Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Proyecto:

Implementación del algoritmo de Mills para la sincronización de relojes físicos.

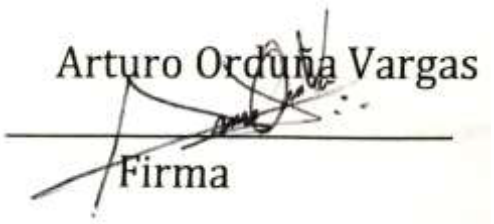
Modalidad: Proyecto Tecnológico

Alumno:
Arturo Orduña Vargas
210328436
al210328436@correo.azc.uam.mx

Asesor:
M. en C. José Alfredo Estrada Soto
Titular C
Departamento de Electrónica
jestrada@correo.azc.uam.mx
Núm. eco. 22003

Yo, José Alfredo Estrada Soto, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Yo, Arturo Orduña Vargas, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.



Resumen

Las mediciones de tiempo a través de medios computacionales han llevado a estudiar una variedad de herramientas de software y hardware propias de los equipos actuales. La sincronización de relojes físicos es una tarea compleja cuando la información y el procesamiento se mantienen en diferentes nodos. Por ello, Mills retoma algunos principios de los algoritmos actuales y propone una solución que contemple el uso de recursos tales como el NTP. En este trabajo se muestra la forma de realizar la sincronización de relojes físicos a través de los algoritmos de Mills.

Índice General

Índice de Figuras

Introducción

Para mejorar la velocidad de procesamiento y comunicaciones en aplicaciones distribuidas, actualmente se cuenta con una serie de propuestas que contemplan mediciones de tiempo que los equipos actuales no pueden resolver de manera autónoma. En un sistema distribuido la exactitud es un atributo que dependerá de cuán sincronizados se encuentren los relojes del sistema, se trata por así decirlo, de un atributo del "reloj distribuido" del sistema.

Disponer en una computadora de una referencia de tiempo ajustada a una escala conocida (hora) ha sido, además de una necesidad, una realidad desde los primeros equipos. Casi todas ellas tienen implementado algún tipo de dispositivo para cumplir esta función (reloj). Con el advenimiento de las comunicaciones entre computadoras, aparecen los sistemas distribuidos. Un sistema distribuido se define como una colección de computadoras autónomas conectados por una red, y con un software distribuido adecuado para que el sistema sea visto por los usuarios como una única entidad capaz de proporcionar el compartimiento fácil y eficiente de los recursos. Es importante mencionar que, para que un sistema sea distribuido debe cumplir con tres requisitos:

Inexistencia de Reloj Global: Los programas que necesitan cooperar coordinan sus acciones mediante el intercambio de mensajes, que depende de una idea compartida del instante en el que ocurren las acciones de los programas.

Concurrencia: Es realizar varios trabajos en distintas computadoras al mismo tiempo. Fallos independientes: Los sistemas pueden fallar de distintas formas, las fallas en la red producen aislamiento, lo que permite que los demás equipos continúen sin que detengan su ejecución.

Los avances en hardware de comunicaciones han posibilitado que la resolución de los tiempos a medirse realice en el orden de los microsegundos. Sin embargo, se debe tener en cuenta que no es facil lograr resoluciones de microsegundo en un reloj distribuido, esto es por que la tarea de sincronizar relojes implica un compromiso entre varias variables como lo son: precisión, tolerancia a fallas, sobrecarga del sistema, exactitud, fiabilidad, interoperabilidad y seguridad, lo cual hace que por optimizar un aspecto, se degrade otro.

Justificación

En el mundo actual en el que vivimos, el concepto de sincronización es aplicable en muchos aspectos de nuestra vida cotidiana. Por ejemplo, requerimos la hora exacta para poder llegar a tiempo a nuestros trabajos, a nuestra escuela, a las horas de comida, a las citas con otras personas, entre otras. Es así que los relojes de computadoras por defecto, al igual que los relojes convencionales con los que los seres humanos vivimos el día a día, tienden a estar en desacuerdo con otros relojes; éstos tienen una diferencia de tiempo que puede ser considerada o no, y por tanto, también puede ser en diferentes unidades de tiempo, dependiendo la granularidad del reloj, a tal diferencia se le llama Sesgo del Reloj.

La razón de la divergencia de los relojes de las computadoras, depende de un oscilador de cristal y la convergencia de cada una de las oscilaciones de un reloj respecto a las de otro es casi imposible.

El sesgo entre los relojes es extremadamente pequeño, sin embargo, no es una oscilación la que provoca una diferencia entre relojes significativa y visible, sino la acumulación de éstas oscilaciones de un periodo de tiempo determinado, la hora con la que fueron inicializados los relojes es independiente a las variaciones que puede tener el tiempo en un futuro por consecuencias del sesgo del reloj.

Una fuente de hora en una red esta caracterizada por el jitter, la derivada y la confiabilidad. El jitter es el valor cuadrático medio de una serie de diferencias de horas de los relojes, mientras que la derivada es el valor cuadrático medio de las frecuencias del reloj. En tanto que la confiabilidad de un sistema de sincronización de hora, es la fracción de tiempo en la que está conectado y operando correctamente en cuanto a tolerancias de estabilidad y precisión.

Objetivo General

Construir un sistema que permita la sincronización de relojes físicos a través de la implementación de los algoritmos de Mills.

Objetivos específicos

- Analizar la granularidad en tiempo de los diferentes sistemas operativos que se van a utilizar.
- Implementar un proceso maestro, para realizar un conjunto de procesos corriendo en distintos procesadores.
- Diseñar e implementar un mecanismo de comunicación entre procesos que determine los tiempos locales de los relojes esclavos.
- Construir un mecanismo que sea tolerante a fallos para eliminar las lecturas de los relojes incorrectas.
- Diseñar e implementar un mecanismo que actualice los relojes, tanto del maestro como el de los esclavos.
- Implementar un lazo enganchado en fase, para proveer una forma de ajustar el reloj local en hora y frecuencia en respuesta a las correcciones del protocolo de sincronización.
- Estimar la precisión del protocolo mediante pruebas al sistema.

Trabajos Relacionados

Se han realizado investigaciones de las que han surgido diferentes algoritmos e implementaciones [1][2]. Una de las primeras fue la relacionada con el algoritmo de sincronización de computadoras de Lamport.

El algoritmo de Lamport [3] tiene como objetivo alcanzar un ordenamiento de los sucesos en un sistema distribuido. Cuestiones tales como hora real y resolución, son dejadas de lado. En aplicaciones distribuidas en las cuales el orden de eventos puede llevar a diferentes estados finales, este algoritmo representa una posible solución; como por ejemplo en el caso de las transacciones bancarias, en las cuales un depósito y una extracción en una cuenta en diferente orden generan o un crédito o un débito de intereses.

La cuestión de la hora real en computadoras es un tema heredado de los problemas con respecto a la posición de navegación marítima y más tarde, aérea.

Para determinar la latitud no hay implicaciones de hora pero respecto de la longitud ecuatoriana surge la necesidad de saber la hora real para determinar la posición.

Este problema encuentra solución con la aparición de las transmisiones de radio y la consecuente sincronización, que se consigue enviando, por este medio, referencias con destino a los relojes a sincronizar en las naves.

