



Using Network Time Protocol (NTP): Introduction and Recommended Practices

**International SEMATECH Manufacturing Initiative
Technology Transfer #06024736A-ENG**

SEMATECH and the **SEMATECH logo** are registered service marks of SEMATECH, Inc.

International SEMATECH Manufacturing Initiative and the **International SEMATECH Manufacturing Initiative logo** are registered service marks of International SEMATECH Manufacturing Initiative, Inc., a wholly-owned subsidiary of SEMATECH, Inc.

Product names and company names used in this publication are for identification purposes only and may be trademarks or service marks of their respective companies.

**Using Network Time Protocol (NTP): Introduction and Recommended
Practices
Technology Transfer #06024736A-ENG
International SEMATECH Manufacturing Initiative
February 28, 2006**

Abstract: This report from the MFGM031M project provides an introduction to Network Time Protocol (NTP) as well as recommendations for architecting an NTP infrastructure to meet the timing demands of next-generation semiconductor manufacturing applications. It is intended to help factory network administrators and equipment engineers understand the key factors in designing an NTP network or equipment clock module.

Keywords: Computer Software, Factory Automation, Time Synchronization, Data Communications, Automated Data Collection

Authors: Ya-Shian Li (NIST), Gino Crispieri (ISMI) and Harvey Wohlwend (ISMI)

Approvals: Harvey Wohlwend, Project Manager
Brad Van Eck, Program Manager
Scott Kramer, Director
Laurie Modrey, Technology Transfer Team Leader

Table of Contents

1	EXECUTIVE SUMMARY	1
2	INTRODUCTION	1
3	NTP OVERVIEW	2
4	PROTOCOLS FOR DISTRIBUTING SYNCHRONIZED TIME	4
	4.1 Protocol Requirements	4
	4.2 Protocol Configuration	4
	4.3 Protocol Time Stamps	5
	4.4 NTP Version 4	5
	4.5 SNTP Version 4 [6][7]	5
5	NTP TIME SOURCES	6
6	NTP TIME SERVERS	7
7	NTP NETWORK TOPOLOGY DESIGN [11][12][13]	8
	7.1 Client/Server Mode	8
	7.2 Symmetric Active/Passive Mode	8
	7.3 Broadcast/Multicast Mode	9
	7.4 Manycast Mode	11
8	NTP ACCURACY	12
	8.1 Refresh Rate/Polling Interval	12
	8.2 Nanokernel [14][16]	12
	8.3 Windows NTP Accuracy	13
9	NTP SECURITY	13
	9.1 Access Control	13
	9.2 Authentication [18]	13
	9.3 Key Management	14
	9.4 Fault Tolerance	15
	9.5 Time Correction Value	15
10	NTP MONITORING	15
11	NTP RECOMMENDED PRACTICES	16
	11.1 NTP Topology Design	16
	11.2 NTP Basic Settings and Usage	17
	11.3 NTP Security	18
	11.4 NTP Monitoring	18
	11.5 Operating System Enhancements	19
	11.5.1 Hardware Enhancements	19
12	CONCLUSION	19
13	REFERENCES	20
	Appendix A – Definitions	22
	A.1 Acronyms	22
	A.2 Terminology	23

List of Figures

Figure 1	Factors Causing Degradation in Synchronization Accuracy in an Ethernet-based Distributed Network.....	2
Figure 2	Simplified Example of an NTP Topology with 4 Stratum Levels (in addition to a Reference Time Source Stratum level).....	3
Figure 3	Propagating NTP Time to Client Nodes Using Dedicated Time Servers or Routers	7
Figure 4	Client/Server Mode	8
Figure 5	Symmetric Active/Passive Mode	9
Figure 6	Broadcast/Multicast Mode	10
Figure 7	Manycast Mode	11
Figure 8	Autokey Protocol Exchange.....	14

List of Tables

Table 1	Reference Time Sources Receive Time from an Atomic, Traceable UTC Source.....	6
---------	---	---

Acknowledgments

The authors would like to acknowledge and thank Kang Lee (NIST) and John Messina (NIST) for their review of the paper.¹

¹ Official contribution of the National Institute of Standards and Technology (NIST) not subject to copyright in the United States.

1 EXECUTIVE SUMMARY

With future e-Manufacturing applications requiring precisely synchronized time for merging data from heterogeneous sources, it is necessary to understand the factors that degrade synchronization in the distributed systems of a semiconductor factory and ways to effectively leverage mainstream time synchronization protocols. The Network Time Protocol (NTP) is among the most widely used Internet protocols for synchronizing distributed clocks. For office-based applications where timing accuracy requirements are on the order of seconds, NTP configuration and management are trivial. In an industrial environment, synchronized time requirements tend to be more stringent and may provide critical value for e-Manufacturing applications, especially in Advanced Process Control (APC). Achieving synchronization accuracy requirements on the order of 1 millisecond (ms) requires careful design of the NTP network in terms of synchronization accuracy, security, network device extensibility, and synchronization management complexity. With proper design of the NTP topology, the protocol can support nominal accuracy on the order of a millisecond.

This document provides an introduction to NTP as well as recommendations in architecting an NTP infrastructure to meet the timing demands of next-generation semiconductor manufacturing applications. It is intended to help factory network administrators and equipment engineers to understand the key factors in designing an NTP network or equipment clock module to reliably and securely meet next-generation e-Manufacturing timing requirements.

2 INTRODUCTION

The de facto Internet time synchronization protocol, NTP, provides sufficient synchronization capabilities for many current factory automation applications [1]. In a common distributed network environment, synchronized network time has been a fundamental attribute in ensuring efficiency, reliability, security, and quality in a diverse array of applications.

Common applications for maintaining Information Technology (IT) infrastructure within an enterprise require synchronized time. Maintaining distributed databases, file backup systems, network security, and network measurement and control all require accurate Universal Time Coordinated (UTC) time stamps to ensure that the applications function efficiently and correctly. In real-time network applications such as multimedia conferencing, audio and video packets must be merged so the speech and image appear synchronized. While packets typically have sequence numbers in the headers, a single video frame may be streamed using several packets. Therefore, several packets may have the same time stamp, but different sequence numbers. Real-Time Protocol (RTP) typically relies on NTP to ensure multimedia packets are time-stamped accurately and to precisely measure network jitter for optimizing playback quality [2]. In authentication schemes, time stamps can be embedded into authentication tickets or certificates to prevent replay attacks by limiting the duration of the session key validity [3].

Inside the semiconductor manufacturing facility, several categories of applications can be enhanced with synchronized time and accurate time stamps [1]. The key applications requiring synchronized time are run-to-run control, statistical process control (SPC), fault detection classification (FDC), integrated metrology, e-Diagnostics, factory scheduling and dispatching, and automated material handling. Accurate clock synchronization of distributed systems is also fundamental for maintaining an enterprise's IT infrastructure. The most stringent industry clock synchronization needs currently lie around 1 ms for applications requiring quality time stamps. Most factories already rely on NTP for their basic IT functions and for synchronizing equipment

host time. Synchronizing factory devices within 1 ms accuracy by NTP is not a trivial task. The lack of determinism such as network delay, clock stability, network stack latencies, CPU load, and operating system latency can significantly degrade the synchronization performance and affect the applications relying on time stamps (Figure 1).

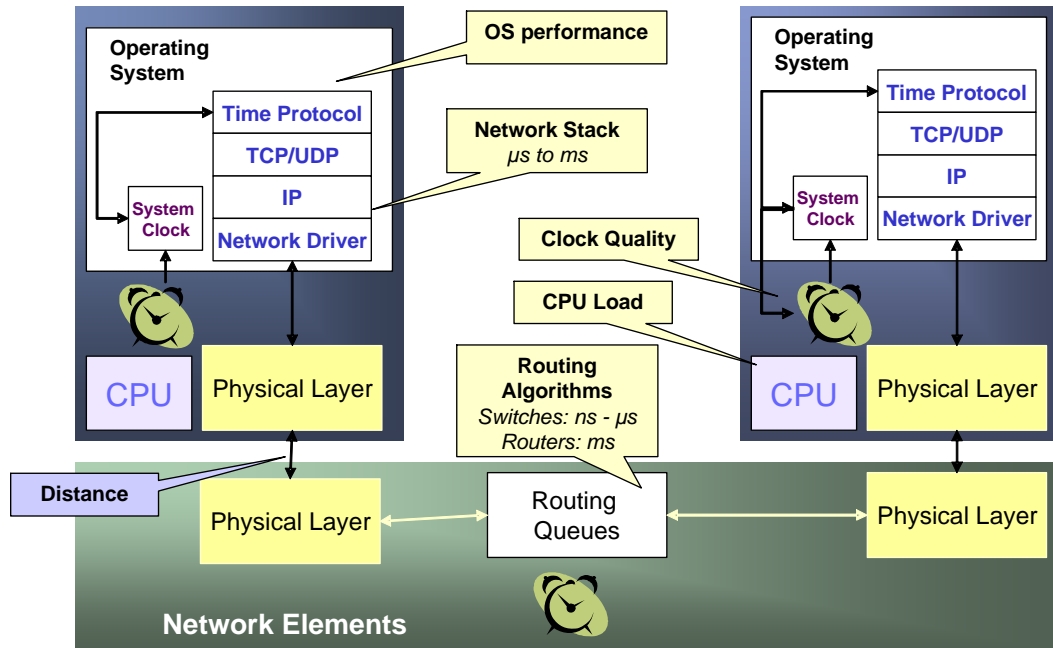


Figure 1 Factors Causing Degradation in Synchronization Accuracy in an Ethernet-based Distributed Network

