TIME DISTRIBUTION USING THE WORLD WIDE WEB

Andrew N. Novick, Paul R. Franchois, Michael A. Lombardi
National Institute of Standards and Technology
325 Broadway
Boulder, CO 80305
303-497-3378
novick@boulder.nist.gov

Abstract - The National Institute of Standards and Technology (NIST) is currently providing users of the Internet with a running time-of-day clock on a web page. It uses a combination of Hyper Text Markup Language (HTML), Perl, Java and JavaScript to show the time for a chosen time zone. Processing and path delays are estimated to provide the client with an idea of how close the time displayed is to Coordinated Universal Time - UTC(NIST). Several future possibilities and uses of this technology will be discussed.

WHAT IT DOES

A client visits the web sites at either **http://www.time.gov** or **http://www.nist.time.gov** and is presented with a map of the United States (Figure 1). Time zones are shown in alternating colors, and outlines of the states are visible. As the cursor passes over different time zones, a message is displayed specifying which time zone the cursor is in. When the client clicks in a time zone, a running or "animated" clock with the correct time for the chosen time zone is displayed. Also, a gray-line map shows where the sun is shining on the Earth at the present time.

Figure 1 - The time zone map page.

HOW IT WORKS

The time zone map is an image map, displayed by an HTML page, and each time zone on the map is a link. When the client clicks a time zone link, a string of parameters is sent to a Common Gateway Interface (CGI) script written in Perl (*timezone.cgi*). Four parameters are sent: the name of the selected time zone (Pacific, Mountain, etc.), a character indicating if this particular time zone observes Daylight Saving Time (DST), the time offset the selected time zone is from UTC (in hours), and whether the site should display a running clock or a static clock that provides a "snapshot" of the time. A call to the CGI script might look like this:

http://www.time.gov/timezone.cgi?Eastern/d/-5/java

The parameters indicate that the Eastern time zone was selected. The "d" signifies that this time zone does observe DST – Hawaii, most of Arizona and parts of Indiana are special cases that don't observe DST. The "-5" indicates that Eastern Standard Time is 5 hours behind Coordinated Universal Time (UTC). The HTML code attempts to determine if the client's web browser supports Java. If it does, then the word "java" is included with the parameters and the server will display a running clock in the client's web browser. If it does not, then the word "java" is not sent to *timezone.cgi* and the server will display a static "snapshot" of the current time to the client. Likewise, if the client clicks on "DISABLE JAVA ANIMATION", the word "java" is also omitted from the string.

When it is launched, the *timezone.cgi* script first checks the current time and date by getting a time string from the web server it is running on. The web server clock is periodically synchronized with a NIST Internet time server. [1] The script then calculates if DST is in effect using the current DST rules for the United States. DST begins at 2:00

a.m. on the first Sunday in April and ends at 2:00 a.m. on the last Sunday on October. Since this transition does not happen at midnight, the code has to take into account the time of day, not just the date, when making the calculation. Also, because the DST rollover happens at 2:00 a.m. *locally*, the script has to determine whether the current time is before or after this time change for the specific time zone chosen. In April, clocks are moved an hour forward and in October, clocks are moved an hour back. Further, after the October change, a flag must be set that indicates the change has already happened. This is because when the clock has been set back an hour (to 1:00 a.m.), the time change would happen again at 2:00 a.m., getting stuck in an endless loop. The April time change does not have this problem because when the time reaches 2:00 a.m., it jumps to 3:00 a.m.

Next, the input string is checked to see whether it is a Java or non-Java request. If it is a Java request, the HTML file *clock_top.html* is sent to the browser, and then Javascript variables are setup so that the client will be able to call up the correct gray-line map for the current solar position. An additional HTML file, *clock_mid.html*, is then sent to the browser. It continues formatting the framework of the page in the browser.

The *timezone.cgi* file then sends the Java applet to the browser of the client. When the applet code contained in the file, *utcnist4.class*, is executed, it calls some methods in two additional files: *correctnist4.class* and *utcgate3.class*. For security purposes, it is verified that the applet originated from a NIST domain. At this time, NIST does not allow people to write their own applets that use the functions of the web site.