En las transmisiones de referencias de hora a través de radio, dado que las transmisiones requerían de un cierto tiempo en arribar a destino, se recurría a medir el tiempo de ida y vuelta mediante un mensaje de respuesta, enviado desde el destino en forma inmediata después del mensaje recibido. Una vez recibida esta respuesta, el emisor primario puede calcular el tiempo de ida como la mitad del tiempo de ida y vuelta. Gracias a este dato se puede determinar la verdadera hora en base a dicha referencia de hora de radio. Esta referencia de hora así transmitida puede ser capturada por una computadora conectada a un receptor de radio. Este tipo de hardware presenta el inconveniente de su precio, además de requerir que la computadora ocupe gran parte del tiempo en manejar la comunicación con la radio.

Gracias a la disponibilidad de este tipo de dispositivos y a la aparición de redes de computadora, se analiza la posibilidad de distribuir la referencia de hora radial o de cualquier tipo a través de una red, en un modelo cliente-servidor [4]. Se hace hincapié en que el servidor proporcione a cada cliente una referencia con el menor error y, en lo posible, caracterizado. Como toda arquitectura cliente-servidor, tiene un cuello de botella en el servidor, lo cual complica escalar el algoritmo. Además, ante una falla del servidor, deja al sistema sin referencia de hora.

En función de la robustez del sistema, el Algoritmo de Berkeley [5] proporciona una manera de obtener, sin una referencia externa, una hora similar entre las máquinas de una red local.

Se transmiten diferencias de horas desde el servidor a los clientes para dar la referencia de corrección, con lo que no influyen los tiempos de transmisión de los paquetes.

El algoritmo se ve enriquecido en cuanto a robustez por el agregado de algoritmos de elección de un nuevo servidor en caso de caída o fallo del mismo.

En las últimas décadas se ha trabajado en la elaboración de diferentes versiones del protocolo NTP (Network Time Protocol) [6][7][8], en el cual se encuentran, implementadas también cuestiones como la de obtener una hora mas exacta entre un arreglo de relojes computadoras del que tendrían cada una por separado[10] (varianza de Allan). En el protocolo NTP se evita el cuello de botella de los sistemas cliente-servidor a través de un sistema jerárquico de estratos. En cuanto a la robustez se utilizan algoritmos de elección, permitiéndose en la configuración especificar varios servidores.

Sincronización

El tiempo es un argumento irreversible que afecta globalmente a todas las actividades humanas y es componente clave para todas las relaciones causales entre procesos. Las relaciones de dependencia entre unos hechos y otros son función del orden en el que se realizan cada uno de ellos y suelen ser manifestación de las relaciones causales que los unen.

Por este motivo, la sincronía es un concepto que se puede llevar a diferentes parámetros como pueden ser: instantes, lugares, longitudes, contenidos, entre otros. Por ejemplo, dos personas u objetos pueden estar sincronizados si se encuentran en el mismo lugar, es decir, si se llegara a presentar el caso de la celebración de un examen o prueba evaluatoria.

En este tipo de escenarios, todos los participantes deben tener las mismas oportunidades, por lo que se les reúne simultáneamente en un mismo lugar, se les plantean las mismas preguntas y se les otorga el mismo tiempo para confeccionar sus respuestas.

Por lo que al final del periodo marcado, la igualdad de oportunidades para cada uno de los participantes son las mismas, sin afectar o modificar su rendimiento, esto es debido a que el inicio de la prueba fue sincronizado para todos los participante en el mismo tiempo y el termino de la misma sin modificar o dar ventaja a los participantes, dando origen a la necesidad de marcar instantes, tales como el inicio o fin de un proceso.

Por lo que es importante establecer métodos por los cuales los procesos de cada uno de los equipos del sistema, lleguen a una sincronía en el tiempo en el que ocurren cada uno de los procesos que se ejecutan.

Una manera más sencilla de poder tener esta sincronía en el tiempo para los procesos, es establecer un sistema centralizado en el que toda la información y los procesos se alojen en un servidor, y los otros equipos sean, clientes o terminales tontas que soliciten la información a ese servidor. Más sin embargo, cada una de estas sesiones que ejecutara procesos en el mismo servidor carecerá de autonomía, por que no cuenta con un registro ni mucho menos una administración.

La sincronización de relojes en ambientes distribuidos es un tema que presenta variadas soluciones en función de los requerimientos de los algoritmos distribuidos.

- La información relevante se distribuye entre varias máquinas.
- Los procesos toman las decisiones solo con base en la información disponible en forma local.

La sincronización de relojes de computadoras puede entenderse desde dos puntos de vista:

Sincronización interna: es el resultado de obtener en diferentes máquinas las mismas referencias de tiempo para un instante dado.

De esta manera, al alcanzar la sincronización interna con un error acotado y conocido, se pueden medir eventos como el tiempo de transmisión de un mensaje, en donde las dos marcas de hora serán producto de dos relojes ubicados en máquinas distintas pero sincronizadas.

Sincronización externa: si se desea saber en una máquina en particular a qué hora del día sucedió un evento, es necesario sincronizar la hora de esa máquina con algún reloj o fuente de hora autorizada, en donde permite hacer peticiones a una hora autorizada y hacer sus correcciones de forma correcta y eficiente.

Por lo que, la diferencia entre un sistema centralizado y uno distribuido es el lugar en donde se almacena toda la información y desde donde se ejecutan los procesos y programas. Para el caso del sistema centralizado toda la información se almacena servidor un arreglo local de servidores que suministran información a cada uno de los clientes que lo solicitan, de igual forma si los clientes necesitan utilizar algún programa o aplicación, quien arrancará la ejecución de los mismos y tendrá los procesos que estos genere será el servidor que centraliza la información, lo que causa que toda la administración de recursos la lleve dicho centro de datos y tenga más riesgo de fallar y que la suministración de información será considerablemente afectada o suspendida.

En un sistema distribuido el almacenamiento de la información y la ejecución de programas se encuentran los diferentes modos el sistema, esto implica que la administración de los recursos de los que se hace uso, necesitan una coordinación de eventos y esto parte de una sincronización.

Es muy importante definir los intervalos de tiempo en los que uno de los nodos comienza a usar un recurso y hasta que momento lo hace para que el recurso este disponible para los demás nodos.

Óbien, si este ya se encuentra solicitado por otro nodo, inmediatamente que se termine de proporcionar el servicio al primer solicitante seguirá al segundo en forma sucesiva hasta poder cumplir el número de solicitudes pendientes.

La sincronización de los procesos en un sistema distribuido no es trivial ni automática como para el caso de los sistemas centralizados, en los cuales para saber la hora del sistema o el tiempo de la ejecución del programa únicamente se solicita el tiempo del sistema o se calcula el intervalo de tiempo respectivamente.

Para un sistema distribuido es necesario hacer un análisis sobre la arquitectura que se tiene en la red que conecta a cada uno de los nodos, sobre la fuente de información, su colocación, la arquitectura de servidores y finalmente el análisis de las plataformas y los procesadores con lo que trabaja cada uno de los equipos de los nodos.

Es preciso diseñar procedimientos para obtener la hora del sistema de cada uno de los nodos y para poder tener un punto de partida, posteriormente ya que no hay un centro de datos de Coordinador, hay que diseñar el procedimiento para elección de un coordinador en el sistema, con los relojes sincronizados en todos los del sistema, todas las ejecuciones y transferencia de datos estarán sincronizadas, por lo menos en el tiempo que es el objetivo primordial de este proyecto terminal.