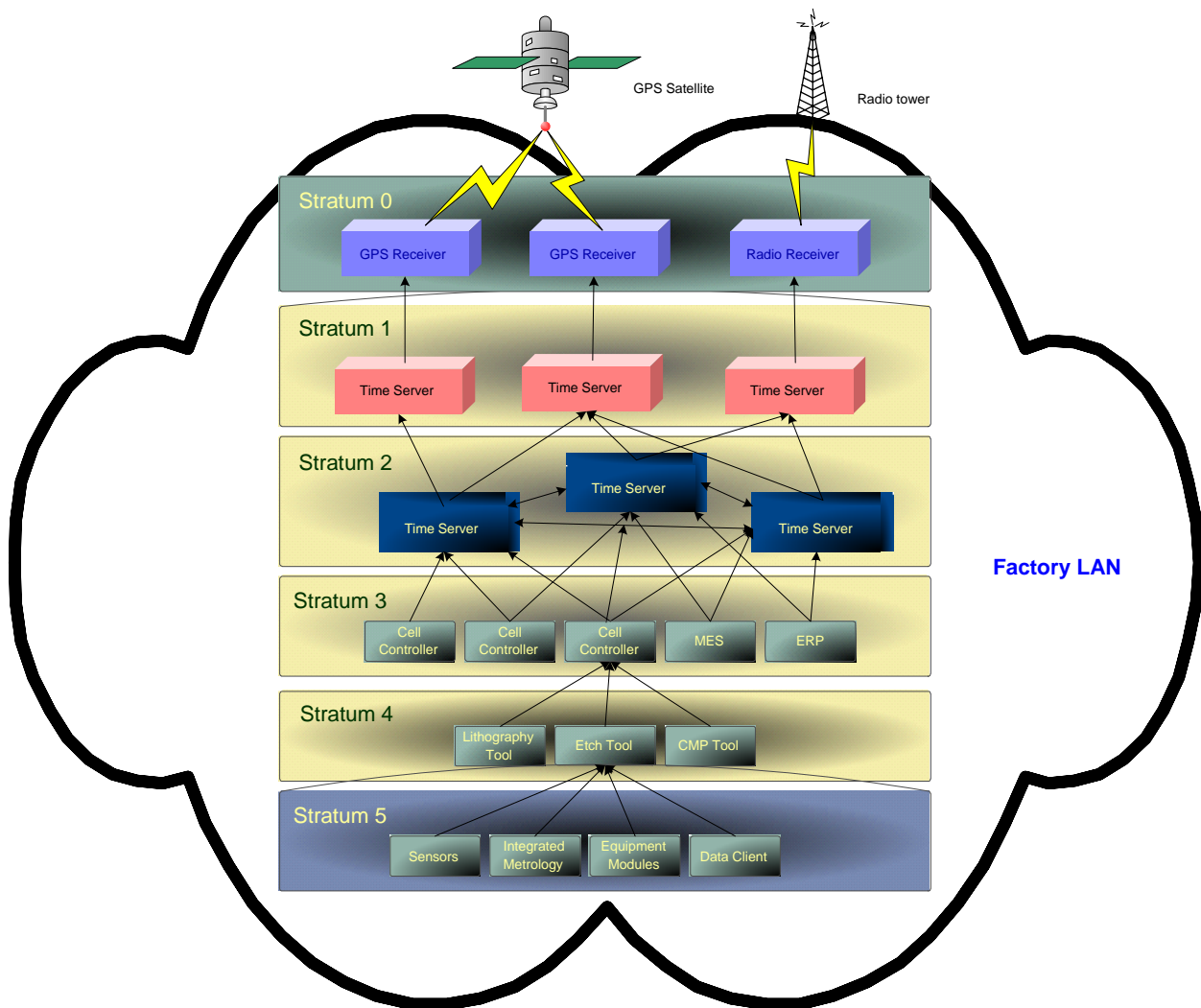
3 NTP OVERVIEW

Evolving over two decades, NTP provides the latest software synchronization capabilities by leveraging current computing power to meet the demands of today's distributed clock synchronization requirements.

NTP runs over User Data Protocol/Internet Protocol (UDP/IP) to ensure efficient service for propagating time throughout a distributed network of nodes. In contrast, TCP, a protocol guaranteeing delivery of packets, can end up with a number of retransmits that would potentially and unnecessarily overload the network. NTP mitigates the non-deterministic nature of networks by estimating three key variables: *network delay*, *dispersion* of time packet exchanges, and *offset*. NTP has built-in fault tolerance capabilities. The NTP algorithms are designed to select the best clocks within the set of reachable servers and to disregard any clock that varies significantly from the clocks of the other servers. An NTP client will synchronize only to an NTP server when the time on the clock does not vary substantially from other source clocks even if the server clock is on a lower stratum (Figure 2). In addition to adjusting the client clock to achieve improved clock accuracy, NTP algorithms also discipline the client clock by minimizing jitter and wander. NTP can typically achieve accuracy levels of 10 ms over WANs and about 1 ms over LANs. However, because of the lack of determinism in Internet-based networks, there is no worst-case accuracy guarantee.

An NTP network has the most accurate time when its time server(s) receives timing information from an authoritative source such as a radio or an atomic clock. An NTP client synchronizes with its server by a series of transactions over a polling interval (from 64 to 1024 seconds) that dynamically adjusts depending on the client clock's accuracy and network conditions between the NTP server and its client. Port 123 is used as both the source and destination port. The nodes to be synchronized must be identified and configured with NTP to form a synchronization subnet.

NTP *strata* indicate how many levels away a node is from an authoritative time source. Figure 2 illustrates an example of how NTP strata can be implemented within a factory. For instance, a stratum 1 time server has direct access to a radio or atomic clock. The strata range from 0–16, with stratum 0 being an atomic frequency source and stratum 1 as the time server synchronized to the atomic frequency source by a global positioning system (GPS) or radio receiver. In a large enterprise, the layered hierarchy ensures greater extensibility of growing IT networks. Stratum 2 servers can synchronize to a single or multiple stratum 1 servers and can also be organized to



Note: Strata can range from 0 to 16, which indicates the accuracy of the server with respect to its time source.

Figure 2 Simplified Example of an NTP Topology with 4 Stratum Levels (in addition to a Reference Time Source Stratum level)

synchronize among other stratum 2 servers to verify all the servers are propagating the same time to the rest of the network. A stratum 3 NTP host can typically serve multiple lower level devices or the clients of a particular system residing at stratum 4. For example, an equipment host (cell controller) could be responsible for synchronizing multiple pieces of equipment (e.g., lithography tool, etch tool, chemical mechanical polishing tool, etc.). The equipment serves as a stratum 4 host to its subsystems at the stratum 5 level, such as the sensors, data client, integrated metrology, and other modules within the equipment. The factory should design the hierarchy based on its synchronization requirements and network.

NTP provides a variety of features to ensure accuracy, reliability, and security of the time synchronization mechanism. NTP provides two security features to avoid accidental or malicious setting of inaccurate time. NTP allows for an access list restriction scheme as well as an encrypted authentication mechanism to ensure the packets originate from a trusted clock server.

4 PROTOCOLS FOR DISTRIBUTING SYNCHRONIZED TIME

The NTP suite is comprised of two protocol standards, NTP and SNTP, for disseminating accurate time throughout a distributed network. NTP provides the full time synchronization functions, but may become unwieldy for lower level devices due to the number of message exchanges and algorithm computation complexity. SNTP implementations typically forgo some of the features for ensuring reliability and accuracy to provide a simplified, lightweight version of the protocol for systems with accuracy requirements on the order of seconds or fractions of a second. The latest version of NTP for UNIX/Linux distributions is available as a free download.² Windows³ operating systems provide a default Windows Time Service (W32Time) application containing the suite of NTP algorithms.

