ‘‘unusual’’ clock behavior for the purpose of intrusion or misuse detection. If an attacker on a local machine modifies the system clock to hide his tracks, this can be detected by the monitor and an alarm may be generated.

If we want to reliably monitor the clocks of remote hosts, we need to determine the following:

- What methods can we use to measure time of remote hosts on the Internet?
- What kind of accuracy can be achieved?
- How much data need to be stored?
- Looking at the observed clock behavior, what if anything can be deduced about the past clock behavior of that host?

In this paper we will only address the first item. Should we find that investigating computer clocks remotely is feasible and that timestamp correlation is problem likely to occur in practice, then the remaining items need to be addressed in our future research.

3. Related work

In this section we describe previous work in the field of measuring or synchronizing computer clocks that relates to our study. We will briefly talk about clock synchronization and then give an overview of the two clock models that are concerned with forensic time correlation, which may benefit from the results of this study. Finally, we present previous research where small numbers of computer clocks have been remotely monitored.

3.1. *NTP time synchronization*

The problem of synchronizing time on distributed systems is not new, and has been researched before it became relevant for forensic investigations. Lamport and Melliar-Smith give one of the earliest discussions about clock synchronization in the context of distributed multiprocess systems (Lamport and Melliar-Smith, 1985).

The Network Time Protocol specified by Mills is explicitly designed to keep a group of networked hosts' clocks synchronized to within a certain range (Mills, 1992). For this purpose, a time server is considered to be the authority that supplies all its clients with the valid time. On the client side, an NTP daemon runs on the host and manipulates the system clock. So as not to constantly have to communicate with the time server, the NTP daemon estimates the client's clock skew and only synchronizes with the server when deemed necessary. When synchronizing their times, the client has to take any network delays into account during communication with the server.

Generally, when utilizing NTP, a client's clock is very close to its time server's reference time. However, Buchholz identified small irregularities in the measured times of a synchronized host (Buchholz, submitted for publication). Even when NTP is utilized, there may be small discrepancies in the clocks between different computers. While the time differences are in the millisecond range this may be a problem if a high degree of accuracy is needed, potentially making time correlation difficult again. For most forensic investigations this may not be important, but when looking at the analysis of the 2003 power blackout, about 10,000 separate incidents were recorded spanning a mere 9 s (McAlpin, 2003; Koff, 2003).

3.2. *Measurements of computer clocks*

Large-scale studies of the Network Time Protocol network have already been conducted. Guyton and Schwartz (1994) describe a methodology to query the NTP hierarchy of different Stratum-level NTP servers, and the most recent study of the NTP network was conducted by Minar (1999). They discovered great discrepancies among the NTP servers with times being months and even years in the extreme. But even though Minar sampled over 175,000 servers the study can only be used to question the quality of a clock synchronized via NTP. From a forensics perspective we further want to study these additional factors:

1. How do those computer clocks behave that do not utilize NTP and how large is that number of hosts? While it is important to note that even when utilizing NTP a computer's clock may be wrong, the digital forensic community needs to be aware of the quality and quantity of the other hosts, as well.
2. How do clocks behave (change) over time? All previous clock studies that we are aware of merely sample each host once and compare it to a reference time. Our study sampled each host once every day (and a small number of hosts every hour) over the period of six months. This may give insights of how certain clocks behave in general, how frequently they are set to new times, or whether NTP is enabled or disabled for whatever reasons. In some future research some of those hosts may be classified and predictable behavior be qualified, which may allow a forensic investigator to estimate how a clock may have behaved in the past.

Paxson was one of the first to address the problem of computer clocks skewing off synchronized time (Paxson, 1998). He noticed misconfigured clocks having an adverse effect when trying to measure delay experienced by network packets. He also notes that even when hosts are synchronized utilizing NTP, some of these adverse effects persist. This supports the notion that NTP can only provide a certain precision not sufficient for some situations.

Kohno et al. measured computer clocks so that their clock skew could be used to uniquely identify the hosts on the Internet (Kohno et al., 2005). They established that hosts could be identified using their clock information even when physically moved to different locations or placed behind a firewall using NAT. While they mention the use of ICMP timestamp packets to measure time remotely as we do in this study, they focus almost exclusively on the TCP timestamp options. Given that the system clock is not used to create these timestamps, but rather a network-stack internal clock, the kind of measuring they describe will not work for the goals we pursue with our research.

In the context of forensic time correlation, Schatz et al. observed the computer clocks of a small network to see if web browser information can be used to put bounds on events (Schatz et al., 2006). Also, Buchholz performed some controlled measurements of just a single machine's clock to determine if clock skew is indeed linear, and what effect factors such as system load, turning the computer off, or using NTP have on a computer's clock (Buchholz, submitted for publication). In either case, these are measurements of small scale that do not allow us to make general observations or give us broad insights about how computer clocks may behave.

3.3. *Describing clocks and bounding events*

When a forensic investigator needs to determine to what time a computer's clock was set at a given time in UTC, or find out at what time a timestamp recorded on a system really occurred, he/she needs to utilize either a clock model or time bounding techniques.

The clock model introduced by Stevens (2005) takes a reference time (usually UTC) as a base, and then the investigator has to specify offsets to the reference time for any clock he/she wishes to describe. These offsets can be dynamic in nature, although Stevens does not elaborate on how to define an offset that changes its value over time. This model has its strength in showing what values a computer's clock had over time, but to map a host timestamp back to its reference time, the inverses of all offsets need to be computed and applied to the timestamp. Furthermore, Stevens neglects the fact that a timestamp value found on a host may map back to several times in the reference time, which is a result of a host clock being set back in time.

The model described by Buchholz (submitted for publication) uses a graphical approach to describe a computer's clock, and from there he derives formulas to quickly map timestamps back to the possible reference times, taking the possibility of a 1:*n* mapping into account. The model is based on only one dynamic component, clock skew, and discrete points in time when the clock is explicitly modified by external factors (such as a program setting the clock). Fig. 1 shows a simplified example of how Buchholz predicts clocks will generally behave. Here we see a clock description of a host with clock skew *s* and two instances where the clock was modified. It is one aspect of our study to determine if this is the general behavior of the clocks we observe.

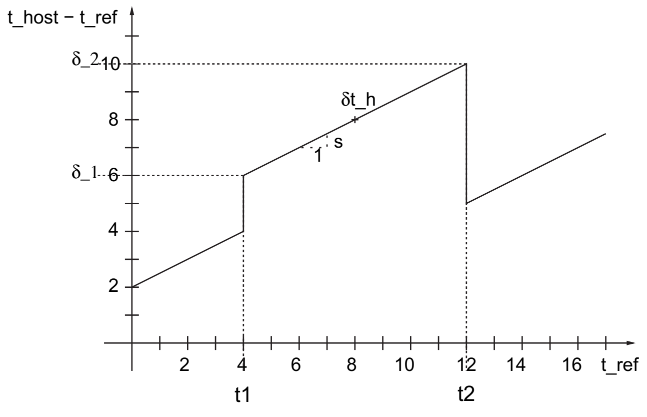

Fig. 1 – A simple clock description.