In the applet code, a method *measure* is called, which is the basis for getting the correct time and finding out how long it takes to get it. To get the time, another method, *GetDeltaT* is called, which uses a standard call to the server clock called *GetServerDateStrng*. A string containing the time and date of the NIST web server is sent back to *GetDeltaT*. Then, the time is read from the client's computer clock. Both time strings have a resolution of 1 ms, and they are converted to the number of elapsed milliseconds since the start of January 1, 1970, which is a time epoch used by the UNIX operating system. The difference between the server clock and the client's clock (in milliseconds) is calculated and saved as the variable *deltaT*. The time and date of the client's clock do not need to be close to the correct values for this to work correctly. For example, if the client's clock were behind by exactly one day, then *deltaT* would equal $+8.640 \times 10^7$ (the number of milliseconds in one day). This value is returned to the original applet method *measure* and stored as a variable *c.value*. The time shown in the web browser is the client's computer clock corrected by *c.value* (Figure 2).

To measure the delay of the call to *GetDeltaT*, the client's computer clock is used as a timer. Before the method *GetDeltaT* is called, a variable is set to the client's current time. When the variable *deltaT* is returned, a second variable is set to the current time of the client. The delay is calculated by comparing these two variables and stored in a variable *accessDelay*. The delay measurement includes the time it takes for the applet to call the time server, query the client's clock, convert the strings to time since the epoch, calculate the difference and send the variable back. The process times out if no values are returned within 3 s. If a value for *deltaT* comes back in less than 1 s, then the program continues. If not, it tries again. If it works three times but no values are returned in less than 1 s, then the shortest delay is used and the program continues. It will try a total of five times to get three readings. Upon failure, it will quit and display the message "Net Congestion". The goal is to show an error of less than 1 s and to show the time only if the error is less than 3 s.

Figure 2 – A Successful Time Request.

The displayed gray-line map is chosen from thousands of files that show where the sun is shining on Earth, based on the time of day and the day of year. This is calculated using Javascript in an HTML file named *clock-bot.html*.

If the applet is left running in a browser, it starts the process over every ten minutes, so that as the

client's clock drifts, the time shown is still correct. A typical computer clock gains or loses at least several seconds per day. The process is also restarted if the client's clock is adjusted, or the browser window is resized.

If the client is behind a firewall, the java applet may not work. The applet running on the local computer contacts the NIST web server on Transmission Control Protocol (TCP) port 8013, an arbitrary port chosen by NIST. Modern firewalls, by default, often close all ports that are not commonly used, so port 8013 must be reopened in order to implement the Java clock.

If the non-Java process is used, instead of running the applet and the Javascript, the HTML file *clock-nojava.html* is sent to the browser. In this case, the web server is called to return a time string containing the current time and date, which is adjusted for the selected time zone and is displayed in the browser. The amount of the offset includes the DST offset, if applicable, which was determined earlier. This is a static time display, and the user can click a "refresh time snapshot" link (Figure 3). This process uses TCP port 80, the standard port for Hyper Text Transfer Protocol (HTTP) requests.

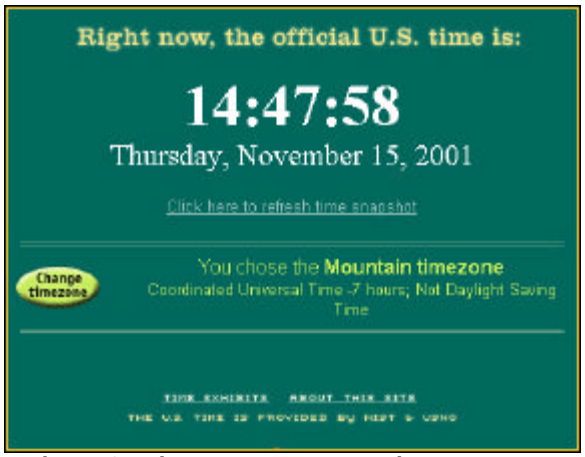

Figure 3 - The non-Java, "snapshot" Page.