4.1 Protocol Requirements

The NTP protocol requires the ability to support client/server and broadcast communication. While the protocol supports other association modes, the network of NTP servers supporting the clients should be configured in a hierarchical structure to provide a reliable time reference. Additionally, the network should support DNS to allow configuration files to use server names rather than IP addresses. Server names are preferred because they can be mapped to multiple addresses for greater reliability. Most implementations rely on UDP/IP and communicate NTP information through Port 123. Hosts supporting the latest version of NTP should also be able to perform 64-bit integer arithmetic.

4.2 Protocol Configuration

In typical UNIX and UNIX-based systems such as Linux, the NTP host can be configured through the */etc/NTP.conf* file. The Windows XP Professional and Server 2003 contain a default NTP implementation that requires manipulation of registry values under *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time* [4]. Third-party

² NTP Software Downloads. <http://NTP.isc.org/bin/view/Main/SoftwareDownloads>

³ Certain commercial products are identified to foster understanding. Such identification does not imply that they are recommended or endorsed by the National Institute of Standards and Technology or that they are necessarily the best available for the purpose.

suppliers of any platform may follow the use of the *NTP.conf* file or include a user-friendly interface to perform the NTP host configuration.

Before deploying an NTP system, the enterprise will need to select the devices responsible for distributing synchronized time throughout the company.

4.3 Protocol Time Stamps

NTP and SNTP use 64-bit fixed-point time stamps, where the first 32 bits represent seconds, followed by another 32 bits to represent the fractional part of a second. Because 32 bits are often more than sufficient for the fractional part, the remaining unused bits in the time stamp are 0-padded. The time stamps are determined by the number of seconds since January 1, 1900.

4.4 NTP Version 4

NTP version 4 [5] offers significant additional features while ensuring compatibility with legacy systems relying on previous versions of NTP. The latest version offers an improved clock synchronization algorithm for better accuracy, less network jitter, and dynamic adjustment of polling intervals to maintain a specific level of accuracy. To ease NTP administration, a new association mode, *manycast*, allows clients to automatically discover the servers within its proximity given a broadcast address. Security features are also enhanced to support public key cryptography. Additionally, NTP version 4 supports the nanokernel implementation to allow operating systems to provide nanosecond resolution and to improve on the algorithms of NTP at the kernel level for enhanced accuracy. NTP version 4 provides for greater extensibility of the number of devices a time server can support by allowing clients to prolong poll intervals to reduce network traffic without compromising accuracy performance; in version 3, clients running the NTP application continuously were required to maintain a minimum poll interval of 64 seconds. Dynamic polling intervals can offer greater NTP extensibility by reserving network, server, and client resources when sufficient synchronization accuracies have been reached. NTP headers have also been changed to support Internet Protocol version 6 (IPv6).

4.5 SNTP Version 4 [6][7]

NTP has become a rather large application because of its additional features. Since typical high stratum systems do not require the full facilities of NTP, SNTP has been designed to meet the needs of such clients without expending resources on features the client does not need for synchronizing time. SNTP provides accuracy typically within 100 ms [8].

Because SNTP and NTP have the same packet formats, the two protocols are readily interoperable; an NTP server responds to an SNTP and NTP client in an identical manner. Unlike NTP, SNTP typically communicates with a single time server. SNTP requires a single request and reply exchange, while NTP algorithms require more message exchanges to properly estimate the network delays and search for the most accurate clock with which to synchronize.

Furthermore, SNTP does not require data from previous exchanges with the server to be stored. The algorithms in SNTP are typically less complex, but accuracy and reliability are often compromised. SNTP has been useful for embedded devices where resources are limited and accuracy requirements are on the order of 100 ms or more. The latest version of SNTP includes support for IPv6 and OSI addressing and optional extensions including *anycast* mode and authentication schemes based on *multicast* and *anycast* modes.

5 NTP TIME SOURCES

In a factory environment, where critical systems must be carefully monitored and controlled, the administrator should avoid using public NTP servers. The accuracy and security of the time propagated by public time servers are beyond the control of the factory network administrators. It is recommended the factory use an atomic Universal Time Coordinated (UTC) time source coupled with dedicated time servers for the most accurate and stable NTP deployment.

Exploiting the maximum accuracy capabilities of NTP requires an atomic UTC time source attached (typically by the serial port at 9600 bps) to the stratum 1 time servers internal to the factory. When selecting a reference time source, the key factors are traceability to a standard UTC source (e.g., UTC, NIST) and the ability to preserve the time signal accuracy in terms of resolution, hardware, and software performance.

Common sources of UTC time include GPS, radio, modem, or Internet. GPS systems provide a measurement uncertainty of 1 microsecond (μs) to UTC (Table 1) [9]. Drivers for integrating GPS and radio reference time sources are freely available for UNIX/Linux systems [10], but require more configuration by the network administrator. For Windows systems, the default Windows Time Services does not support an additional hardware time source. However, the third-party supplier will typically provide the necessary software to accompany their reference clock.

Table 1 Reference Time Sources Receive Time from an Atomic, Traceable UTC Source

Method	Time Uncertainty to UTC	Hardware
GPS	$\leq 1 \mu\text{s}$	GPS Receivers
Radio	1–20 ms 0.1–15.0 ms	Radio Receivers WWV, WWH (U.S.) – HF Receiver WWB (U.S.) – LF Receiver DCF77 (Europe)
Modem	< 15 ms	Analog Modem Automated Computer Time Service (ACTS)
Internet	< 100 ms < 2 sec	NTP Time Servers Internet Time Service (ITS) nist.time.gov

To synchronize clocks and to calibrate and control oscillators with GPS receivers requires the ability to equip an outdoor antenna (roof or window mounted) for line-of-sight reception with GPS satellites. The GPS constellation of satellites includes at least 24 satellites at any point in time. Each orbiting satellite carries a cesium or rubidium-based atomic clock. A GPS receiver can typically communicate with 8 to 12 of the satellites and averages the signals to determine its current time. The GPS receiver can often provide time-of-day and date information in a computer readable format by RS-232 interface to a stratum 1 NTP server. NTP servers with integrated GPS receivers can also be purchased from various suppliers. Such products often contain a stable oscillator to maintain synchronization when GPS signals are not available because the line of sight is obstructed.

6 NTP TIME SERVERS

In a typical NTP deployment, two types of devices generally act as the host for reliably distributing synchronized time. Factories can choose to have either type of device or a combination of dedicated time servers and network routers to disseminate accurate timing information to their NTP clients (Figure 3). Such devices contain an NTP server implementation allowing it to exchange information with the client to provide a reliable time synchronization source to a sufficient level of accuracy.

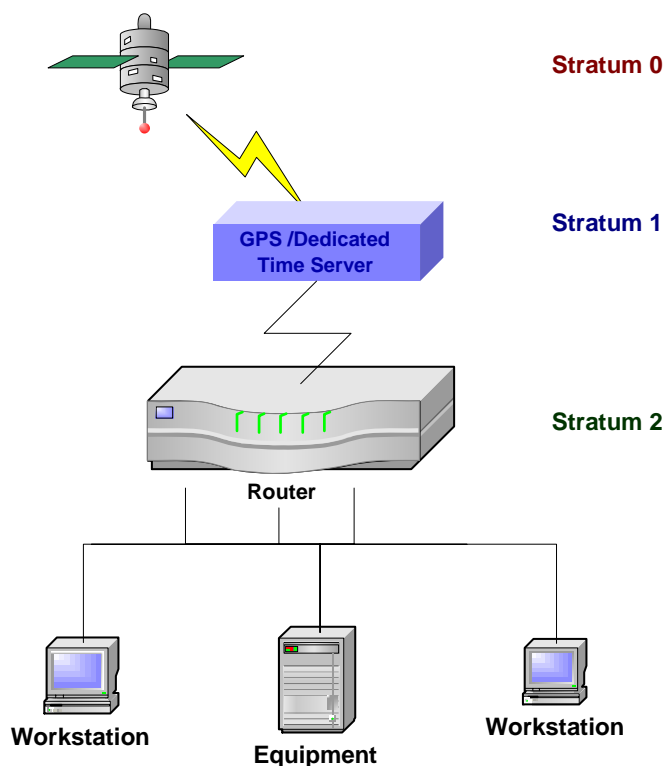


Figure 3 Propagating NTP Time to Client Nodes Using Dedicated Time Servers or Routers

Dedicated time servers are systems that only disseminate accurate timing information. The time servers are typically responsible for acquiring timing information from a reliable source such as a GPS or radio receiver or another time server at a lower stratum level. The server can ensure it has an accurate sense of time by exchanging time information with its peers. The time server is also responsible for propagating the acquired information to a set of clients at a sufficient rate to ensure the desired level of accuracy. Because dedicated time servers often have an RS232 and an RJ-45 Ethernet interface, they can be readily integrated into an Ethernet-based network.