Es importante mencionar que los relojes se sincronizan acelerando o retrasando una variable llamada Derivada de Reloj según sea necesario, partiendo de la información del tiempo que tenga el nodo Coordinador o el UTC.

Relojes

El principio de la sincronización u ordenamiento en el tiempo son las marcas de tiempo en donde hay dos variables importantes: el día y la fecha. Los equipos de cómputo por defecto disponen de un reloj del sistema que toma la hora en la configuración que el usuario le asigne por defecto, por modificación o por instalación, ese es el inicio del intervalo de tiempo del sistema, pero la forma en la que es ese reloj obtiene la medición del tiempo, es mediante el reloj físico de la computadora, que es un oscilador de cristal y en donde cada una de sus oscilaciones está asociada a un segundo para sistema.

La razón de la divergencia de los relojes de las computadoras es porque como se dijo antes, dependen de un oscilador de cristal y la convergencia de cada una de las oscilaciones de un reloj respecto al del otro es casi imposible. Las oscilaciones de un reloj físico son almacenadas en un contador, el cual es el encargado de transformarlas en segundos y aquí se establece un nuevo contador el cual cada 60 segundos se forma un minuto y así sucesivamente hasta la hora. El sesgo entre los relojes es extremadamente pequeño, sin embargo no es una oscilación la que provocó la diferencia entre los relojes significativamente y visible, sino la acumulación de estas oscilaciones a lo largo de un periodo de tiempo determinado, la hora con la que fueron inicializado los relojes es independiente a las operaciones que pueden tener el tiempo en un futuro por consecuencias del sesgo del reloj. El ritmo de la derivada del reloj es la compensación necesaria para la sincronización de dos o más relojes en un sistema distribuido.

Dicha variable es la que se altera para llegar a la sincronía en un sistema independiente del número de nodos que haya, para poder realizar esta operación es necesario tener un reloj de referencia.

Las fuentes de tiempo externas de alta precisión como los relojes atómicos son una herramienta fundamental para llegar a la sincronización de tiempo, también conocidas como Tiempo Universal Coordinado (UTC), mediante la alteración del ritmo de la derivada de los relojes a partir del sesgo que estos tengan con el reloj atómico, por lo que estas fuentes de tiempo externas serán de mucha utilidad para mejorar la precisión y garantizar la sincronización.

Sincronización de Relojes Físicos

La medición del tiempo ha tenido una gran importancia en las personas por conocer los instantes en los acontecen las actividades personales de cada una de las personas, tanto en su vida laboral como en su vida personal.

Es por esto que las personas a diario hacen uso de un reloj que les permita saber la hora en todo momento, y el cual les permita organizar sus tiempos para poder llegar a sus trabajos, hogares, a los estudios e incluso en los descansos, por lo que tener con precisión y de forma sincronizada cada uno de los relojes, permite tener a las personas reunidas de forma simultánea en un mismo lugar y en el mismo momento para realizar cada una de su actividades diarias.

Por lo que para el caso de las computadoras, la utilización de este principio es el mismo, ya que necesitan tener una referencia de tiempo, para llevar al resto de los relojes a la sincronía general. Por lo cual, para este proyecto terminal en particular, la granularidad que se va a manejar, será la de milisegundos, ya que el recurso primordial para el que se esta buscando la sincronización entre los relojes que tiene cada uno de los equipos, es representado a través de cada uno de los nodos.

Algoritmos de Sincronización

En los últimos años se ha abordado el problema de la sincronización de relojes de computadoras desde diferentes puntos de vista. En esta sección se exponen los principales algoritmos que resuelven el problema:

Algoritmo de Lamport

Es uno de los primeros propuestos para la sincronización de sistemas distribuidos y se basa en la relación "sucede antes" más la utilización de los mensajes entre las computadoras como indicadores precisos de esta relación.

Más específicamente, un mensaje no puede ser recibido antes de ser enviado y, por lo tanto, si se tienen marcas de tiempo de los envíos de los mensajes se puede verificar si el tiempo actual es coherente con la definición de la relación "antes de". Para ello se definen relojes lógicos. Un resumen de lo que realiza el algoritmo sería[23]:

• Se analiza qué sucede antes

A->b significa a antes de b

Si en un proceso el evento b sucede después de a, entonces a->b es verdadero

Si a identifica el evento de enviar un mensaje y b el de recibirlo, ab es verdadero (no puede ser recibido antes de ser enviado)

Si asigno valores en el tiempo C(a) y C(b), $C(a) < C(b)$

Si no se cumple, adiciono el valor necesario a C(b).

Nunca resto, ya que el tiempo debe ser siempre creciente.

En un mismo proceso, para dos eventos a y b C(a) debe

ser diferente de C(b) (exigencia de resolución de reloj).

Las tareas a llevar a cabo en cada computadora son relativamente sencillas, aunque afectan, en cierta manera, la forma en que se procesan los mensajes:

1.Se tiene un reloj local.
2.Cada vez que se envía un mensaje, se le agrega al mismo una marca de tiempo (timestamp) con el tiempo local del que envía.

3. Cada vez que llega un mensaje, se analiza la marca de tiempo del que envió, y

a) si la marca de tiempo es menor que el tiempo local, se asume que las computadoras están sincronizadas.
b) si la marca de tiempo es mayor o igual que el tiempo local, se cambia el tiempo local con la marca de tiempo del mensaje que se recibió más 1.

En la ilustración1 se observa esta mecánica en acción sobre tres hosts identificados como “0”,”1” y “2”. A la izquierda se observa el intercambio de mensajes y a la derecha los cambios realizados una vez que se intercambiaron los mensajes:

El mensaje A va de host0 a host1. Como el reloj es mayor que su timestamp, no lo cambia.

Lo mismo el mensaje B de host1 a host2.

El mensaje B tiene un timestamp de 60 saliendo de host2 y llega a host1 que tiene 56, lo que implica que tenga que ajustar su reloj local a 61, o sea 60 + 1.

El D lleva desde host1 un timestamp de 69, al llegar a host0 provoca un ajuste del reloj local a 70, o sea 69 + 1.

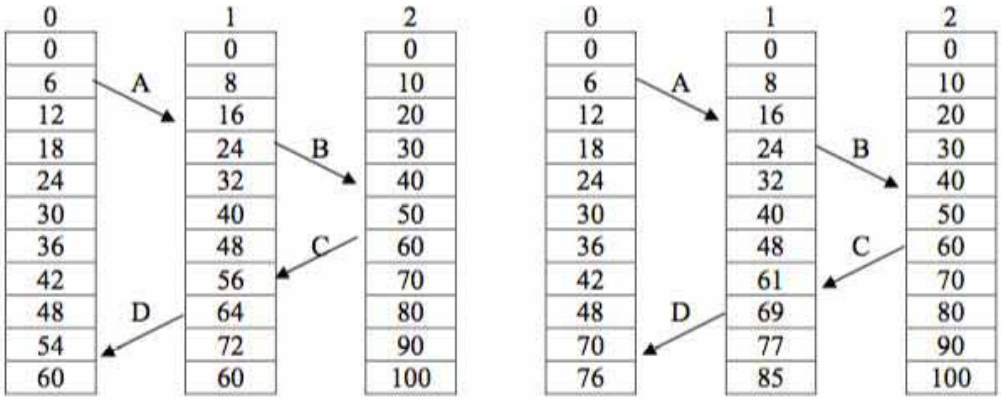


Ilustración 1 Intercambio de mensajes

La ilustración2 muestra más en detalle el avance del tiempo en dos computadoras, C1 y C2, y la forma en que la computadora C2 cambia su tiempo local a partir de la llegada de un mensaje con una marca de tiempo mayor que su tiempo local.