HARDWARE

The Java clock sites run on NIST Boulder's external web server. It resides on a 350 MHz machine with four Gigabytes of main memory and four 64-bit processors. It runs the a version of the UNIX operating system. It uses a "highly available" two-node cluster for failure protection. This is referred to as a highly available system because two identical nodes run simultaneously, and if one fails, the other takes over. The dual node configuration is not used for load balancing, just for fail-over protection.

THE CLOCK

The web server clock is the reference for the time served to web clients. It uses an algorithm called *Autolock* to steer the time server clock to UTC(NIST).[2] The algorithm periodically synchronizes the web server clock with a NIST Internet time server that transmits UTC(NIST). The internal oscillator of the web server clock is characterized over time and steered to correct its frequency. This algorithm could be considered to be a slow frequency-locked loop. The benefit of steering instead of just synchronizing more often is that if the network link to the NIST server is temporarily lost, the clock frequency will stay close to the correct frequency for an extended period, keeping the web server on time. This algorithm holds the time uncertainty of the web server clock to less than 100 ms with respect to UTC(NIST) at all times, even if the link to NIST is unavailable for more than a day.

HISTORY

The original web clock produced by the NIST Time and Frequency Division was part of the division web site located at **http://www.boulder.nist.gov/timefreq**. This clock was added to the web site in 1997, and was easily the site's most popular feature. Due to its popularity the decision was made to create a new web site, independent of the division site, whose sole purpose would be to display the correct time. As a result, the time.gov domain name was registered on June 23, 1999 and the time.gov web site went on-line immediately afterwards.

Although the time.gov web site was designed and is maintained by NIST, the decision to launch the original site was made jointly with the United States Naval Observatory (USNO), which, like NIST, is an official source of time in the United States. The information displayed on the time.gov pages pertains to both NIST and the USNO as part of a mutual recognition agreement between the agencies. The nist.time.gov site was launched on January 5, 2001. Although its inner workings are essentially the same as time.gov, it displays some information that is NIST-specific. The time.gov web sites are already the most visited web sites maintained by NIST, and their usage is expected to increase at a rapid rate during the coming years.

USERS

By December 1999, the time.gov site was receiving about one million time requests per month. The average number of time requests on

time.gov in 2001 has been around two million per month, and the maximum was over three million per month. April and October are the peak months most likely due to the Daylight Saving Time transition days. An alternate page, nist.time.gov, was implemented early in 2001 to focus on and link solely back to NIST. All NIST web pages link to nist.time.gov, but time.gov is linked extensively by other sites on the Internet. The traffic on nist.time.gov grew to over one million time requests per month by October 2001. Many of the requests for these pages are undoubtedly the same users checking different time zones or coming back later (or daily) to check the time. Some users leave the applet running in their browsers for extended periods of time. Since the applet automatically starts its process over every 10 minutes, this type of user accounts for 6 time requests per hour.

The demographics of the users are unknown; however, over 60% of the requests originate from *.com* and *.net* domains. Around 5% are from *.edu* domains, and 1% are from *.gov* domains. Although the sites are extensively linked from around the World Wide Web, most users originate from search engines and other NIST sites. Figure 4 shows the traffic growth of both sites.

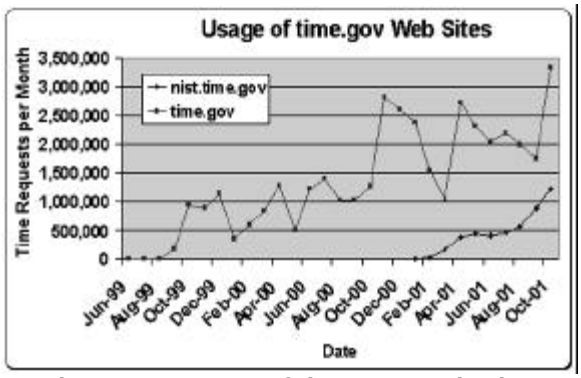

Figure 4 – Usage of time.gov Web Sites

OTHER TYPES OF WEB CLOCKS

The goal of time.gov and nist.time.gov is to efficiently provide the correct time with a running clock to as many users as possible. There are several other examples of clocks being displayed on web sites; however, most of these merely display the time from the client's own computer clock. There is at least one other example where a running clock is "streamed" from a web server to a client's browser. It is a graphical representation of a clock which is being constantly refreshed by the server. This method is highly inefficient, since it requires a connection to each client for the entire duration over which the clock is displayed. For this reason, it disconnects after twenty seconds.