Time bounding techniques, such as described by Gladyshev and Patel (2005) try to establish a causal order between individual events and thus an overall order of all events on the system. If some of those events can be attributed to a time, it may be possible to define ranges of times when other events must have happened. Time bounding techniques will most likely be used in conjunction with a clock model: the model will provide some of the timestamps associated with events while the time bounding technique can be used to verify or disprove that the clock description that is derived for a computer is consistent with the evidence found during the forensic investigation.

4. Experimental setup

To begin our investigation, we first needed a list of a wide variety of Internet hosts whose clocks we could sample. We obtained a list of domain names from the DMOZ Top Listed Domains website. Adding the prefix ''www'' before each domain name gave us 8329 unique fully qualified domain names (FQDN). Resolving each of these FQDNs into its IP address(es) using DNS resulted in 8410 unique IP addresses. This is due to the fact that some FQDNs resolved to multiple different IP addresses while some unique FQDNs resolved to the same IP address. This list of 8410 unique IP addresses that we generated included machines located all over the world in the .biz, .com, .edu, .gov, .mil, .org, and other domains.

4.1. *Using HTTP to sample the clock on remote systems*

We set up a Linux host named chronos from which to run our experiments. This machine is configured using `ntpd` to synchronize its time with the official JMU time server. We wrote a tool, called web-time, that ran on chronos each night. For each IP address on our list, web-time would first take a reading of chronos' clock using the ftime() function which returns a structure containing the current date and time (including milliseconds). Web-time next sent an HTTP request to the IP address from the list, and used ftime() to take another clock reading when the HTTP response was received. Web-time then computed the difference (in milliseconds) between the timestamp returned in the ''Date'' header field from the HTTP reply and the midpoint between the time the HTTP request was sent and the time the HTTP reply was received. This value was then rounded to the nearest second since the ''Date'' field in the HTTP reply header contains a date and time including seconds but not milliseconds. In this manner, the web-time program used HTTP to measure the time difference in seconds between our local machine, chronos, and each of the 8410 remote hosts on our list. The ''Date'' header field was added for the HTTP 1.1 protocol and is mandatory in all HTTP replies. This is used for caching purposes on most HTTP clients (Fielding et al., 1999). It is therefore of interest to somebody who maintains a web server to keep the time returned in this field accurate so that clients use cached values rather than generate traffic for the server.

One measurement was made of each remote host every night so that we could plot the drifts of their clocks relative to ours over time. It took approximately 45 min each night to sample all 8410 hosts one after another using web-time. More than 90% of the 8410 hosts on our list routinely answered. Less than 1% of the hosts we tested either never answered or never returned a "Date" field in the HTTP header. An analysis of the results we obtained using web-time is given in Section 5.

4.2. *Using IP/ICMP to sample the clock on remote systems*

A separate tool that we used to sample the time difference between chronos and the 8410 hosts on our list was clockdiff – part of the iputils package. Clockdiff sends multiple IP or ICMP packets to a remote host and attempts to measure the difference in time between the local host and a remote host to the nearest millisecond. By default, clockdiff sends ICMP timestamp request (type 13) messages and expects ICMP timestamp reply (type 14) messages in response. Clockdiff also supports sending two different versions of the IP timestamp option so that time differences can be measured on remote hosts that do not receive/answer ICMP requests. Using any of these three options, clockdiff can estimate the difference between the time on the local host and the time on a remote host to the nearest millisecond. However, due to the limited length of the fields in the ICMP and IP timestamp messages, all values reported by clockdiff are modulo 24 days. In other words, if a remote clock was exactly 12 days and 1 ms behind the local host, clockdiff would report the time difference as −1 ms. So clockdiff gives a fairly fine-grained estimate of the time difference between two machines as long as they are roughly within a week of each other.

As with the HTTP experiment described above, we used clockdiff to sample the time difference between chronos and all 8410 hosts on our list once a night for several months. Due to the multiple packets that clockdiff sends and receives, it took several days to sample all 8410 hosts using clockdiff. About 41% of the hosts on our list (3471 out of 8410) responded to the default ICMP request by clockdiff. This was actually a much higher success rate than we had expected as we thought that most sites would block unnecessary ICMP traffic for security reasons. An additional 540 hosts responded to the first IP option used by clockdiff, and another 402 replied only to the second IP option used by clockdiff. So clockdiff could only be used on 4413 of 8410 hosts on our list (3471 that answered ICMP, plus 540 that did not answer ICMP but answered IP option 1, plus 402 that only answered IP option 2). We removed the 3997 hosts from our nightly run list for clockdiff (but kept all 8410 on the web-time run list) and only sampled the hosts with clockdiff each night that we knew would answer. This resulted in us being able to sample each of the 4413 hosts using clockdiff once every day for several months. Each day's clockdiff sampling run took approximately 18 h to complete.

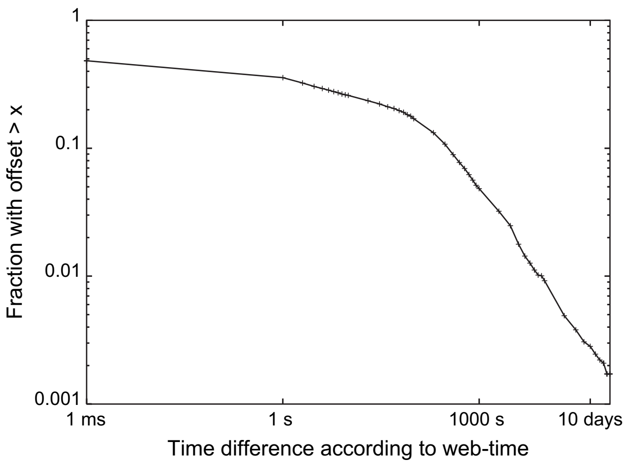

Fig. 2 – The cumulative distribution function for all hosts (measured with web-time).

5. Analysis of results

We began to analyze the data we were collecting each day by creating graphs of the cumulative distribution functions for both our web-time and clockdiff data to try to understand how closely synchronized all the hosts we were measuring were as a group. Fig. 2 shows results of one run performed October 18, 2006. On that night, 8410 hosts were queried using our web-time program. A total of 8149 of those hosts responded with a valid timestamp, and 261 hosts either did not respond at all, or replied without a valid timestamp in the response.