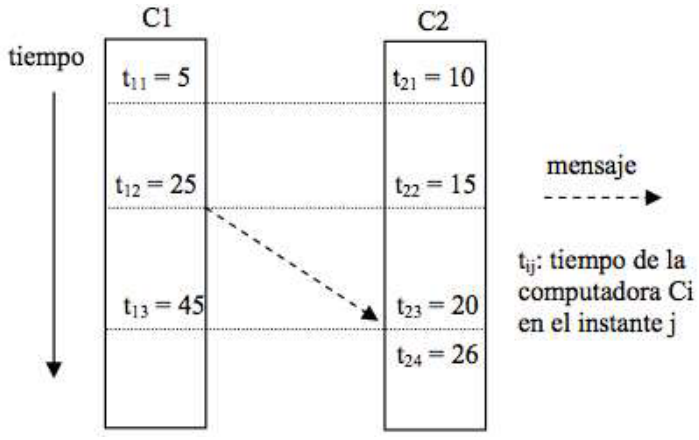


Ilustración 2 Avance del Tiempo C1 C2

Este algoritmo soluciona el problema de la escalabilidad, ya que solo exige un mensaje entrante y uno saliente para sincronizar cada máquina, pero no tiene un orden total de los eventos que suceden entre diferentes computadoras, excepto los relacionados directamente con envío y recepción de mensajes. Por ejemplo, en el caso de la ilustración 2, no es posible determinar la relación de tiempos entre los eventos que suceden en la computadora C1 entre los instantes de tiempo t12 y t13 con los que suceden en la computadora C2 entre los instantes de tiempo t21 y t23.

Algoritmo de Cristian

Cristian definió los conceptos de sincronización interna y externa[11], ya vistos. Recordamos que sincronización interna se refiere a mantener un grupo de relojes sincronizados, no importa qué hora tengan pero que en el grupo sea la misma, o con un margen de diferencias acotado a algún valor conocido. La sincronización externa podría definirse como mantener sincronización interna con alguna fuente fiable de hora tipo UTC.

En este marco de definición, Cristian propone sincronizar un conjunto de relojes de máquinas a partir de una que esté sincronizada externamente (tiempo u hora real), a través de la red de comunicaciones de datos entre computadoras.

El algoritmo de Cristian es probabilístico ya que no garantiza que un procesador pueda leer un reloj remoto con una precisión específica. Sin embargo, intentando una cantidad de veces suficiente, la hora recuperada puede ser leída con una precisión dada, con una probabilidad cerca de uno, según lo deseado.

Por otro lado, no se sabe cuánto es el máximo tiempo de envío de un mensaje entre dos computadoras. Algunos algoritmos suponen un tiempo máximo de envío. Son algoritmos determinísticos en el sentido que suponen que siempre llega el mensaje con la referencia en un tiempo menor al tiempo máximo. No pueden ser usados para sincronizar sistemas asincrónicos, ya que es imposible determinar el tiempo máximo. Estos algoritmos suponen un intercambio de mensajes, para n procesos a sincronizar, de n^2.

Esto dificulta la escalabilidad en redes con gran número de nodos.

Cristian asume que:

1) El tiempo mínimo de demora de un mensaje t_{min} es conocido.

2) La función de distribución de la demora en los mensajes es conocida.

Esto lo asume en función de haberlo probado experimentalmente en forma estadística: se envían sucesivos paquetes hacia una máquina que los contesta en la forma mas inmediata posible.

Se registra la hora de salida del paquete y la hora de recepción de la respuesta, computándose por diferencia el tiempo de ida y vuelta (round trip time, rtt). Se supone que los valores de moda, mínimo y máximo encontrados serán constantes para tiempos posteriores.

Si un proceso q recibe un mensaje m desde un proceso p, evidentemente enviado por p, m sufre una demora, arbitraria y aleatoria.

Definimos:

- ρ como la tasa de deriva de la frecuencia del reloj local de p, este valor se mide en partes por millón (ppm) y por lo general lo caracteriza el fabricante.
- $H_p(t)$ es el valor que se lee en el contador del reloj de hardware de p en el tiempo real t. El tiempo real t es definido de igual manera que el tiempo verdadero, como el más cercano a algún estándar como UTC.
- $H_p(s)$, el medido en el tiempo s. Se asume una máxima demora σ en la cual un proceso es despertado por el sistema operativo para darle el uso de la CPU, o sea, una demora del planificador máxima.

De acuerdo a este algoritmo y definiciones anteriores, el tiempo medido por un reloj será correcto respecto del tiempo real t, si está dentro del intervalo:

$$(1-\rho)(t-s) \leq H_p(t) - H_p(s) \leq (1+\rho)(t-s)$$

Esta fórmula marca los límites de la deriva o drift de un reloj. Por otrolado, si un proceso en tiempo t adquiere una medida del timer de W unidades de tiempo, el sistema operativo despertará al proceso en el intervalo de tiempo:

$$[t + (1-\rho)W, t + (1+\rho)W + \sigma]$$

Pueden ocurrir tres tipos de fallas en un proceso que trata de sincronizar:

- Por rotura: cuando ante la pérdida de un evento no puede recuperarse.
- Por performance: cuando reacciona ante el evento muy lentamente, posiblemente por culpa del sistema operativo y excede la demora σ.
- Arbitraria: Son las demás fallas. Por ejemplo un proceso que reacciona demasiado rápido o que envía información errónea.

Un reloj correcto será consecuencia de un proceso correcto, o sea, sin fallas.

Tomando en cuenta estos intervalos, se puede describir la idea básica del algoritmo como:

i. Un proceso p (cliente) envía a un proceso q (servidor) una solicitud de hora en un mensaje m_1 en su tiempo local t_0.
ii. El proceso q consulta su reloj local y contesta con un mensaje m_2 con su tiempo local t_1.
iii. Al llegar el mensaje m_2 al cliente p, este toma su tiempo local t_2.
iv. Estima la demora en llegar el mensaje con la hora remota en $D/2 = (t_2 - t_0)/2$
v. Si el error en la lectura está dentro de un error constante A, el algoritmo toma el valor como correcto; caso contrario, lo descarta.

El algoritmo supone que se retransmitirá varias veces el pedido de consulta de referencia remota hasta que dé con error aceptable, y si transcurren una cantidad de tiempo excesiva sin que ello ocurra, se considera que hay una falla en la lectura remota de la referencia.

Con éste valor corrige su hora, no en el reloj de hardware local sino en un reloj virtual o reloj de software, que no es sino una posición de memoria asociada al proceso que corrige la hora. Ilutración 3.