Factories can configure a router's (or switch's) time to be an authoritative time server for each device on its subnet. System clocks in routers maintain time internally based on UTC, but can be configured to display the time in the local time zone. Routers can distribute time to nodes on its network path. Some routers also can attach to a radio or atomic clock.

From each of the time servers, association modes must be established with all nodes requiring synchronized time. To provide reliable and sufficient accuracy down to 1 ms each of these devices and their clients requiring 1 ms accuracy must be carefully configured.

7 NTP NETWORK TOPOLOGY DESIGN [11][12][13]

Based on the dedicated devices available in the enterprise for propagating accurate time, the NTP topology must be designed to attain a reliable and efficient configuration for achieving sufficiently accurate synchronization with all nodes requiring synchronized time. Before designing the topology, it is useful to understand the various association modes offered by the specification.

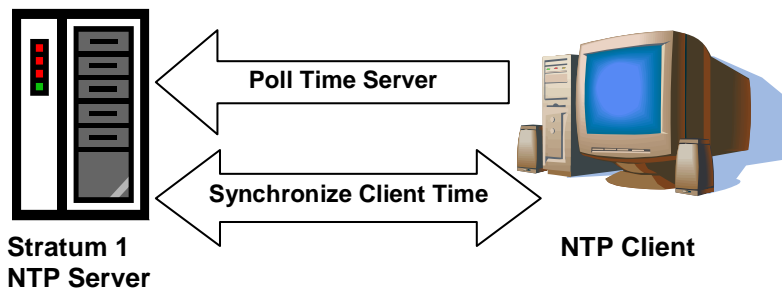
NTP is designed to work under several association modes to promote flexibility and accommodate different system requirements. The available association modes are client/server, active/passive symmetric, broadcast/multicast, and manycast.

7.1 Client/Server Mode

In the client/server mode, the client depends on a server within the group, and no member in the same group and stratum can synchronize to the client. The client makes a request to the server by periodically polling the server for time information and expects a reply in the future (Figure 4). This setup is generally more robust against malfunction and protocol security attacks.

Client/server mode can be specified in the configuration file by including a *server* declaration in the client node indicating the client would like to receive time from the specified server but does not want to provide time to the remote server. A client can specify multiple servers to ensure robustness, when one server is unreachable or does not provide reliable time. No specific configuration is needed on the server-side.

For the factory to build a reliable NTP deployment foundation, it is fundamental that the factory support the client/server mode. The client/server mode provides a robust hierarchical subnet of NTP servers for synchronizing the system's NTP clients.



Note: Client polls a server specified in the client's configuration file to initiate the synchronization exchange. The server receives the current time of the client and sends its current time to the client.

Figure 4 Client/Server Mode

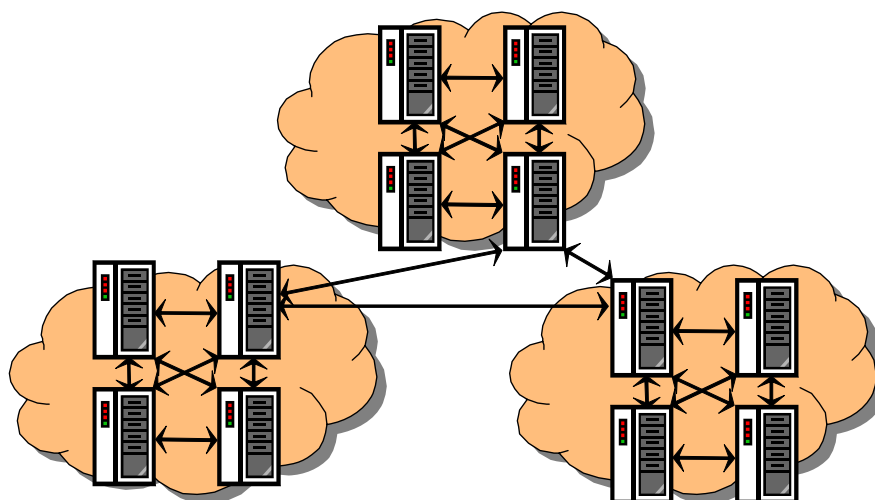
7.2 Symmetric Active/Passive Mode

Symmetric mode allows for a group of peers, typically in the lower strata, to operate as mutually redundant NTP servers. In other words, they serve as backups of each other. Each peer can be equipped with a primary reference source such as a GPS or be synchronized to another group of mutually redundant servers at the same stratum. In symmetric mode, a host will poll other hosts on the same stratum, but will also listen for and respond to other hosts on the same stratum. The goal of this mode is to provide a system of checks and balances among a group of servers sharing the same stratum to ensure it is not propagating poor timing information to higher stratum

servers or clients. If one peer loses its time source or ceases to exist on the network, the remaining peers would automatically reconfigure.

To specify symmetric mode, the *peer* declaration is used to specify another host with which it will perform the symmetric mode synchronizations. For symmetric-active mode, both hosts must have the *peer* declaration. Otherwise, NTP automatically defaults to symmetric-passive mode if only one host has the *peer* declaration. In symmetric-active mode, the host with the *peer* declaration signifies its willingness to synchronize and be synchronized, while a host in symmetric-passive mode without the *peer* declaration will signify its willingness to be synchronized. Due to the susceptibility of an intruder to impersonate a peer and introduce deviant times into the system, symmetric mode should always be validated using the NTP security features.

In large NTP deployments the recommended topology is to include clusters of lower stratum level servers (Figure 5). These clusters can also be physically dispersed throughout an enterprise to minimize latencies to the clients they serve.



Note: NTP redundant server clusters based on peer relationships are useful for enterprises with a large number of devices that need to be synchronized. Within each cluster, a server is dedicated to peer with chosen servers in other clusters. This ensures the servers are distributing a common, synchronized time throughout the enterprise.

Figure 5 Symmetric Active/Passive Mode

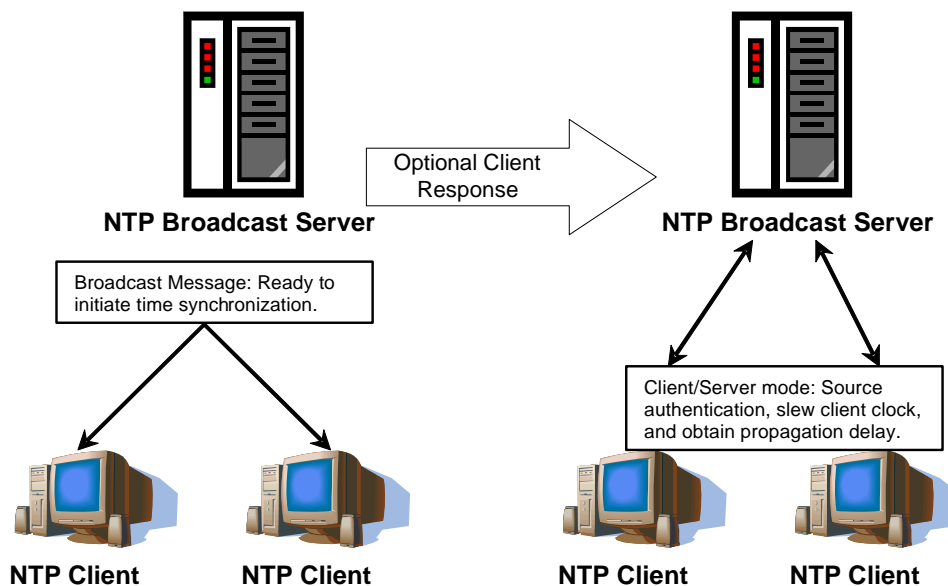
7.3 Broadcast/Multicast Mode

When accuracy and reliability of time information is not as stringent for a set of clients, this set can be configured to rely on broadcast or multicast NTP messages. The trade-off lies in management complexity versus optimal NTP synchronization performance. The NTP topology is typically a set of NTP servers with a large group of clients all on the local subnet. Broadcast and multicast modes allow the administrator to use the same configuration file for many clients, since clients would be configured with the same broadcast server. Clients listening to broadcast and multicast messages are not required to respond to the server. However, configuration on the client side allows the client to set up an exchange with the server to improve estimates of the message propagation delay.