Approximately 74% (6040 out of 8149) of all hosts that responded were within ±10 s of our reference time. The other 26% (2109 out of 8149) that responded were more than 10 s out of synchronization – some by seconds, some by minutes, some by hours, some by days, some by weeks, some by months, and some by years. About two thirds (5510 of 8149) of the hosts that responded were tightly synchronized to within ±2 s of our reference time, and a little more than half (4213 of 8149) showed a time difference of 0 s as measured by web-time. We were a bit surprised to see only 50% of the hosts with no time difference from our machine. Before running the experiment we had expected that this number would be much higher as these are popular Internet web servers and we had expected the vast majority to be tightly synchronized. We were even more surprised at the large percentage of hosts (approximately 26%) that were off by more than 10 s.

One interesting phenomenon that we noticed is that more hosts' clocks were behind our reference time than were ahead of it. We observed that 4213 hosts differ from our reference time by exactly 0 s. Of the remaining 3936 hosts that answered, 2315 were behind our clock by one or more seconds, and 1621 were ahead of our clock by one or more seconds. Moreover, the clocks that were behind were further behind (on average and at maximum) than the clocks that were ahead. Of the 1621 clocks that were ahead, the average number of seconds that they were ahead by was 1348 (about 22 min). For the 2315 clocks that were behind, the average number of seconds that they were behind was 1,876,820 (about 21 days). The five clocks that were set furthest in the past gave readings (on October 18, 2006) of:

- October 18, 1906,
- January 1, 1970,
- May 19, 2006,
- July 12, 2006,
- October 6, 2006.

Clearly, the first two clocks have strange, but explainable values. The first is exactly 100 years behind, and the second is stuck on a value corresponding to 0 ms past the beginning of 1970 (this means that likely a value of 0 is returned for the timestamp as this corresponds to the beginning of the UNIX time epoch). Even if we discard these first two extreme values from the list of clocks running behind, the remaining 2313 slow clocks are, on average, 12,029 s (almost three and a half hours) behind. By comparison, the five clocks that were set furthest in the future gave readings (on October 18, 2006) of:

- October 18, 2006 (14 h ahead),
- October 18, 2006 (16 h ahead),
- October 19, 2006 (17 h ahead),
- October 19, 2006 (24 h ahead),
- October 19, 2006 (24 h ahead).

The results presented above are typical of all runs done to date. Every day for the six months our experiment has been running, approximately 50% of hosts that respond to our web-time query show a time difference of 0 s. Some hosts are almost always in this group while other hosts are in it sometimes and not at others. Typically, about 75% of the hosts we test each night are within ±10 s of our reference time. As of March 31, 2007, the two unusual hosts discussed above have always been 100 years behind and stuck on midnight, January 1, 1970, respectively. There are other hosts whose clocks are days, weeks, months, or years behind although some of these clocks become synchronized occasionally, while other clocks that were roughly synchronized have been observed to jump backwards or forwards suddenly by several days, months, or years. We will illustrate this phenomenon and discuss the behavior of several interesting individual clocks in a subsequent section.

Since we had more than one mechanism for measuring the clocks on the remote hosts, we were interested to see how closely the time reading obtained with web-time would agree with the time reading obtained with clockdiff. As mentioned earlier, far fewer hosts replied to clockdiff requests than to web-time queries. On the nightly run on October 18, 2006, 3767 of the hosts queried with clockdiff responded with a valid timestamp. Fig. 3 shows the cumulative distribution function for the results.

About 74% (2789 out of 3767) of all hosts that responded were within ±10 s of our reference time. The other 26% (978 out of 3767) that responded were more than 10 s out of synchronization – some by seconds, some by minutes, and some by hours. As noted earlier, due to the size of the integer used for timestamps, clockdiff reports the difference in time modulo 24 days, so it is not possible to observe time differences greater than that using clockdiff. About two thirds

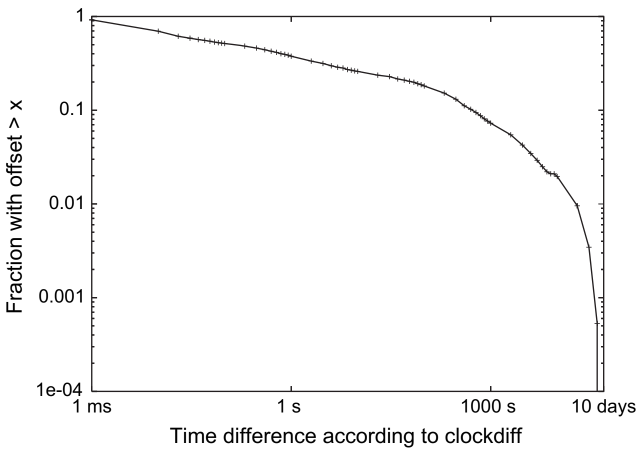

Fig. 3 – The cumulative distribution function for all hosts (measured with clockdiff).

(2508 of 3767) of the hosts that responded were tightly synchronized to within ±2 s of our reference time, and a little more than 57% (2161 of 2767) showed a time difference of between −500 and 500 ms as measured by clockdiff. Hosts (129) showed an offset of exactly 0 ms. This distribution is fairly similar to the one observed with web-time (though clockdiff was only able to measure the time on about half as many hosts as web-time).

As with the web-time measurements, clockdiff showed that more clocks were behind our reference time (2028 of 3638) than ahead (1610 of 3638). However, perhaps because of the 24 day modulus of clockdiff, we did not observe substantially larger average or maximal values. For the 2028 clocks that were behind, the average number of milliseconds that they are behind was 599,144 (about 10 min). For the 1610 clocks that were ahead, the average number of milliseconds that they were ahead was 721,994 (about 12 min). The five clocks that were set the furthest in the past according to clockdiff gave readings that were approximately 11, 10, 10, 10, and 9 h in the past. The five clocks that were set the furthest in the future according to clockdiff gave readings that were approximately 12, 10, 9, 9, and 8 h in the future.

