Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: April 25, 2019

A. Malhotra
Boston University
K. Teichel
PTB
M. Hoffmann
W. Toorop
NLnet Labs

October 22, 2018

On Implementing Time
draft-aanchal-time-implementation-guidance-01

Abstract

This document describes the properties of different types of clocks available on digital systems. It provides implementors of applications with guidance on choices they have to make when working with time to provide basic functionality and security guarantees.

Status of This Memo

Copyright Notice

1. Introduction

It is hard to understate the importance of time in modern digital systems. The functionality and security of applications (distributed or local to one system) and that of network protocols generally hinge on some notion of time. For implementation, these applications and protocols have to choose one of the types of clocks available on their system, each of which has its own specific properties. However, currently many of these applications seem to be oblivious to the implications of choosing one or the other clock for implementation. This behavior can be attributed to: a) the lack of clear understanding of the distinct properties of these clocks, b) trade-offs of using one or the other for an application, and c) availability and compatibility of these clocks on different systems. This document discusses a) and b).

More specifically, in this document we first define different methods used by protocols and applications to express time. We then define properties of clocks maintained by modern digital systems. Next we describe how systems obtain these values from these clocks and the security considerations of using these values to implement protocols and applications that use time. Finally we discuss trade-offs between security and precision of choosing a clock. The document aims to provide guidance to the implementors make an informed choice with an example of POSIX system.

2. Scope of the document

This document aims to provide software developers implementing protocols and applications that have to deal with time with the knowledge and understanding to make informed decisions regarding the available clocks and their respective trade-offs.

It does not describe functionality that is specific to the architecture of a PC, or other devices such as phones, IoT devices, switches, routers, base stations, or synchrophasors. Nor is the document applicable to a specific operating system. Throughout the document we assume that one or the other clock is available on most devices. How these clocks are available on different PCs or other devices is out of scope of this document.

We do not exactly recommend which clock should be used. We discuss the available options and trade-offs. The final decision would vary

depending on the availability of clocks and the security requirements of the specific application under implementation.

Note: Since there is a lack of standards on terminology related to time, we define some terms in the following section. Also, throughout the document, we define the terms as they become relevant. Different systems, depending on their OS, may use different terms for the same types of clocks. A survey on this is not in the scope of this document. We provide a discussion on how to access these values on POSIX and Windows systems. On other systems, implementors will have to determine themselves which of these values are available.

3. Expressing Time

Protocols and applications can express time in several forms, depending on whether they need to express a point in time or a time interval.

3.1. Absolute Time

Absolute time expresses a universally agreed upon reference to a specific point in time. Such a reference can be expressed in different ways. For instance, Unix Time refers to the number of seconds since midnight UTC, January 1st, 1970, while in everyday life, we reference such a point through year, month, day, and so on.

Because absolute time expresses a shared view of time, a system needs to synchronize its clock with a common reference clock, for instance one based on UTC.

Absolute time is often used to express the start or end of the validity of objects with a limited lifetime that are shared over the network.

3.2. Relative Time

Relative time measures the time interval that has elapsed from some well-defined reference point (e.g., 20 minutes from the time of your query).

Since relative time does not express a point in time, it does not rely on synchronized clocks between systems but only on a shared rate of passage of this time.

Relative time is commonly used in network protocols, for instance to determine when a packet should be considered dropped or to express Time To Live (TTL) values that govern the length of time for which an object is valid or usable.

4. Keeping Time: Different Clocks

Because time is relative to an observer, there cannot be a universally agreed upon time. At best we can achieve an approximation by constantly updating our own clocks against a common reference clock. Remaining close to this reference clock is a complex process that comes with its own set of difficulties.

In this section, we will have a look at the different clocks a system uses and how it maintains these clocks.

4.1. Native Clock

Each system has its own perception of time. It gains access to it via its native clock. Typcially, this clock counts cycles of an oscillator but some systems use process CPU times or thread CPU timers (via timers provided by the CPU). The quality of the native clock therefore dependends on either the stability of the oscillator or the CPU timer.

The timescale of the native clock is purely subjective -- no general meaning can be attached to any specific clock value. One can only obtain relative time by comparing two values. Because the value of the native clock always grows at a steady pace, never decreases, never make unexpected jumps, and never skips, the difference between two clock values provides the time interval between the two measurements.

The independence of the native clock from any external time sources renders it resistant to any manipulation but in return there is no guarantee that its clock rate is similar to that of any other system. This difference in rate, especially when compared to a reference clock, is called clock drift.

Clock drift depends on the quality of the clock itself but also on factors such as system load or ambient temperatur which makes it hard to predict.

4.2. World Clock

The native clock only provides means to measure relative time. In order to be able to also process absolute time, the system needs to be synchronized with a global reference clock. Since this clock strives to be the same on all systems, we call it the world clock.

There are a number of ways to maintain the world clock based on the system's native clock.

The first is to manually maintain an offset between values of the native clock and the reference world clock. Because of the clock drift of the native clock, this offset needs to be updated from time to time if a minimal divergence from the reference clock is to be maintained.

Secondly, a hardware clock provided by the system and set to be equivalent to the reference time can be used, allowing the system to retain the offset across reboots.