In broadcast mode, the server broadcasts a message to all of its clients to initiate synchronization (Figure 6). Following the receipt of the broadcast message, a client will wait at random intervals before attempting to respond to the server to avoid a bombardment of messages if all the clients synchronized at once. The client and server continue to exchange a series of messages to authenticate the source, set the clock, and obtain an estimate of the roundtrip delay until the client time is set, at which time the exchange terminates; however if the server does not respond, the client will use a default propagation delay to correct its clock.

Unless a router is specifically configured to forward NTP broadcast messages, broadcast mode will be effective only within a subnet. To specify a broadcast server, the server should have the *broadcast* declaration in the configuration file along with the local subnet address. A client receiving the broadcast message should indicate *broadcastclient* in its configuration file allowing the NTP client to respond to broadcast messages.

In multicast mode, only the clients configured to listen for an NTP server multicast message will receive messages from a multicast server. In IPv6 or in a router configured to forward NTP broadcast messages, multicasting can be established on any node where its operating system supports it and the routers support Internet Group Management Protocol (IGMP). To specify multicast mode, a server is required to have a *broadcast* command, but using a multicast address. A multicast client will have the *multicastclient* keyword along with the multicast address in its configuration file.



Note: In Broadcast/Multicast Mode the server initiates synchronization with the client. It is optional for the client to respond and continue a series of unicast messages with the server to estimate the propagation delay of the timing messages. Clients wait for a random period of time before responding to ensure the server will not be overloaded.

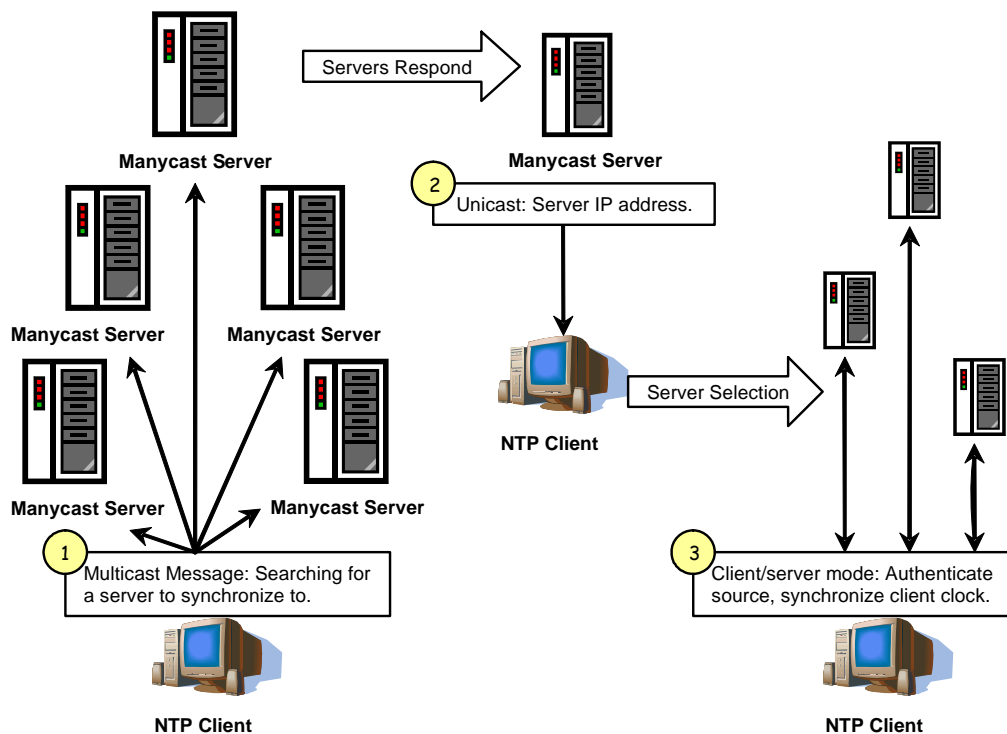
Figure 6 Broadcast/Multicast Mode

7.4 Multicast Mode

As a new feature in NTP version 4, the multicast association mode is designed to enhance the reliability and ease of administration of the protocol. In multicast mode, NTP clients survey prospective servers. The clients can evaluate the time values of several servers using an expanding ring search of the IP multicast tree before deciding to synchronize with a set of the three most accurate and stable servers within its range as determined by the client-side NTP software.

As depicted in Figure 7, the client initiates a multicast exchange by multicasting messages at intervals with a specified time-to-live (TTL). A multicast server at an equal or lower stratum level receiving the message replies to the client with its address. Once the client receives the IP address, the client can establish a unicast exchange and operate in client/server mode. The goal is to enable the use of the best available server and to automatically reconfigure if a current server fails. The multicast mode is especially useful in ad hoc networks, where mobile, wireless systems can synchronize to time servers within its range. However, wired systems can also benefit from the flexibility and robustness of the multicast paradigm.

A client can deploy the multicast association mode by using the *multicastclient* keyword in its configuration file. The IP address used with the keyword must match an address associated with the *multicastserver* keyword on a designated multicast server. Multicast servers will listen on the broadcast address for multicast client messages. NTP servers providing multicasting capabilities can serve multiple IP subnets.



Note: NTP client initiates a multicast association mode to determine the set of servers it should synchronize to. The reachable multicast servers at an equal or lower stratum level will respond with their specific IP address. The client selects the most stable and accurate servers available for synchronization in client/server mode.

Figure 7 Multicast Mode

SNTP version 4 refers to this mode as *anycast*. An anycast client preferably communicates with only one server, as opposed to a few servers when operating in multicast mode. The client initiates the communication by sending to an anycast IP address, and the routers are responsible for transmitting the datagram to preferably a single server offering NTP anycast service. Because NTP suppliers often consider the distinction between anycast and multicast synonymous, the latest draft SNTP revision has switched to the use of multicast for consistency [5]. In multicast mode, the SNTP client differs from an NTP multicast client in that it establishes a unicast communication with the first reply it receives and ignores the later replies from other servers.

The Windows Time Service in Server 2003 and XP currently does not support multicast or broadcast modes with peers [4]. For Windows clients requiring stringent accuracy, the clients should be explicitly configured to operate with local time servers.

Once the overall architecture of the distributed NTP system is decided and each host has an allocated role, each NTP node can be configured to achieve an optimal balance of accuracy, security, and monitoring capabilities while maintaining system performance.

8 NTP ACCURACY

NTP accuracy is affected by the ability to accurately measure the other clocks on the network. Network determinism, NTP polling interval, clock stability, operating system latencies, and kernel clock resolution and adjustments are all factors affecting synchronization accuracy. Within the NTP application, the administrator has control over the polling interval and, for some operating systems, the kernel clock.

8.1 Refresh Rate/Polling Interval

NTP version 4 has significant modifications to improve polling frequency. Current algorithms are designed to maintain accurate time using dynamic polling intervals ranging from 16 seconds to more than 1 day. NTP can determine poll intervals dynamically based on clock offset measurements. The default configuration relies on dynamic polling. For applications or systems that are unable to offer this flexibility, polling frequencies can be specified by *minpoll* and *maxpoll* parameters in the configuration file. The intervals are specified in seconds as a power of 2. For example, a *maxpoll* of 10 is equivalent to 1024 seconds. Similarly, polling intervals can also be configured in the same manner for SNTP clients.

8.2 Nanokernel [14][16]

To enhance the accuracy capabilities of system clocks, kernel software has been developed to mitigate jitter and to discipline the clock directly from a precision time source. One key feature is to provide improved resolution down to a nanosecond, since typical operating systems can support only millisecond resolution or less. The improved resolution allows for clock offsets to be determined at a finer granularity and improves accuracy levels down to microseconds. The accuracy attained using a nanokernel with a precision time source such as a GPS receiver is around 1 μ s. The nanokernel is available for the FreeBSD 4.x and Linux kernels. Solaris and Tru64 also provide microsecond and nanosecond resolutions with additional kernel support.

8.3 Windows NTP Accuracy

Current Windows operating systems do not support nanokernel in its platforms. Windows XP and Windows Server 2003 contain NTP by default through W32Time. The W32Time application in previous versions of Windows-based servers, such as Windows 2000, relies on SNTP. The maximum and minimum polling intervals can be adjusted through the W32Time variables in the Windows registry. Additionally, other NTP implementations for Windows provided by third-party suppliers typically have a graphical interface for configuring the polling interval.