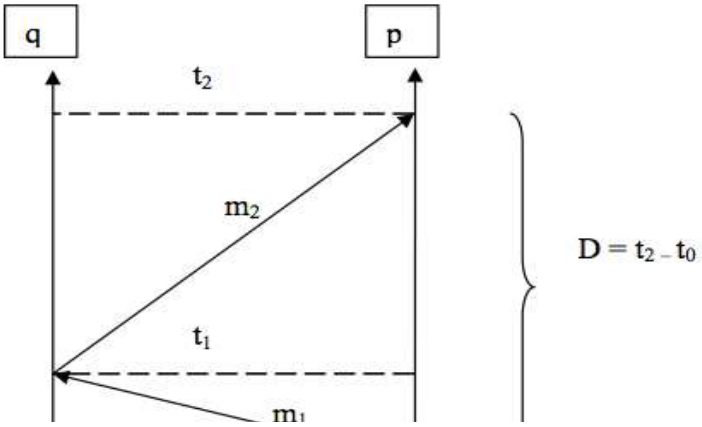


Ilustración 3Algoritmo de Cristian

Corrección de hora

Por ello este método es estadístico, ya que estima que con probabilidad cercana a 1 conseguirá un valor dentro de la banda de error A, la cual definiremos a continuación. Para definir esta banda debemos considerar que el proceso p puede derivar su hora real en proporción al parámetro ρ, que mide las partes por millón en que varía la frecuencia de un oscilador a cristal. De esta manera, el tiempo de ida y vuelta (límite superior de A) no puede ser mayor a:

$(t_2\text{-}t_0)(1+\rho)$

Este límite puede ser usado para determinar un tiempo máximo para la demora de la transmisión del mensaje en un sentido del mensaje m_2. El tiempo min surge de estadísticas sobre mensajes, o sea, es la mitad del mínimo rtt, estimado por el reloj local del cliente. En las siguientes ecuaciones, se destaca que los tiempos serían los del reloj local del cliente:

$\max(t(m_2)) \equiv (t_2 - t_0)(1+\rho) - \min$

- La condición $t(m_2) \leq \max(m_2)$ se cumple ya que
  $(t_2 - t_0)(1+\rho) \geq t(m_1) + t(m_2)$ y $\min \leq t(m_1)$,
  por la definición de tiempo de transmisión mínimo min.

- Estos valores min y max pueden ser usados para aproximar el valor del reloj del proceso q en el tiempo t_2.
- El valor del reloj de q se incrementa por min(1-ρ) al menos y por max(m_2)(1-ρ) cuanto mucho durante la transmisión de m_2.

Por lo tanto, A estará limitado por estos valores de [min, max].

$\min(1-\rho) < A < \max(m_2)(1-\rho)$

O sea que ρdel reloj local del cliente afecta a ambas medidas. Para minimizar el peor caso de error que p tiene en la estimación, el reloj de q en la hora t_2, simbolizado como C_q (t_2 ,p), estará definido como el punto medio del intervalo:

$[t_1 + \min(1-\rho), t_1 + \max(m_2)(1+\rho)]$

Por lo tanto, el punto medio quedaría:

$$C_q(t_2, p) \equiv t_1 + \frac{\max(m_2)(1+\rho) + \min(1-\rho)}{2}$$

Ilustración 4 Ecuación del punto medio

El limite del peor caso de error $C_q(t_{2,\ p})$ que p puede cometer en la aproximación de la hora del reloj de q a la hora t_2, en la estimación de la mitad del intervalo será:

$$E_q(t_2, p) \equiv \frac{\max(m_2)(1+\rho) + \min(1-\rho)}{2}$$

Ilustración 5 Estimación del intervalo

El proceso p espera un cierto tiempo antes y después de enviar un mensaje de referencia para enviar otro.

Si A es el máximo error aceptable, T_0 la hora en que p empieza a estimar la hora de q, t_0 un punto en el tiempo real tal que $C_p(t_0)=T_0$,D el máximo tiempo que p se toma para leer con un cierto error acotado la hora de q, S una constante positiva, T una hora entre T0+DyT0+D+S en la cual p necesita leer la hora de q con un cierto error menor que A. t es un punto en el tiempo real en el cual el reloj virtual de p muestra $T = C_p(t)$.

Para que una lectura probabilística requerida para acceder a p y estimar: C_q (T ,p) del reloj de q con una función de error E_q (T , p) sea correcta, deben satisfacerse lassiguientes condiciones:

- Puntualidad: la lectura del reloj remoto toma a lo sumo D unidades de tiempo del reloj p.
- Límite de error: Si p y q no sufren fallas arbitrarias entre los tiempos T0 y T de la lectura remota, la diferencia entre el valor actual de q y el estimado por p deben ser: $|C_q(t) - C_q(T,p)| \leq E_q(T,p)$
- Manejo del error: si p es correcto pero q falla durante el intervalo $[t_0, t]$ en que dura la lectura, el error es infinito.
- Si ninguno de los dos falla durante el intervalo de tiempo $[t_0, t]$, la probabilidad de que el error sea menor o igual que A es estrictamente positiva:

El algoritmo de Cristian presenta ciertos inconvenientes [10]:

- Al contar con un servidor de hora, si este falla, queda sin referencia. Cristian propone que los clientes hagan los requerimientos a varios servidores y se queden con la referencia que llegue primero.
- La segunda cuestión es con relación al ajuste del reloj. Si la hora de referencia recibida desde el máster es menor a la que se desea ajustar, dicho ajuste implica un atraso, con lo cual dicho reloj pasa dos veces por la misma hora.
  Esto traería aparejados múltiples inconvenientes; entre otros, uno bastante grave: la hora de actualización de archivos. En algunos aparecerían en el futuro modificaciones del pasado, con los inconvenientes que ello presenta, por ejemplo, para utilidades como make. En las implementaciones de este algoritmo, para atrasar, se baja la frecuencia del reloj durante el tiempo necesario para atrasarlo sin escalones.
- Tiene los problemas de escalabilidad de toda arquitectura cliente-servidor.

Algoritmo de Berkeley

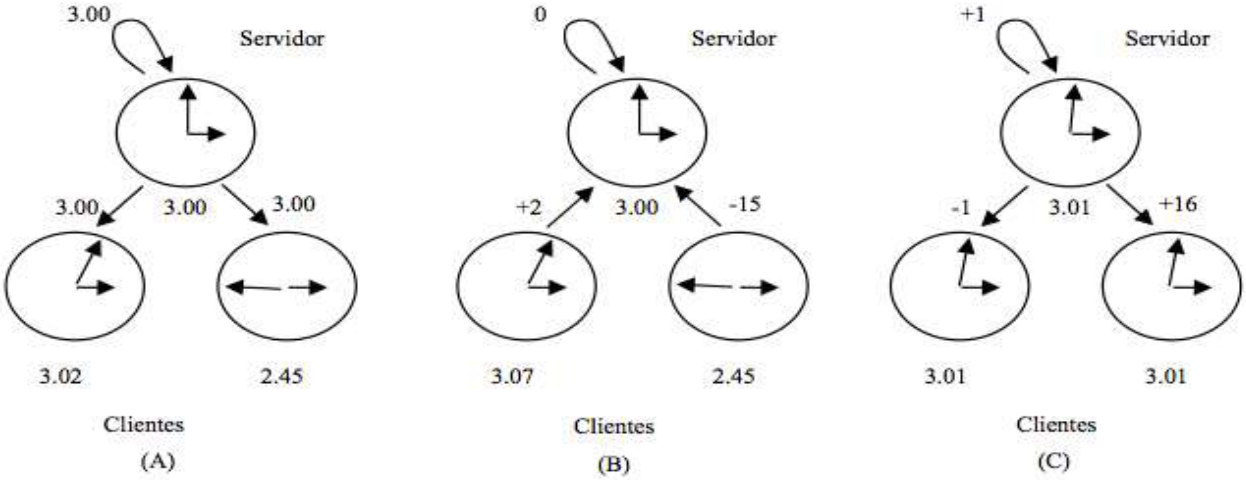


Ilustración 6 Algoritmo de Berkey

El algoritmo pasa por tres fases [12][5] y las podemos observar en la Ilustración 4:

1) El servidor (máster) es activo, pregunta a cada cliente (slave) por su hora.