The next question we explored was: for those hosts that answered both web-time and clockdiff requests, how closely did the two different clock measurements agree with one another? To answer this question we computed the delta (or difference) between the time offset (to our reference time) obtained using clockdiff and the time offset obtained using web-time. These two measurements were taken as close to one another as possible for each host every night. We were able to compute the delta between clockdiff and web-time on 3714 hosts, which responded with a valid timestamp to both requests. About 95% of the hosts (3527 of 3714) had a delta of less than 10 s. Note that this does not mean that all those hosts were synchronized with our reference time. It simply means that if we took a measurement of a host with clockdiff, and clockdiff reported that the host was 20 s behind our reference time, then the web-time measurement would have to say that the host was

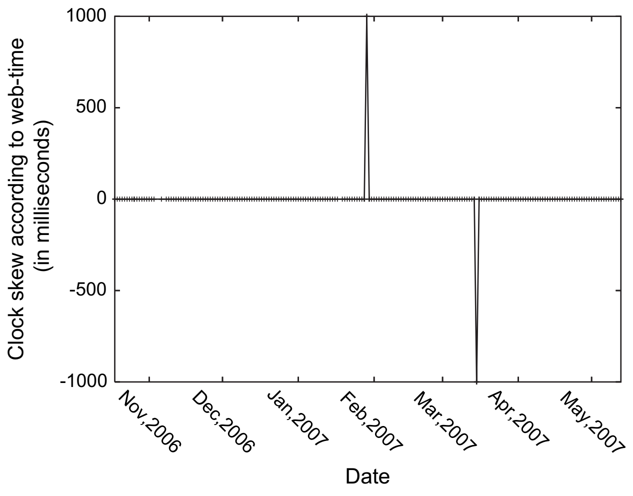

Fig. 4 – Time difference according to web-time (in ms).

between 10 and 30 s behind our reference time to be included in this group. About 92% of the hosts (3430 out of 3714) had a delta of less than 1 s.

So on most hosts it seemed that the two time measurement techniques were in at least rough agreement with one another. However, there were some notable exceptions. Fig. 4 shows the clock skew between one host and our machine, chronos, as measured by web-time over a period of about six months.

According to web-time, this host is very tightly synchronized with our reference time – the difference is almost always zero except for one day when the reading was -1 s and another day when the reading was $+1$ s. These two small losses of synchronization could be legitimate or due to a measurement error perhaps introduced by an unusually large network delay. Next we look at the clock skew graph for the same host, measured a fraction of second later each night using clockdiff for the same period of time (Fig. 5).

The measurement taken with clockdiff reports (night after night for months) that the clock is about 27 million milliseconds (about seven and a half hours) behind our reference time. Furthermore, clockdiff reports that the clock is losing a fraction of a second each day. This is a very different view of the remote system's clock than the one given by web-time. There are several possible explanations for this behavior. One explanation would be that the web server obtains its time from a different (and synchronized) time source that is not the system clock, while clockdiff giving us a reading from a poorly synchronized (and drifting) system clock. Or, perhaps, web-time is measuring the system time and clockdiff is getting a different time – possibly from a completely different machine that is responding to ICMP requests before they reach the target host. We have found examples on other hosts where the time reported by clockdiff seems to be more closely synchronized with our reference time and the time reported by web-time is very different. Certainly, additional remote time measurement mechanisms could give us further evidence about what the true time on a remote system might be.

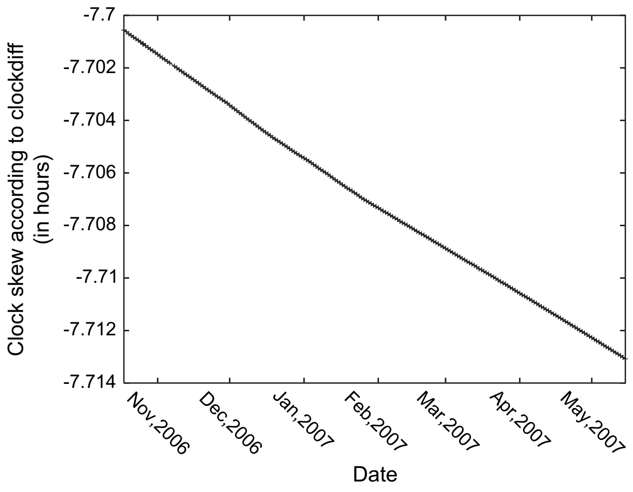

Fig. 5 – Time difference according to clockdiff (in h).

5.1. *Daylight Saving Time*

Daylight Saving Time (DST) is a clock adjustment that many countries observe to prolong the daylight for afternoons during the summer. Each year during the spring the clocks in those countries are set forward by 1 h, and in the fall they are set back by 1 h. UTC is not affected by Daylight Saving Time and thus we should not observe any changes in the clocks we monitored for our study during the Daylight Saving Time adjustments. This is because both ICMP/IP timestamp requests as well as the HTTP date field require that the reply is sent in UTC.

There is no global agreement of when those Daylight Saving Time adjustments are done. Essentially, each country may devise its own rules regarding DST. Europe, for example, switches to DST the last Sunday each March at 1 am, and changes back the last Sunday each October. The United States this year changed its DST adjustment times. Instead of switching to DST the first Sunday of April at 2 am, the switch now takes place the second Sunday in March. The change back now occurs the first Sunday of November instead of the last Sunday in October. This was put into effect as of Spring 2007 as part of the Energy Policy Act of 2005 (United States House of Representatives, 2005).

We were curious if any of the clocks we observed actually (incorrectly) show the DST adjustments in their clock behavior. This is especially interesting given that many computing systems needed a special patch to accommodate for the new rules for the United States DST switch. Our study currently spans a time period from mid-October 2006 through April 2007, which means that we cover both the switch back from DST in October 2006, as well as the switch to DST for both the US and Europe (including the date the US would have switched under the old rules).

For the switch back from DST in October we did not find a statistically significant difference in the number of hosts that were 1 h or more off our reference time. However, on March 11 (the day the US switched to DST under the new system) we observed an increase of roughly 20% (the number of hosts changed from a daily range of 180–190 to a range of 225–240). While the overall number of hosts where we observe

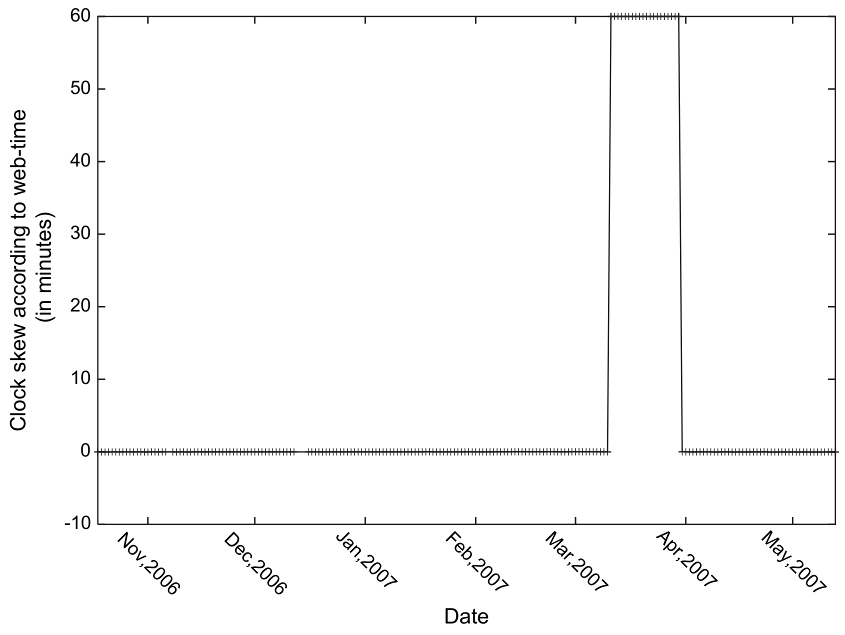

Fig. 6 – A host that does not follow the new DST rules for the US.

this phenomenon is low (at most 60 out of 8410), we should be aware of this. Fig. 6 shows a host belonging to a US university, whose clock jumps 1 h ahead of our reference time on Sunday, March 11th, and then is again synchronized with UTC on Sunday, April 1st. We believe that this is caused by the way UTC is determined on this host. If UTC is calculated as local time minus the offset, then this would explain why the host's clock would yield this behavior.