Achieving 1 ms or better accuracy on a Windows system for applications requiring accurate relative time stamps, such as software benchmarking, currently require the use of a CPU counter or additional hardware. The Intel IA32 CPU contains the *Read Time Stamp Counter* (RDTSC) instruction, which allows the user to read the value of a 64-bit time stamp counter on the CPU that is incremented at every clock cycle. For example, a 3 GHz computer will be able to achieve an accuracy of ± 0.333 ns. For systems with the Intel SpeedStep technology, the computer frequency can vary; however, software is available to mitigate the CPU frequency variations [17]. Another method is to acquire a separate timing card (e.g., a PCI card) with a quality oscillator attached to a Windows system to achieve accurate relative timing. Used in conjunction with NTP and a few additional calculations, both methods can help achieve more accurate timing.

Once the desired accuracy is attained, the synchronization accuracy level must be maintained within an application's tolerance window.

9 NTP SECURITY

It is often good practice to adhere to additional security precautions to avoid accidental or malicious interference with system clock synchronization. Unauthorized access by computers external or internal to the network can introduce inaccurate times and thus degrade the quality of the synchronization scheme. NTP has provisions for both access control and authentication. Authentication is ensured by an initial exchange of keys, which systems are not required to exchange in advance. NTP version 4 contains the *Autokey* feature for authenticating NTP packets. For SNTP clients and servers, access control and authentication are supported in an identical manner to ensure interoperability between the two protocols.

9.1 Access Control

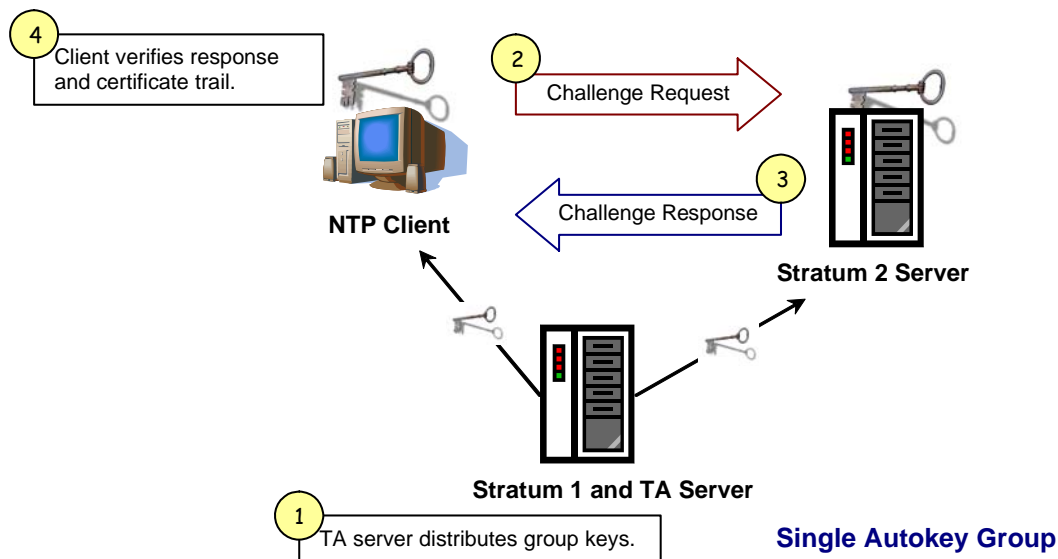
NTP has basic capabilities for allowing only specific nodes or subnets to access certain servers using address filtering. The IP addresses can be specified in the *NTP.conf* file using *restrict* or *discard* options. Because IP addresses can be easily spoofed/faked, address filtering should work in conjunction with a robust authentication scheme.

9.2 Authentication [18]

NTP also includes provisions for ensuring authenticated clock synchronization using the *Autokey* protocol to prevent masquerade and middleman attacks. NTP version 4 supports symmetric key and public key algorithms to verify message and sequence number integrity to prevent replay attacks. NTP version 4 also supports digital signatures, X.509 certificates, and several identity schemes based on cryptographic challenge-response algorithms to authenticate the source of the packets. The Autokey protocol is based on the OpenSSL library's PKI algorithms. The public

key management can either be provided by commercial services or produced by utility programs available in the OpenSSL library. Three variants of the Autokey protocol allow the security features to function with any of the association modes.

The Autokey protocol is enabled on an NTP node by default; however, it is ineffective unless it is properly configured. The *autokey* command specifies the interval between key refreshes. The default value is 12 or about every 1.1 hours. The node will check for the encrypted authentication fields. The *auth* flag ensures new associations, and remote configurations must be authenticated. If the flag is disabled, new association modes, such as a broadcast server, can potentially disrupt the client's timing (Figure 8).



Note: How each exchange takes place is determined by the authentication mechanism and the challenge-response identity scheme used. The Trusted Authority (TA) server is typically, but not restricted to, a stratum 1 server. Also, the TA server does not have to be an NTP host.

Figure 8 Autokey Protocol Exchange

9.3 Key Management

Hosts adhering to the NTP security model possess a group key generated by a Trusted Authority (TA). A dynamic trusted certificate trail should be built to verify its validity with a TA. Trusted groups encompass a group of hosts with trusted and non-trusted certificates. Trusted hosts must operate at the lowest stratum in the group and hold a trusted, self-signed certificate. Hosts with non-trusted certificates must have a certificate trail ending at a trusted host. NTP has defined this chain of trust as the *provenance* of the client. When a client authenticates a group server, the server provides the group key. Once the client receives the key from the server, it verifies the server has the same secret group key as itself and the server's linked certificate trail to a trusted host.

NTP keys are generated by an NTP utility program called *NTP-keygen* available in the NTP software distribution. The utility program generates several files containing a suite of authentication mechanisms which include Message Digest (MD5) symmetric keys, RSA and DSA public keys, identity group keys, and self-signed X.509 certificates. As vulnerabilities have been discovered in MD5, new NTP deployments should rely on public key mechanism for

enhanced security. The utility also manages the key data to ensure they are frequently refreshed. Once the key is refreshed, the protocol ascertains that NTP synchronization continues uninterrupted by the updates.

The Windows Time Service relies on shared key authentication based on standard domain security features to ensure the validity of NTP packets; therefore, the application itself does not currently support the *Autokey* feature. Windows-based NTP software by other suppliers typically has a form of NTP security implemented, but deployment of the security features depends on the supplier.

9.4 Fault Tolerance

In addition to the two explicit security features, NTP algorithms rely on redundancy to ensure outliers (e.g., potentially compromised systems) cannot easily disrupt the system. Because NTP algorithms filters unreliable time, launching an attack on time synchronization in an NTP system require the ability to spoof a large number of internal, lower stratum servers to provide a statistically significant change to the NTP time in a *robustly* deployed NTP environment. In a weak deployment, where a single server synchronizes all its clients, the attack would have to be launched only on the server node. Simply providing redundant servers and peers is one mode of defense against security attacks.

9.5 Time Correction Value

Another mechanism to defend against harmful time alterations is to set a maximum time correction value. The network administrator will need to decide on the actual value because of the disadvantages of setting the value too low. When this occurs, hosts that have been offline for a lengthy time can potentially fail to synchronize. Setting the value too high leaves more room for an attacker to create problems on the network.

NTP has many built-in facilities to ensure time is distributed throughout the network in a reliable manner. In addition to using the security features, monitoring the performance of NTP is helpful if deviations in time distribution were to occur.

10 NTP MONITORING

For critical applications or nodes where synchronized time provides significant cost benefits, monitoring the synchronization performance is essential. NTP provides features to enable continuous, long-term monitoring of server and client clock performance by generating log files of various clock and synchronization statistics. In *ntpd*, the command *statistics* enables the tracking of six types of clock statistic information. These include *clockstats*, *cryptostats*, *loopstats*, *peerstats*, *rawstats*, and *sysstats* [19]. Rigorous monitoring is typically done on stratum 1 and 2 servers; as the stratum becomes higher, less monitoring is necessary. The *clockstats* option allows the reference time source clock driver statistics to be recorded. A clock driver can record statistics of the hardware clock in addition to the default information, which includes the date, time, hardware clock address, and the last timecode received from the clock. The *cryptostats* command allows monitoring of the public key protocol; however, this option currently requires the OpenSSL cryptographic software library. The *loopstats* file contains statistics about the NTP filtering algorithms and the host's clock stability. *Peerstats* file provides clock statistics data about each of the peers the current server associates with. The *rawstats* file contains a log of all the messages received from peers and clock drivers from which the host is

configured to receive timing information. *Sysstats* logs the events of the *ntpd* daemon and maintains figures for the number of erroneous NTP packets or packets that failed authentication.