2) Calcula el promedio, descontando los que están lejos del mismo.

3) Informa a cada cliente cómo debe cambiar la hora.

Estos intercambios de información se realizan a través de paquetes de datos que circulan por la red de interconexión entre las computadoras que se desea sincronizar, con las demoras correspondientes. Estas demoras deberán tenerse en cuenta a la hora de evaluar la corrección para sincronizar los relojes.

Supongamos una función derivable y continua C tal que aplicada al reloj de una computadora A nos de un valor $C_A(t)$, siendo t el tiempo Galileano Universal.

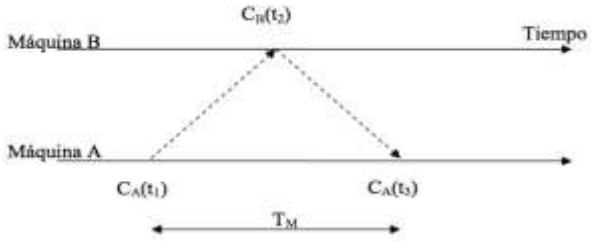


Ilustración 7 Algoritmo deBerkeley Intercambio de referencia

Intercambio de mensajes de referencias de hora entre dos máquinas, donde se puede apreciar el tiempo de ida y vuelta del mensaje.

La diferencia entre los relojes será:

$$\frac{C_A(t_1)+C_A(t_3)}{2}-C_B(t_2)$$

Ilustración 8 Diferencia entre los relojes

Ejemplo de implementación: timedde Unix (Linux)

- Se puede correr varios máster, eligen entre todos uno, pero si se caen, vuelvena elegir a través de un algoritmo de elección.
- No toma en cuenta el rtt, pero como trabaja con diferencias, no le afecta.

Algoritmos de Mills

David L. Mills a través de sus inventigaciones, realizo una recopilación los algoritmosdesincronización de relojesfísicosdentro de los sistemas distribuidos y centralizados, como lo son: El algoritmo de Lamport, de Cristian y el de Berkeley. Dichos algoritmos y el modelo matemático, fueros la base para desarrollar e implementar el protocolo NTP (Network Time Protocol). Por lo que para su desarrollo, comenzó estableciendo un modelo matemático del tiempo.

Modelo Matemático de Tiempo

T(t) mostrado por un reloj en la época t:

$$T(t)=T(t)+R(t)[t-t]+1/2D(t)(t-t)^2+x(t)$$

Donde $T(t_0)$ es el tiempo en alguna época previa t_0, $R(t_0)$ es la frecuencia y $D(t_0)$es la deriva (derivada primera de la frecuencia) por unidad de tiempo. $T(t_0)$ y $R(t_0)$ son estimados por algún proceso.

El término de segundo orden es ignorado. La naturaleza aleatoria del reloj estará caracterizada por x, en términos de frecuencia, fase o medidas de varianza.

- Mills define offset de tiempo/hora como la diferencia de hora entre dos relojes en cualquier instante.
- Como offset de frecuencia a las diferentes frecuencias de ambos relojes.

La diferencia de frecuencias, es un error sistemático.

Se puede medir, y calculando la constante de proporcionalidad f1/f2 se corrigen los relojes que pudieran construirse con dichos osciladores. Este tipo de ajuste se conoce como calibración de un reloj cuando se lo realiza con un patrón tal como un reloj atómico controlado.

A este error sistemático se le agrega la deriva de frecuencia (drift en inglés). La frecuencia del reloj puede no ser estable, debido a cambios ambientales tales como la temperatura, tensión de alimentación del oscilador, etc. En este caso, aún llevado a cabo la calibración, al cambiar la frecuencia, cambia la constante de calibración. Este driftesta especificado en partes por millón.

El primer término de la ecuación de Mills es offset de tiempo (inicial, el valor a partir del cual se actualizan ambos relojes), el segundo (a) es offset de frecuencia, el tercero (b) (1/2 D((...)) sería el debido a la deriva en las frecuencias (drift) y por último, (c) es el término aleatorio x(t) que no mide, se desprecia.

Se debe comparar el tiempo y las frecuencias a través de servidores de tiempo/hora, para proveer sincronización en la subred.

Esta comparación se logra intercambiando mensajes entre máquinas que deseen sincronizar, cada uno con 3 referencias de tiempo, tal como se ve en la figura:

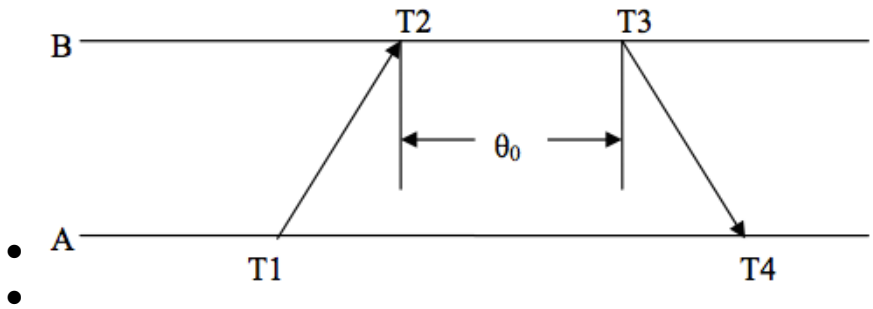


Ilustración 9 Intercambio de mensajes

- T1 hora de envío de A hacia B
- T2 hora de arribo a B
- T3 hora de envío de B hacia A

La referencia de tiempo T4 se toma al momento de llegar a la computadora A de vuelta, o sea no forma parte del mensaje.

Intercambio de mensajes entre dos máquinas

Se asume T3 > T2. Y que las demoras para enviar un mensaje desde A hasta B son iguales a enviarlo desde B hacia A (tiempos simétricos). O lo que es lo mismo, se considera pequeña la diferencia de demora introducida por la red (demora diferencial). Si a =T2−T1 y b =T3−T4, el offset del reloj θ y el tiempo ida y vuelta (round tripdelay) δ de B relativo a A en el tiempo T4 será:

$$\theta = \frac{a+b}{2}$$

$$\delta = a - b$$

Ilustración 10 Offset del reloj

Cada máquina puede calcular su offset o diferencia de hora y demora o delay. El cálculo del offset y el delay es continuo, en cada paquete. Este método es utilizado en telefonía. Tiene por ventaja que alguna pérdida de paquetes no afecta al método. El ajuste de la hora se hace para lograr un reloj monotónico, o sea que siempre avance, con lo cual en caso de estar adelantado, se baja su frecuencia hasta lograr gradualmente un ajuste.

Mills a través de éste modelo y la recopilacionde los algoritmos ya antes vistos, desarrollo el protocolo de internet NTP, el cúal presenta un modelo cliente-servidor. Éste protocolo recurre a una arquitectura de estratos, en la cual la hora suministrada por los relojes de referencia (estrato 0) es repartida entre los de estrato 1, que a su vez los distribuye entre los de estrato inferior, y así sucesivamente hasta alcanzar el estrato 16, en una arquitectura tipo árbol.