5.2. *Types of clock behavior*

In addition to examining the general trends of host synchronization on the Internet using clockdiff and web-time, and comparing the measurements we obtain with the two different methods when possible, we identified a number of interesting hosts that exhibited strange clock behavior over time. We are currently performing a more in depth study of these hosts to see if we can determine what is causing these strange behaviors. We discuss some of these interesting hosts below.

As mentioned previously, a fair number of hosts in our sample appear to have their clocks consistently set well in the past or slightly into the future. We also encountered some clocks that do not appear to be running (we are pretty sure that the system clocks on these hosts must be running since the systems are functioning, but they return the same timestamp in response to our queries night after night). We have already mentioned one of these systems previously – one that does not respond to clockdiff but has returned the timestamp (Jan 01, 1970 00:00:00 GMT) every time we have queried it for the past six months. There are other hosts that exhibit this same ''stuck clock'' behavior with different values. Fig. 7 shows the clock skew graph for a host that has been stuck on Jul 12, 2006 20:41:36 GMT for the entire time we have been observing it (though this host stop responding in mid-March).

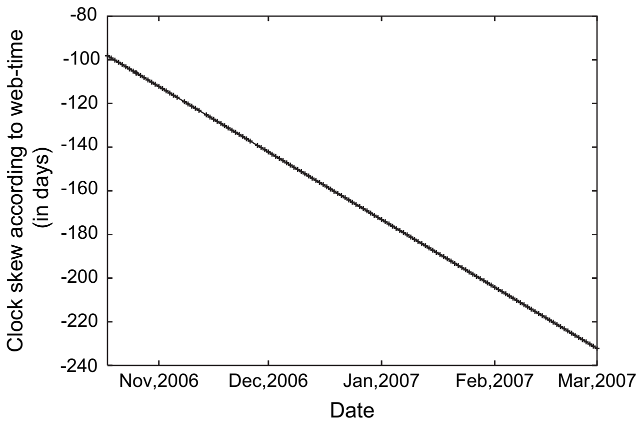

Fig. 7 – A stuck clock.

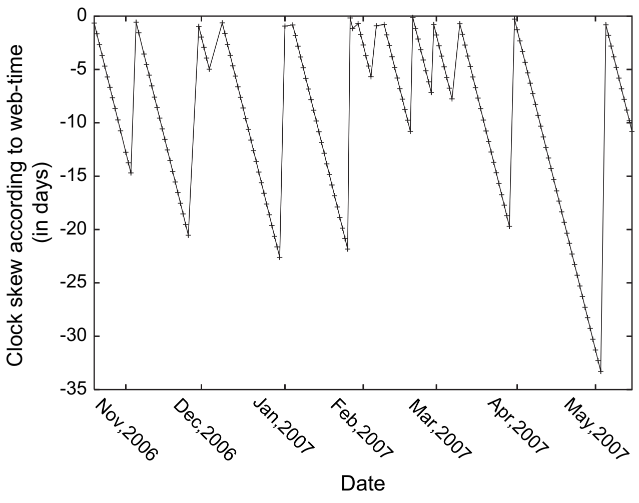

Fig. 8 – A sticky clock.

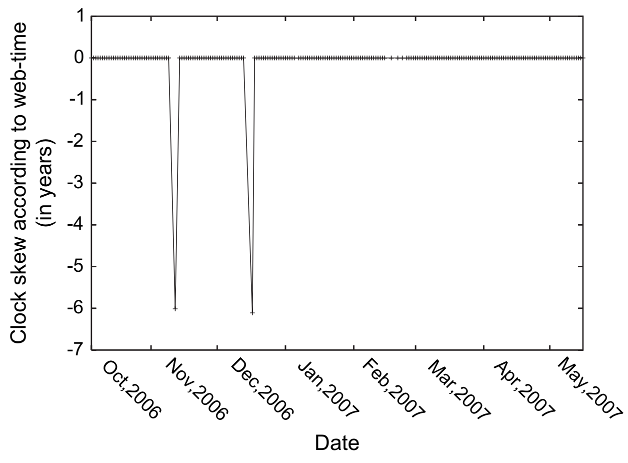

Fig. 9 – Clock jumps.

Obviously, this clock is losing one second per second every day. A related case is several "stuck clocks" that occasionally get unstuck. Fig. 8 shows a clock that synchronizes occasionally and then gets stuck for several days or weeks.

We also have observed a large number of clock jumps. Fig. 9 shows a clock that jumped more than six years into the past on two occasions.

Fig. 10 shows a clock that jumps about 6 h into the future quite often.

Fig. 11 shows a host whose clock was consistently about 12 days behind for several months. It was then synchronized for about a month, and is now about 5 days behind.

Other hosts displayed a consistent clock drift. Fig. 12 shows a clock that loses about 100 min each day but is occasionally synchronized back close to our reference time.

Other hosts show clock drift without synchronization. Fig. 13 shows a host whose clock loses about 4 s a day and has not been adjusted during the time we have observed it.

5.3. *Mapping to the clock model*

When looking at how well the sampled clocks correspond to the predictions of Buchholz's clock model, we get mixed results. Some clocks exhibit a description graph that directly maps to a behavior of linear skew with discrete jumps in the clock value.

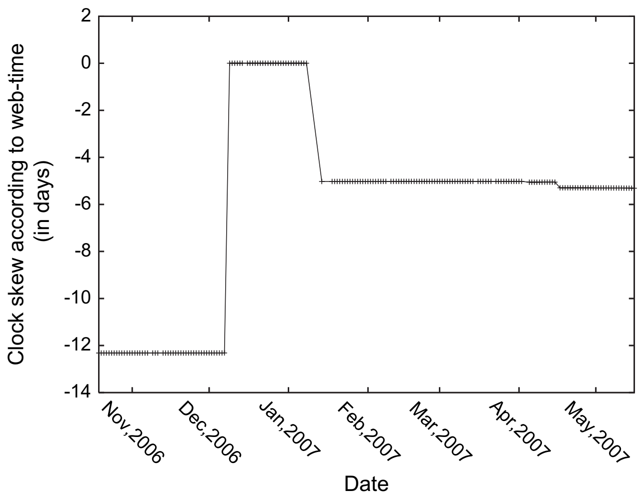

Fig. 11 – Clock jumps.