In the NTP daemon distribution available to UNIX and Linux platforms, the *ntpq* standard NTP query program, and *ntpd* special NTP query program are additional utilities that verify correct operation of NTP on a local or remote host. A *trap receiver* program that listens for NTP mode 6 packets can be used to detect exceptions by hosts in the system. However, for security purposes, remote access to the NTP utilities may be disabled by the administrator.

While SNTP does not prevent statistics collection, it is typically not implemented or directly deployed on the SNTP host due to limited resources. However, a system dedicated to monitoring the distributed synchronization can remotely monitor and control an SNTP client using a query program such as *ntpq* or *ntpd*.

Under W32Time, the component that monitors the current system state and logs events is the Windows Time Service Manager. In the default configuration of Windows XP, the events can be viewed under *Start* → *Control Panel* → *Administrative Tools* → *Event Viewer* → *System*. The events log registers current information as well as any synchronization warnings or errors.

Once NTP is configured properly, maintenance of the protocol should be trivial.

11 NTP RECOMMENDED PRACTICES

Each enterprise network has unique characteristics; therefore, deploying factory-wide synchronized time should be designed with some key factors in mind. Once NTP has been configured on the respective hosts, NTP allows the synchronization process to run with minimal administration.

Achieving 1 ms accuracy within a LAN synchronized to a local time source has become routine, given the currently available networking and processing capabilities.

11.1 NTP Topology Design

- ***Design the topology before deploying NTP***

The devices that need to be synchronized in the factory need to be examined and categorized based on subnet and/or function. Once the categories are established, timing requirements are determined. Selecting reference clock sources and association modes should be based on the most stringent synchronization accuracy requirements.

- ***A stable source of time is essential***

Having a stratum 1 server (i.e., a server directly linked to a radio clock or GPS) behind factory firewalls provides a reliable source for synchronizing clocks within a LAN. However, if the time source must be from the Internet, the company firewall would have to allow UDP traffic on port 123 of the factory server and the external server that is doing the synchronizing. While measurements taken over short intervals such as a second can rely on the local system clock, measurements taken over extended periods would be timed more accurately if a more stable frequency source such as UTC were used.

- ***Maintain server redundancy***

Servers synchronizing a sizable or critical system should operate in groups of three or more redundant servers in symmetric mode, each with its own set of three or more stratum 1 or 2 servers in client/server mode, to avoid common points of failure [15].

- ***Keep the stratum levels low***

To simplify NTP protocol management and reduce network bandwidth, it is preferable to maintain a small number of stratum levels. Additionally, accuracy degrades as a host is placed farther away from a stratum 1 server. To determine the number of levels needed, the enterprise would examine the routing hierarchy of the network and provide a structure that would offer resiliency when time servers fail, while balancing the trade-offs of network load and synchronization accuracy.

- ***Determine the synchronization needs for each networked device***

By determining the nodes that would require accurate timing, the NTP topology can be simplified. When the deployment model is simplified, it allows the hierarchy to collapse into fewer stratum levels.

- ***Manually configure nodes with more stringent timing requirements***

For each node, all the IP addresses of all devices with which the node should form an association should be manually configured. In LANs, broadcasting NTP messages reduces configuration complexity; however, it also marginally compromises timekeeping accuracy.

- ***SNTP clients should be used only at the highest strata within a subnet***

Consequently, clients relying on SNTP should not act as servers to other devices.

11.2 NTP Basic Settings and Usage

- ***Use the latest versions of NTP and SNTP available***

In the interest of accuracy and reliability, NTP algorithms continue to be honed to deliver improved features.

- ***Increase synchronization frequency as necessary***

The synchronization protocol should be allowed to run continuously and provide updated time information at rates sufficient to compensate for significant drift of the available oscillators. Synchronizing once when a system is booted up is typically not sufficient after a period of time. Ideally, the system should allow NTP to dynamically determine the frequency of synchronization due to fluctuations in clock behavior. For maintaining a 10 ms clock accuracy, typical computer clocks tend to require synchronization about once every 100 seconds. The synchronization frequency can be relaxed if requirements are not stringent and the clocks have proven to be relatively stable.

- ***Use the ntpd with the -q option if it is not practical to run ntpd continuously*** [15]

In systems where *ntpd* cannot be run continuously due to resource or other constraints, *ntpdate* is often configured in *cron* tasks to be run at certain times in the day. However, *ntpdate* does not have the signal processing and error checking algorithms to provide the same level of synchronization accuracy as *ntpd*. It is therefore recommended to use *ntpd* with the *-q* option as a *cron* task. With the *-q* option, the *ntpd* daemon exits immediately after the clock is set. To synchronize more quickly, typically within 10 seconds, the *iburst* option should be used with the *server* configuration. Additionally the *ntpdate* application may be retired in the future. Clients with kernel support (e.g., Solaris, FreeBSD, Tru64, and Linux) for disciplining clock frequency also have an additional feature. The clients will initially run *ntpd* to set the system time, which can take up to hours to reach a satisfactory level of stability as determined by the daemon. The *ntpd* is then stopped and run in a one-time mode as required to maintain sufficient clock discipline based on the kernel commands.

- ***System clocks should have sufficient resolution***

Sufficient resolution of the clock is needed for both synchronization accuracy and accurate measurements of packet arrival jitter.

- ***Time stamp data and events in UTC***

As NTP and SNTP communicate in UTC, applications requiring synchronized time stamps of distributed events would benefit from a single point of reference, thereby eliminating conflicts from daylight savings time and various time zones. If additional storage is available, for example in a database, an additional time stamp representing the local time may also help avoid future conversion calculations.

11.3 NTP Security

- ***Secure system assets***

Time on a machine can serve as a critical resource; therefore, using the security features available in NTP is recommended to avoid accidental or malicious setting of incorrect time, *especially when operating in symmetric or broadcast modes*. An intruder can potentially impersonate a symmetric active peer and provide erroneous time information to the system. Using a PKI infrastructure is recommended for the most secure implementation. PKI also provides a method to automate the distribution of keys in a secure manner instead of manually configuring keys.

11.4 NTP Monitoring

- ***Monitor synchronization performance on critical nodes*** [8]

Some nodes may have critical timing demands or offer more return on investment when it comes to their ability to time stamp accurately. Such nodes should be selected for collecting data on NTP performance. The data can be retrieved through a separate port (e.g., SNMP queries). To parse the data to produce the performance reports necessary for a factory's specific network data analysis, custom software or scripts may need to be developed.

- ***Analyze interrupt latency***

For applications requiring accurate time triggers such as measurement of external events, having a synchronized clock is only part of the solution. Analysis of the interrupt latency can determine whether the code triggering a measurement will actually be scheduled in less than the specified amount of time.

- ***Minimize LAN jitter***

LAN traffic should be analyzed to ensure the network does not induce significant jitter to the NTP packets.

11.5 Operating System Enhancements

- ***Apply the nanokernel for time critical systems***

The nanokernel is an operating system patch extending the current kernel to have nanosecond resolution. The higher resolution enables more precise measurements for the protocol as well as smaller adjustments to the system clock, both of which would improve synchronization accuracy.

11.5.1 Hardware Enhancements

- ***Sufficient clock quality for each node***

For systems with limited computing resources but stringent timing requirements, the clock should have sufficient resolution and stability such that it does not need to be synchronized frequently.

- ***Time-critical systems should have a fallback clock***

Selected hosts should be backed up with external clocks. With unreliable synchronization sources such as the primary and secondary sources, these hosts will still be able to maintain a sense of time with reasonable accuracy.

12 CONCLUSION

NTP continuously evolves to meet the demands of distributed network synchronization. The current NTP working group is also exploring the ability to work with other mainstream time synchronization protocols. This will allow the community interested in synchronized time to freely select which protocol is needed under their specific circumstance. It also will allow the ability to select the most efficient protocol to meet their demands. With the nanokernel software, NTP is able to achieve best-case accuracy on the order of nanoseconds. However, because it is a pure software implementation, it is difficult to guarantee a worst-case accuracy below 1 ms.

The synchronization accuracy achieved with NTP is directly related to how it is deployed in the network. To achieve the desired accuracy, careful understanding of the key accuracy factors is therefore essential. For NTP to perform at its full potential, network latencies must be carefully measured, and network and clock jitter must be minimized. For more stringent applications, hardware redundancy improves determinism within the inherently non-deterministic Internet-based protocol. As the IT industry trend continues to rely more on distributed applications, time

synchronization capabilities will offer a distinct advantage by providing accurate time stamps for quality of service, legal transactions, diagnostics, security, etc. Therefore with a concrete infrastructure for deploying synchronized time, enterprises will be able to reap the benefits of future distributed technology trends.