¿Qué es NTP?

El protocolo NTP, o Network Time Protocol [6][7][8] se usa para sincronizar los relojes de las computadoras a través de la red, de los Clientes instalados en los PC y en los Servidores, tomando como referencia otro Servidor o fuente de tiempo (como puede ser un receptor de satélite).

Esto provee al cliente de una exactitud en la sincronización del orden de los milisegundos en LAN y de centésimas en las WAN, relativos a un servidor primario sincronizado a la escala UTC. La escala UTC se usa en la mayoría de las naciones, y se basa en la rotación de la Tierra alrededor del Sol.

El propósito de estos receptores es estar disponible para muchos de los servicios de divulgación incluyendo el Global Position System (GPS) y otros servicios utilizados por los gobiernos de los países. Por razones de coste y conveniencia, no es posible equipar cada computador con uno de estos receptores. Sin embargo, es posible equipar un número de computadoras actuando como servidores primarios para sincronizar a la mayoría de servidores secundarios y clientes conectados por una red común. Para hacer esto, es necesario un protocolo de red de sincronización de tiempo, que pueda leer un servidor de hora, transmita la lectura a uno o mas clientes y ajuste el reloj de cada cliente como sea necesario.

El principal factor que contribuye a ofrecer una sincronización segura y precisa del tiempo, es la selección de las rutas de acceso y los servidores que serán usados en el archivo de configuración. El soporte que NTP ofrece a uno o varios ordenadores es habitualmente diseñado a partir del NTP de una subred ya existente, que consiste en una jerarquía o stratum redundante de servidores y clientes donde cada nivel está identificado por un número de stratum.

Algunas configuraciones de NTP incluyen autentificación criptográfica para preservar a este protocolo de ataques malintencionados o accidentales.

Antes de la llegada de NTP, habían sido desarrollados otros dos protocolos: The Time Protocoly, TheDayTimeProtocol; estos permitían la sincronización automática como el NTP, pero por el contrario no compensaban de ninguna forma el retraso en la transmisión entre un cliente y un servidor.

El protocolo de sincronización determina la diferencia entre la hora del reloj del servidor en relación a la hora del cliente. Bajo petición, el servidor envía un mensaje incluyendo el valor de la hora en ese momento (timestamp), y el cliente almacena esto en su timestamp. Para mayor precisión, el cliente necesita medir el retraso de la propagación desde el servidor al cliente para determinar su offset relativo al servidor. Pero como no es posible determinar el retraso de un camino, a menos que el offset actual sea conocido, el protocolo mide el retraso total en dar toda la vuelta y asume que los tiempos de propagación son estáticamente iguales en cualquier dirección.

Una razón de porque NTP sobresale es su habilidad para sincronizar los relojes de sistema de servidores primarios (Stratum 1) al Tiempo Universal Coordinado (Universal Coordinated Time, UTC) por vía radio, satélite, modem o relojes atómicos locales y entonces proporcionar tiempo para los Stratum 2, servidores bajos y clientes que carecen de hardware especial pero en cambio se sincronizan a la Internet usando su propia copia del programa de NTP.

Esquema de sincronización

Un servidor NTP primario, o Stratum 1, ésta conectado a un reloj de referencia de alta precisión. Esta referencia puede ser, por ejemplo, un reloj atómico, o un receptor de radio o GPS. Además, este servidor cuenta con software para manejar el protocolo NTP [17][18].

Otras computadoras, que funcionan como servidores Stratum2, utilizan un software similar (usualmente el mismo), y consultan automáticamente al servidor primario para sincronizar su reloj. A su vez, éstos pueden sincronizar a otros servidores, que en este caso serán Stratum 3, y asípodría seguirse hasta 16 niveles. La arquitectura también soporta que un cliente haga sus consultas a mas de un servidor y puede haber comunicaciones entre servidores de un mismo stratum.

En la Ilustración 11 puede verse un esquema de esta estructura.

Cuanto más alejado esté una computadora del reloj de referencia, o sea, cuanto más alto sea su Stratum, menos precisa será la sincronización. Sin embargo, cualquier Stratum siempre será suficiente para que el reloj no se aleje más de unos milisegundos de la hora real.

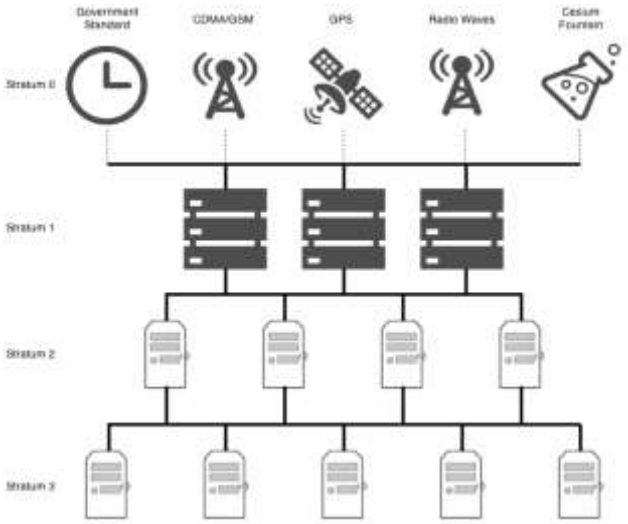


Ilustración 11 Jerarquía en estratos NTP

Hasta ahora definimos que una maquina, que llamaremos cliente, puede sincronizarse con otra o con alguna referencia externa, y también puede comportarse como servidor, y utilizarse para sincronizar otras.

Siempre que haya una asociación entre dos maquinas, donde una se comporte como cliente, y otra como servidor, al cliente le corresponderá el Stratuminmediatamente superior al del servidor. Hay otra posibilidad, donde dos o más maquinas se configuran para comportarse entre sí como clientes o servidores, según quién este´ más cerca de un reloj de referencia, o quien sea más confiable de acuerdo con el algoritmo que rige la sincronización por NTP. En este tipo de asociaciones, los servidores se llaman Peers.

Protocolo NTP

El protocolo NTP puede trabajar en uno o mas modos de trabajo, uno de ellos es el modo cliente/servidor, también llamado maestro/esclavo. En este modo, un cliente se sincroniza con un servidor igual que en el modo RPC convencional.

NTP también soporta un modo simétrico, el cual permite a cada uno de los dos servidores sincronizarse con otro, para proporcionarse copias de seguridad mutuamente.

NTP también soporta el modo broadcast por el cual muchos clientes pueden sincronizarse con uno o varios servidores, reduciendo el tráfico en la red cuando están involucrados un gran número de clientes.

En NTP, el multicast IP también puede ser usado cuando la subred se abarca múltiples redes de trabajo.

La configuración puede ser un serio problema en grandes subredes. Varios esquemas están en bases de datos públicas y servicios de directorios en red que son usados para descubrir servidores. NTP usa el modo broadcast para soportar grandes cantidades de clientes pero para los clientes que solo escuchan es difícil calibrar el retraso y la precisión puede sufrir. En NTP, los clientes determinan el retraso a la vez que buscan un servidor en modo cliente/servidor y luego cambian a modo solo escucha. Además, los clientes NTP pueden hacer un broadcast de un mensaje especial para solicitar respuestas de servidores cercanos y continuar en modo cliente/servidor con los que le respondan.