Finally, the reference clock can be obtained from an external time source. Typically, the Internet is used through a variety of timing protocols including the Network Time Protocol (NTP) [RFC5905], Chrony, SNTP, OpenNTP and others.

Each of these approaches has own problems attached to it.

Manual configurations can be subject to errors and misconfiguration. Also, for mobile devices, when moving between time zones, the offset must be corrected manually.

Accessing the hardware clock requires an I/O operation which is resource intensive, therefore many systems use the hardware clock only upon reboot, to initialize the clock offset; subsequent updates are made either manually or through timing protocols.

Further, on many systems the quality of the hardware clock isn't very high, leading to a large clock drift if solely relying on it. Worse, systems like microcontrollers that operate within embedded systems (e.g., Raspberry Pi, Arduino, etc.) often lack hardware clocks altogether. These systems rely on external time sources upon reboot and have no means to process absolute time until synchronization with these sources has completed.

Relying on Internet timing protocols opens up the system time to attack. Recent papers show vulnerabilities in NTP [SECNTP], [MCBG] and SNTP that allow attackers to maliciously alter system's world clock -- pushing it into the past or even into the future. Moreover, many of these time-shifting attacks can be performed by off-path attackers, who do not occupy a privileged position on the network between the victim system and its time sources on the Internet. Researchers have also demonstrated off-path denial of service attacks on timing protocols that prevent systems from synchronizing their clocks.

In other words, the process of obtaining the offset necessary to provide a world clock creates dependencies that can be exploited.

5. Implementation Approaches

Because absolute time relies on a shared interpretation of a value expressing time, the world clock is necessary when processing such values.

For relative time, however, where only the rate of passage of time needs to be close enough to that of the other systems involved, there is no need to rely on the world clock when determining whether an interval has passed.

Instead, by obtaining a value from the native clock when the interval has started only the native clock is necessary to determine when this interval ends. As the native clock does not rely on any external time sources, the implementation becomes resistant to the difficulties of coordinating with these sources.

However, using the native clock in this way comes with a caveat. Since the native clock is not subject to any adjustments by timing protocols, it is not adjusted for the error introduced by clock drift. While this is likely of little consequence for short intervals, it may become significant for intervals that span long periods of time.

Consequently, the choice of clock to be used is application-specific. If applications can tolerate a certain amount of clock drift or if the time intervals are short, implementers may prefer using the native clock. If the application relies on precise timing over long periods one has no choice but to fall back to the world clock.

6. Accessing the Native Clock on Selected Operating Systems

In most operating systems, the standard functions to access time use the world clock since that is normally what users would expect. This section provides an overview how the native clock can be accesses on some common operating systems.

6.1. POSIX

POSIX defines a system C API function which may provide native time: clock_gettime(), when used with a clock_id of CLOCK_MONOTONIC (when supported by the system). POSIX does not make a distinction between raw time and adjusted raw time in the definition of this function. Beware that, with some systems, CLOCK_MONOTONIC deliveres adjusted raw time and that CLOCK_MONOTONIC_RAW needs to be used as clock_id to get unadjusted raw time. Non-POSIX systems may provide different APIs.

6.2. Microsoft Windows

In the Microsoft Windows operating system, native time is called 'Windows Time' and can be accessed through the GetTickCount and GetTickCount64 API functions. The returned value is normally the number of milliseconds since system start. GetTickCount will return a 32 bit value while GetTickCount64 returns a value 64 bits wide that will wrap around less often.

7. Acknowledgements

We are thankful to Sharon Goldberg and Benno Overreinder for useful discussions.

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

Time is a fundamental component for the security guarantees claimed by various applications. A system that uses a time distribution protocol may be affected by the security aspects of the time protocol. The security considerations of time protocols in general are discussed in [RFC7384]. This document discusses the security considerations with respect to implementing time values in applications in various sections.

10. Informative References

[CLOCKDRIFT] Marouani, H. and M. Dagenais, "Internal clock drift estimation in computer clusters", 2008, <http://downloads.hindawi.com/journals/jcnc/2008/583162.pdf>.

[MCBG] Malhotra, A., Cohen, I., Brakke, E., and S. Goldberg, "Attacking the Network Time Protocol", 2015, <https://eprint.iacr.org/2015/1020>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, DOI 10.17487/RFC7384, October 2014, <https://www.rfc-editor.org/info/rfc7384>.

[SECNTP] Malhotra, A., Gundy, M., Varia, M., Kennedy, H., Gardner, J., and S. Goldberg, "The Security of NTP's Datagram Protocol", 2016, <http://eprint.iacr.org/2016/1006>.

Authors' Addresses

Aanchal Malhotra
Boston University
111 Cummington Mall
Boston 02215
USA

Email: aanchal4@bu.edu

Kristof Teichel
Physikalisch-Technische Bundesanstalt
Bundesallee 100
Braunschweig D-38116
Germany

Email: kristof.teichel@ptb.de

Martin Hoffmann
NLnet Labs
Science Park 400
Amsterdam 1098 XH
Netherlands

Email: martin@nlnetlabs.nl

Willem Toorop
NLnet Labs
Science Park 400
Amsterdam 1098 XH
Netherlands

Email: willem@nlnetlabs.nl