Fig. 14 shows a clock that has a negative linear clock skew, and we can observe four jumps of the clock value. Twice the clock jumps to a fairly high positive offset to UTC (over 50 min ahead). In both cases the clock jumps back after a few days to what appears to be synchronized time, and the negative skew continues.

The clock described by Fig. 15 exhibits a slightly different behavior, but is still consistent with the clock model. There is positive skew, and also a period in the beginning third of our measurements where the clock is mostly synchronized. During the "synchronized phase," however, the clock jumps ahead by 40 s and back to synchronized a few days later. In the later two thirds of our observation period there is consistent clock skew but the clock jumps back and forth between some seemingly consistent offsets. Also, it is interesting to see that the continuation of the slope of the first spike continues seamlessly with the top slope of the skew when the clock is no longer synchronized. Why this happens exactly

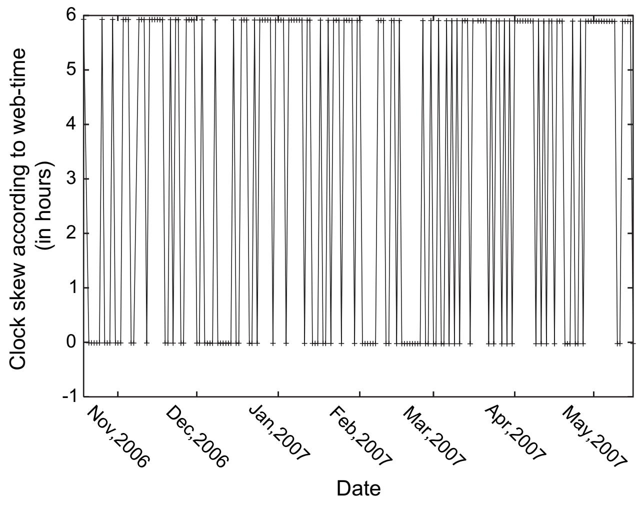

Fig. 10 – Jumpy clock.

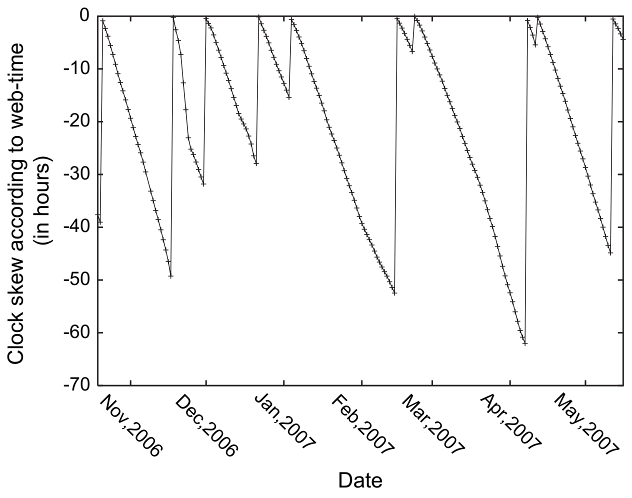

Fig. 12 – Clock drift with synchronization.

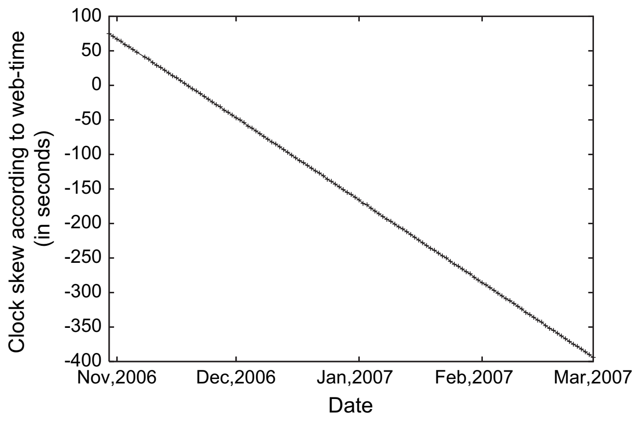

Fig. 13 – Clock drift without synchronization.

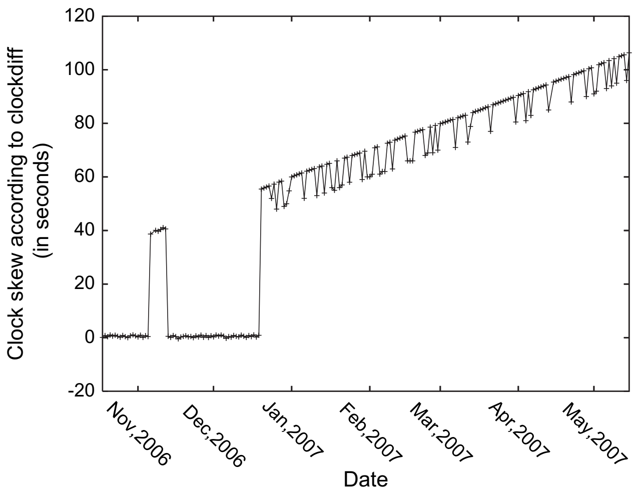

Fig. 15 – A clock that continues with its original skew slope when no longer synchronized.

we cannot tell. It may be that we are sampling two different hosts that are reachable by the same IP address, or it could be that the host was synchronized with NTP for a while and then stopped. This would not explain why the clock resumed at the continuation of the original skew, however, or why there are so many jumps. Further research will be needed to determine what is going on here.

The clock shown in Fig. 16 is again consistent with the clock model until toward the end of our observation period the clock skew changes. This is not supposed to happen for a hardware clock as Kohno et al. (2005) point out. The most likely explanation for this is that the hardware of the web server for this IP address was changed and the new clock has a different skew than the old one. Also, it could be that again multiple hosts take on traffic from this IP address.

Fig. 17 shows another host that conforms with Buchholz's clock model. Two things are remarkable about this clock: first, it seems to jump in fairly regular intervals. This does not always happen, nor are the jumps always of the same magnitude. However, the duration of many of the intervals where the clock drifts off without a jump is roughly about a week long. This could be the result of running a Microsoft Windows operating system (The web server is ''Microsoft IIS''). Hosts running Windows often periodically contact a time server to synchronize their clock. Why the clock is almost never synchronized with our reference time, we cannot say, though.