13 REFERENCES

- [1] Semiconductor Factory and Equipment Clock Synchronization for e-Manufacturing.
<http://ismi.sematech.org/docubase/abstracts/4557aeng.htm>
- [2] *Some Frequently Asked Questions about RTP*.
<http://www.cs.columbia.edu/~hgs/rtp/faq.html>
- [3] C.J. Mitchell, "Timestamps and Authentication Protocols."
<http://www.ma.rhul.ac.uk/techreports/2005/RHUL-MA-2005-3.pdf>
- [4] Windows Time Service Technical Reference.
<http://www.microsoft.com/technet/prodtechnol/windowsserver2003/library/TechRef/a0fcd250-e5f7-41b3-b0e8-240f8236e210.mspix>
- [5] Network Time Protocol Version 4 Protocol Specification, (Internet draft).
<http://www.ietf.org/internet-drafts/draft-ietf-ntp-ntpv4-proto-01.txt>
- [6] D. Mills, Simple Network Time Protocol (SNTP) version 4 for IPv4, IPv6, and OSI.
<http://www.eecis.udel.edu/~mills/database/rfc/rfc-xxxx.pdf>
- [7] Simple Network Time Protocol (SNTP) Version 4, RFC 2030, October 1996.
<http://www.apps.ietf.org/rfc/rfc2030.html>
- [8] *Network Time Protocol: Best Practices White Paper*, Document ID: 19643.
http://www.cisco.com/en/US/tech/tk869/tk769/technologies_white_paper09186a0080117070.shtml
- [9] M.A. Lombardi, *NIST Time and Frequency Services*, NIST Special Publication 432, 2002.
<http://tf.nist.gov/timefreq/general/pdf/1383.pdf>
- [10] D. Mills, Reference Clock Drivers. <http://www.eecis.udel.edu/~mills/ntp/html/refclock.html>
- [11] D. Mills, ntpd – Network Time Protocol Daemon.
<http://www.eecis.udel.edu/~mills/ntp/html/ntpd.html>
- [12] D. Mills, *Internet Time Synchronization: the Network Time Protocol*, *IEEE Trans. Communications COM-39, 10* (October 1991), 1482-1493.
<http://www.eecis.udel.edu/~mills/database/papers/trans.pdf>
- [13] P. Rybaczuk, *Expert Network Time Protocol: An Experience in Time with NTP*, Springer-Verlag, New York, 2005.
- [14] D. Mills and P. Kamp, *The Nanokernel*.
<http://www.eecis.udel.edu/~mills/database/papers/nano/nano2.pdf>
- [15] *Notes on setting up an NTP subnet*. <http://www.eecis.udel.edu/~mills/ntp/html/notes.html>
- [16] D. Mills, *Precision Time Synchronization*. <http://www.eecis.udel.edu/~mills/precise.html>
- [17] P. Work and K. Nguyen, *Measure Code Using Enhanced Timer*.
<http://www.intel.com/cd/ids/developer/asmo-na/eng/dc/code/209859.htm?page=3>
- [18] D. Mills, *The Autokey Security Architecture, Protocol and Algorithms*, RFC Draft.
<http://www.eecis.udel.edu/~mills/database/reports/stime/stime.pdf>

- [19] D. Mills, *Monitoring Options*. <http://www.eecis.udel.edu/~mills/ntp/html/monopt.html>
- [20] Transmission Control Protocol, RFC 793. <http://www.ietf.org/rfc/rfc0793.txt?number=793>
- [21] User Datagram Protocol, RFC 768. <http://www.ietf.org/rfc/rfc0768.txt?number=768>
- [22] *What Time Is It?* Bureau International des Poids et Mesures.
http://www.bipm.fr/en/scientific/tai/time_server.html
- [23] D. Plonka, Requirements for Network Time Protocol Version 4 Draft, October 24, 2005.
<http://www.ietf.org/internet-drafts/draft-ietf-ntp-reqs-01.txt>

Appendix A – Definitions

A.1 Acronyms

<i>APC</i>	Advanced Process Control
<i>CMP</i>	Chemical Mechanical Planarization
<i>CPU</i>	Central Processing Unit
<i>DNS</i>	Domain Name Service
<i>DSA</i>	Digital Signature Algorithm
<i>ERP</i>	Enterprise Resource Management
<i>FDC</i>	Fault Detection Classification
<i>GPS</i>	Global Positioning System
<i>IGMP</i>	Internet Group Management Protocol
<i>IP</i>	Internet Protocol
<i>IPv6</i>	Internet Protocol version 6
<i>LAN</i>	Local Area Network
<i>MD5</i>	Message Digest algorithm 5
<i>MES</i>	Manufacturing Execution System
<i>NIST</i>	National Institute of Standards and Technology
<i>NTP</i>	Network Time Protocol
<i>OSI</i>	Open Systems Interconnection.
<i>PKI</i>	Public Key Infrastructure
<i>RSA</i>	A public key encryption algorithm founded by Ron <u>R</u> ivest, Adi <u>S</u> hamir, and Len <u>A</u> dleman
<i>SNMP</i>	Simple Network Management Protocol
<i>SNTP</i>	Simple Network Time Protocol
<i>SPC</i>	Statistical Process Control
<i>TCP</i>	Transmission Control Protocol
<i>TTL</i>	time-to-live
<i>UDP</i>	User Datagram Protocol
<i>UTC</i>	Universal Time Coordinated
<i>W32Time</i>	Windows Time Service
<i>WAN</i>	Wide-Area Network

A.2 Terminology

accuracy: Proximity of the clock's time to the absolute value of the offset from true time such as that defined by Universal Time Coordinated (UTC) and traceable to atomic time defined by a national standard.

daemon: A computer program that is designed to run in the background. It is often initiated at boot time as a background process, without the need for direct user control. In NTP, *ntpd* is the daemon for synchronizing the system clock.

dispersion: A metric, denoted in seconds, of how scattered the time offsets have been from a specific time server.

drift: Measurement in the variation of clock frequencies with respect to a true time, denoted in hertz per second.

host: A system with the NTP protocol deployed.

jitter: Variation in network latency. In terms of clock oscillators, jitter describes the short-term variations in frequency and can be quantified as the exponential average of the root mean square (RMS) differences among samples in a sliding window of time measurements [23].

MD5: A cryptographic hash function for securely authenticating identity or verifying data integrity. The hash function takes an input of any length and produces a 128-bit output.

network latency: The total time delay for a packet to travel from the source application to the destination application.

node: A device with a local processor.

offset: Difference in time between two clocks.

OSI: A computer communications model consisting of seven layers based on its function.

peers: Time servers at the same stratum level.

precision: The ability to maintain low offsets within a distributed timing system.

provenance: The verification of an NTP host's origin.

reliability: Ability of the clock to maintain operation and network connectivity.

resolution: Smallest unit of time by which a clock is capable of being updated. Resolution is typically defined in seconds, so a 1 ms resolution implies the clock updates time in 0.001 second increments, but does not necessarily imply this is the true amount of time between updates.

skew: The frequency difference between two clocks, which can be measured in hertz or *parts per million* (ppm).

slew: Adjustment of the rate at which time is incremented to gradually correct an offset.

stability: Ability of a clock to maintain a constant frequency, which can be indicated by the skew.

step: A change in the current time to immediately adjust an offset.

stratum: The number of levels, or distance, an NTP host is away from a source of UTC time.

system latency: The total time delay for a system process to wait for other system processes.

TCP: A reliable, connection-oriented transport protocol of the OSI model [20].

TTL: An 8-bit field in the IP header used to specify an upper bound on the time the IP datagram can exist in the network. If a TTL reaches 0 when it is received by the host, the packet is discarded.

UDP: An efficient, connectionless transport protocol of the OSI model [21].

UTC: Maintained by the Bureau International des Poids et Mesures (BIPM), a time scale based on the coordinated input of standard frequencies and time signals from standards organizations around the world. UTC is also adjusted for leap seconds to approximate the time derived based on the rotation of the Earth [22].

wander: Clock oscillator performance based on frequency variation measurements taken over a long period of time. In NTP, wander is the exponential average of the root mean square (RMS) differences among samples in a sliding window of frequency measurements [23].

**International SEMATECH Manufacturing Initiative
Technology Transfer
2706 Montopolis Drive
Austin, TX 78741**

**<http://ismi.sematech.org>
e-mail: info@sematech.org**