Audio clocks are available where a voice announces the current time. These are bandwidth-intensive but, more importantly, due to buffering in the client's audio software, they announce the time 5 to 10 seconds late.

Other Java clocks use a servlet to communicate with the server. This approach, more direct than using an applet, may help circumvent problems for users behind firewalls. However, it does not appear to work with all browsers.

IMPROVEMENTS

The web clock can be improved in many ways, both functionally and aesthetically. Providing a button to change from the current 24-hour clock representation to a 12-hour clock with an indication of a.m. and p.m. will provide many users time in the format they are accustomed to using. Also, enabling clients to observe the time around the world would be very useful. Due to the number of countries and DST rules around the world, this would be a lengthy endeavor. Showing the time in all time zones simultaneously would also improve the usability of the site, as well as cut down on the number of the requests to the CGI script. In other words, on the index page, each time zone would show a running clock with the "local" time for that time zone. Also, the gray-line map calculation could be done once every hour by the server, and it would send the appropriate graphic when the page is requested. This would save the applet from doing orbital calculations every time a request is made.

Another way to streamline the time requests would be to take out some of the repeated calls to the web server. The first request is made just to get a date to see whether DST is in effect or not. Instead, the server could calculate this once per day and include it in the time string sent to the client. Also, the web server is queried several times in order to measure the shortest delay. Since the delay is shown on the web page, and we are not providing the information for making measurements or establishing traceability, these steps could be eliminated.

It would also be very useful if clients could automatically set their computer clocks to the time on the web site. This is possible, but there are security issues involved. A web site is not allowed to alter anything on a client's computer without special permission. Also, different operating

systems would have to be handled accordingly, complicating the process.

MEASUREMENT UNCERTAINTIES

Our goal for the time.gov web sites is to reliably display the time within 1 s of UTC(NIST), and this goal is realized for most time requests. If a Java time request is made, the page displays the estimated error of the time display. This serves as a very coarse estimation of the uncertainty of the displayed time, with a resolution of 100 ms (0.1 s). However, the instabilities in this delay are much larger than the resolution. Due to processing delays introduced by both the server and client (the dominant source of uncertainty), and/or network delays introduced by a busy Internet service provider, it is impossible to provide time to all clients within 1 s of UTC(NIST). In some cases the uncertainty might as large as 3 s or more, so the measurement uncertainty for a typical user is very difficult to characterize.

Since the uncertainty is so difficult to characterize, we recommend that the time.gov web sites in their current form be used a time-of-day displays only, and should not be used to establish traceability to UTC(NIST) or UTC(USNO) for measurements of time interval or frequency, not even for measurements (such as stop watch or timer calibrations) with the most modest requirements. We anticipate that future enhancements made to the sites will make them more suitable for metrological applications.

CONCLUSION

As the number of Internet users grows, more and more people will utilize NIST time services on the web. Both streamlining its operation and increasing its functionality are goals for the evolution of the time.gov and nist.time.gov web pages.

The mention of company or product names in this paper neither constitutes nor implies endorsement by the National Institute of Standards and Technology.

REFERENCES

[1] Levine, Judah. The NIST Internet Time Services. Proc. of 25th Precise Time and Time Interval Planning and Applications Meeting (PTTI), Marina Del Rey, CA. 1993 November. 505-511.

[2] Levine, Judah. Time Synchronization over the Internet using an Adaptive Frequency-Locked Loop. IEEE Trans. On UFFC(46): 888-896; 1999.