The second point of interest about this clock is that the graph in Fig. 17 is an ''hourly'' graph, meaning that we sample this clock every hour. The sample method for the hourly samples is clockdiff using IP option timestamps. In our daily samples, we also query this host, but here we use clockdiff ICMP requests to sample. The ''daily'' graph of this clock is shown in Fig. 18.

The clock graphs do not correspond at all. First, there are very few samples actually successfully taken of this clock using ICMP, and second, the scale of the two graphs is very different. Occasionally when we ran clockdiff with the ICMP option we received a message that the reply was in non-standard format. What we currently suspect is that some ICMP (and possibly IP) clockdiff replies return some random value, which sometimes may be interpreted as a valid timestamp. The resulting graph may then look like Fig. 18.

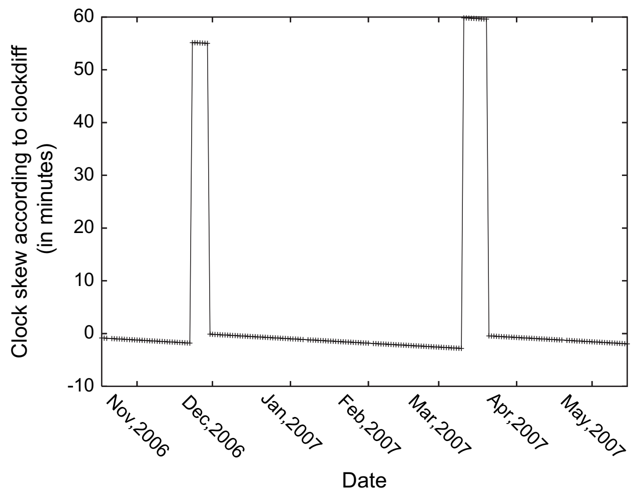

Fig. 14 – A clock behaving in accordance to the model.

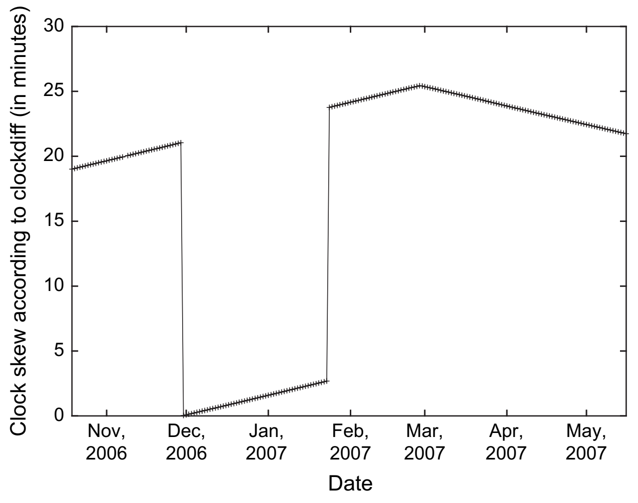

Fig. 16 – Did the hardware for this host change?

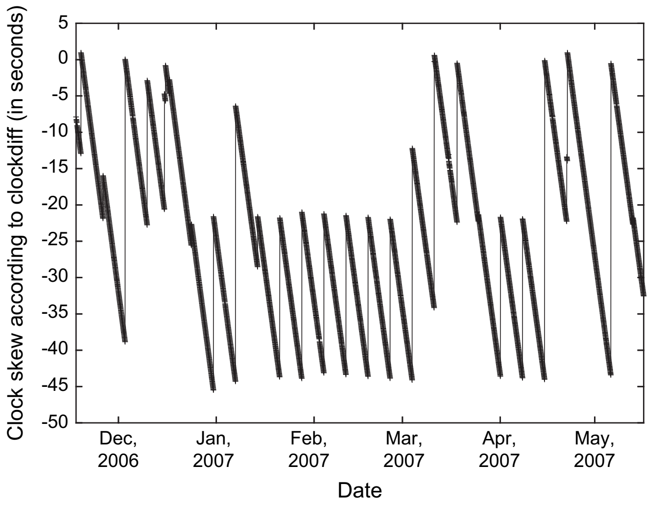

Fig. 17 – A clock with periodic jumps (sampled hourly).

Finally, Fig. 19 shows a clock that seems to have two different clock skews and a very large number of jumps. Both skews are negative and the one we can observe along the top edge of the graph is steeper than the skew at the bottom of the graph. It is very likely that in this case there are indeed two different hosts that handle requests for the same IP address and when we sample the clock we sometimes get an answer from one host, and sometimes an answer from the other. What we cannot explain at this point is why there is an overall positive progression for the top clock, a negative progression for the bottom clock, or why there is a time about in the middle of our observation period where both clocks seem to be synchronized. It could be that there is a common time server or domain controller that is responsible for this behavior.

6. Conclusions and future work

In this paper we have described the first large-scale study of how hosts connected to the Internet manage their clocks. Some of the surprising observations from our study include:

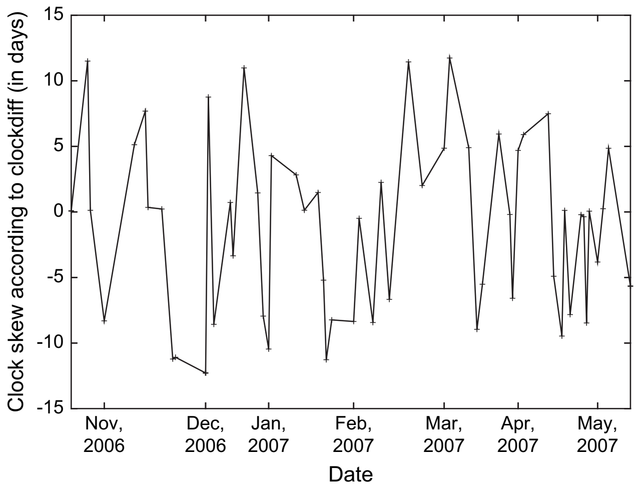

Fig. 18 – The same host as Fig. 17 sampled differently.

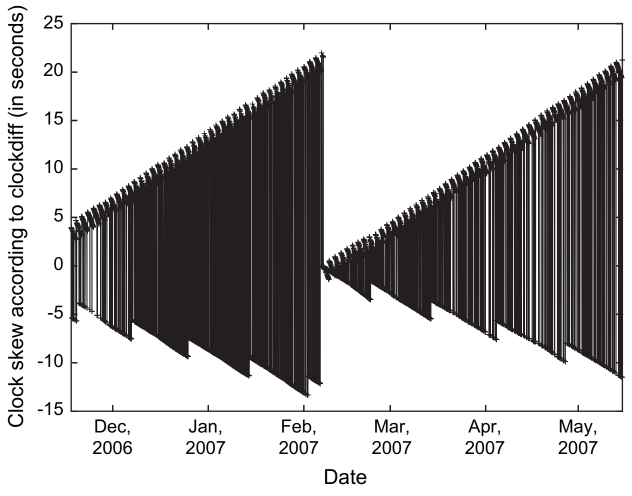

Fig. 19 – Two different hosts "sharing" an IP address?

- A significant number (26%) of the servers we observed differed by more than 10 s from of our reference time (UTC). Some of these hosts were days, months, or years off.
- Some hosts give one clock reading when queried with one of our measuring techniques and a very different reading when queried with another measuring technique.
- Some hosts' clocks fit well with established clock models, and others are inexplicable as of yet.

Our conclusions from this study are:

- More study in the area of clock behavior and time correlation is needed. Clearly, the lack of global agreement on time could potentially hamper forensic investigations in which there is a need for event correlation for events collected from disparate sources, as well as for the correlation of computer events with events in the real world. The large number of hosts that were not synchronized to UTC shows that there is a strong likelihood of event correlation problems when more than one host is involved in a forensic analysis. This is especially true if a high degree of precision is needed for the timestamps.
- Even though many of the clocks we observed were off our reference time, we could sometimes discover certain regularities in their behavior that may be useful to a forensic investigator who needs to correlate timestamps. Some hosts we observed changed their clock value in regular intervals, for others we could recognize a ''pattern'' with the human eye, and further research will be needed to mathematically explain these observations. Ideally, future research could establish that certain classes of hosts exhibit regular clock behaviors. This would enable an investigator to guess at past clock behavior for systems that are beyond the control of the investigation. This can then be used to strengthen the evidence observed locally.
- Additional tools are needed for measuring the time on a remote system over the Internet. Neither of the two tools we made use of, web-time and clockdiff, worked on all hosts. Furthermore, each tool displayed a different granularity of measurement and success rate at measuring hosts.

Additional methods of sampling a remote clock may be able to perform better measurement or at least give us additional evidence about the time on a remote system when performing a forensic analysis.

- Further study is needed of the unusual behavior we observed for some of the clocks. This may lead to refinement of existing clock models, or it may reveal peculiarities of how certain hardware, operating systems, or applications influence timestamps obtained from particular systems under investigation.
- A ‘‘clock logging’’ tool that observes clocks on a local (or remote) system can be devised using the methodology of our study and any future techniques that may be developed. Such a tool could centrally collect and store clock descriptions of important hosts on a network. Once an investigation is necessary, the data stored can be used as extra independent evidence. Any clock manipulations on a host should be observed given that the external logging facility is not also compromised. Thus such a clock observed could also be used in the area of intrusion detection generating an alert when ‘‘unusual’’ clock behavior is detected.

REFERENCES

Boyd Chris, Foster Pete. Time and date issues in forensic computing – a case study. Digit Investig 2004;1(1):18–23.

CBS News. The Alibi: reasonable doubt, <http://www.cbsnews.com/stories/2002/10/10/48hours/main525143.shtml>; January 2005.

DMOZ Top Listed Domains, <http://www.domaintools.com/internet-statistics/dmoz-listings.php> [accessed October 2006].

Fielding R, Gettys J, Mogul J, Frystyk H, Masinter L, Leach P, et al. Hypertext Transfer Protocol – HTTP/1.1. Technical Report RFC 2616, Internet Society; June 1999. <ftp://ftp.isi.edu/in-notes/rfc2616.txt>.

Gladyshev Pavel, Patel Ahmed. Formalising event time bounding in digital investigations. Int J Digit Evid 2005;4(2).

Guyton James D, Schwartz Michael F. Experiences with a survey tool for discovering network time protocol servers. In: USENIX Summer; 1994. p. 257–65.

Koff Stephen. Task force in blackout outlines grid failure. The Plain Dealer, <http://www. cleveland.com/blackout/index.ssf?/blackout/more/106335939217%2910.html> September 2003.

Kohno T, Broido A, Claffy KC. Remote physical device fingerprinting. In: IEEE symposium on security and privacy, Oakland, CA, USA, May 2005.

Lamport L, Melliar-Smith PM. Synchronizing clocks in the presence of faults. J ACM 1985;32(1):52–78.

McAlpin John P. U.S. energy secretary says weeks needed to analyze blackout data. The Standard Times, <http://www.southcoasttoday.com/daily/08-03/08-28-03/a19wn082.htm> August 2003 [AP Press].

Mills D. Network Time Protocol (version 3): specification, implementation and analysis. Technical Report RFC 1305, Network Working Group; March 1992.

Minar Nelson. A survey of the NTP network, <http://www.media.mit. edu/~nelson/research/ntp-survey99/>; December 1999.

Paxson V. On calibrating measurements of packet transit times. Meas Model Comput Syst 1998;11–21.

Schatz Bradley, Mohay George, Clark Andrew. A correlation method for establishing provenance of timestamps in digital evidence. In: Proceedings of the 6th annual digital forensic research workshop, August 2006.

Stevens Malcolm W. Unification of relative time frames for digital forensics. Digit Investig 2005;1(3):225–39.

Symmetricom: Timing, Test, and Measurement Division. How time finally caught up with the power grid, <http://www.symmttm.com/pdf/Gps/wp_PowerGrid.pdf>.

Symmetricom: Timing, Test, and Measurement Division. Stochastic model estimation of network time variance, <http://www.symmttm.com/pdf/Network_Timing/wp_Stochastic_Model.pdf>.

United States House of Representatives: Committee on Energy and Commerce. Energy policy act of 2005, <http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi? dbname=109_cong_bills&docid=f:h6enr.txt.pdf>; 2005 [public law 109-58].

United States Naval Observatory: Astronomical Applications Department. What is universal time?, <http://aa.usno.navy.mil/faq/docs/UT.html>; 2003.

Weil Michael C. Dynamic time & date stamp analysis. Int J Digit Evid 2002;1(2).

Dr. Buchholz received his Ph.D. in Computer Science in August 2005 from Purdue University, where he worked with his adviser, Eugene Spafford. He joined the faculty of the Computer Science Department at James Madison University in September 2005 as an assistant professor. His research focuses on Digital Forensics with an emphasis on time synchronization and event and data reconstruction, Operating System Security, and Network Traceback. He is teaching Computer Forensics and Distributed Security classes at JMU as part of the online Master's program with an emphasis in Information Security, as well as various undergraduate classes, including a Computer Forensics course.

Brett Tjaden received his Ph.D. in Computer Science from the University of Virginia in May 1997. From 1998 to 2002 he was an Assistant Professor in the School of Electrical Engineering and Computer Science at Ohio University. He is currently an Associate Professor in the Department of Computer Science at James Madison University where he teaches undergraduate and graduate courses on information security, network security, and secure operations.