Aquí tenemos un esquema de las diferentes configuraciones [9]:

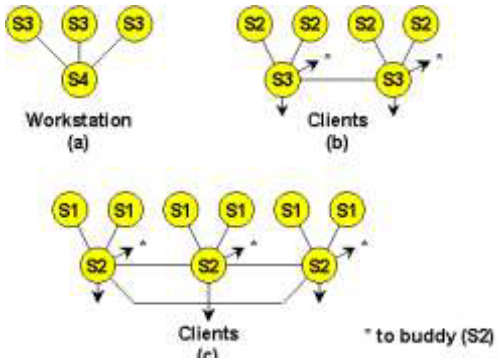


Ilustración 12 Esquema de configuraciones

a) Las workstation usan el modo multicast con múltiples servidores de departamento

b) Los servidores de departamento usan modos cliente/servidor con múltiples servidores secundarios (nivel superior en la subred) y modos simétricos los unos con los otros.

c) Los servidores secundarios usan modos cliente/servidor con más de seis servidores primarios externos, modos simétricos con los otros y un servidor NTP secundario externo.

Con esto lo que podemos ver es que los servidores de un cierto nivel no solo se comunican con un servidor de capa superior sino también con servidores de su misma capa.

Arquitectura NTP

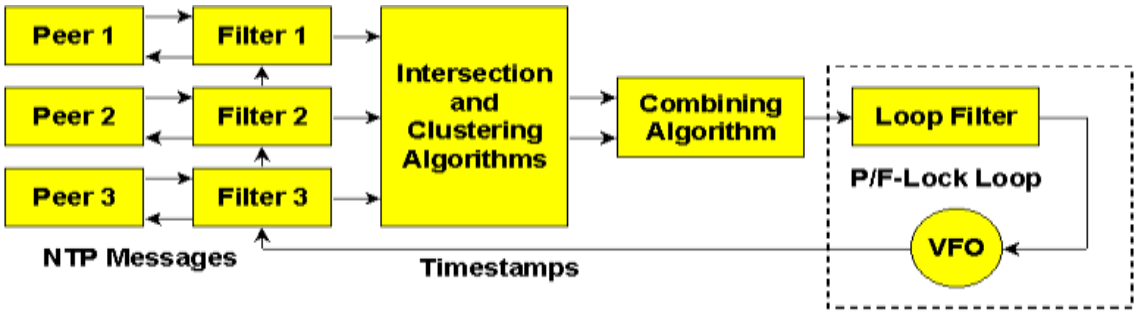


Ilustración 13 Arquitectura NTP

En la imagen anterior podemos ver los servidores (Peer) que proporcionan redundancia y diversidad.

El algoritmo de filtrado usado en NTP se basa en la observación de que el intervalo correcto depende del retraso del viaje de ida y vuelta del mensaje. El algoritmo va acumulado muestras de offsets/retrasos en una ventana y selecciona las muestras de offset que se corresponden con el mínimo retraso.

Los algoritmos de intersección y agrupamientoselecciona el mejor subconjunto de la población de referencias de hora de los mensajes. Primero, los ordena por estrato y luego, por distancia de sincronización. Este algoritmo se encarga de tomar de los ejemplos los verdaderos, descartando los falsos de los mensajes con que se cuente. La condición de verdadero o falso está determinada por estar dentro de intervalos de confianza.

Los algoritmos de combinación computan la media de los offsets de tiempo, para tomar en cuenta el reloj seleccionado y su demora para dar la referencia a la cual se va a ajustar.

Los datos buenos se combinan para obtener una estimación única de la hora a través de un promedio.

El filtro de fase y la variación de la frecuencia del oscilador (VFO) realimenta los filtros de reloj para evitar fluctuaciones, y mediante el resultado de las fases NTP y VFO se calcula el registro de desplazamiento del reloj. Se calcula la fase y la frecuencia por funciones de predicción a través de un lazo enganchado en fase, que provee una forma de ajustar el reloj local en hora y frecuencia en respuesta a las correcciones del protocolo de sincronización.

Una vez calculado VFO, el registro de desplazamiento se ajusta el reloj que controla la frecuencia de oscilador local, llevando así Timestamps (marcas de tiempo) a los filtros de reloj.

Cada proceso de servidor corre independiente en intervalos de muestreo determinado por el proceso del sistema y por servidores remotos.Cada proceso del sistema corre en intervalos de muestreo determinados por la medida de la fase del jitter de red y de la estabilidad de la frecuencia del oscilador del reloj local.

El proceso de ajuste del reloj corre en 1-s intervalos para disciplinar la fase y frecuencia delas variaciones de la frecuencia de oscilación.

Si hacemos un análisis del flujo de datos del NTP, obtenemos lo siguiente:

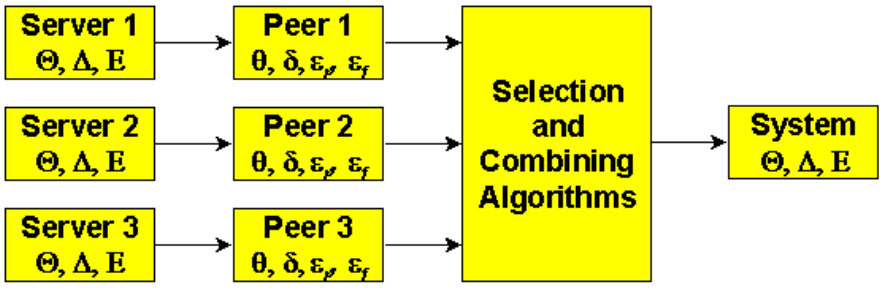


Ilustración 14 Flujo de datos NTP

Cada servidor remoto calcula las variables propias del servidor Θ =offset, Δ=retraso y E=dispersión relativas a la raíz de la sincronización del subárbol.

En cada mensaje NTP de llegada, el peer actualiza el offset θ, retraso δ, la dispersión de la fase ε_ry la dispersión de la frecuencia ε_f propios a partir de las marcas de tiempo (timestamps) y los algoritmos de filtrado de reloj.

En los intervalos de muestreo del sistema, los algoritmos de selección y combinación del reloj actualizan las variables, Θ, Δ y E del sistema.

Las dispersiones de la frecuencia ε_f y la E se incrementan con el tiempo en una tasa que depende de la tolerancia especifica de la frecuencia según el algoritmo de filtrado del reloj.

Algoritmo de filtrado de reloj

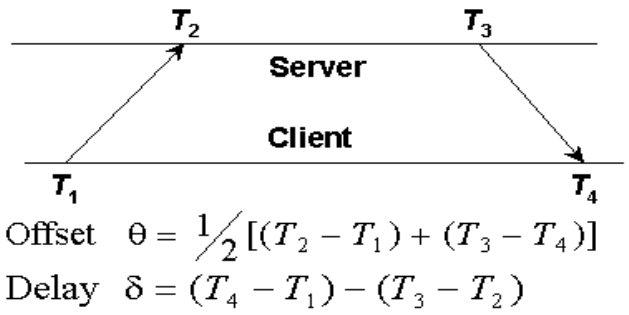


Ilustración 15 Algoritmo de filtrado de reloj

La mayor precisión del offset θ es medida con el menor retraso δ.

La dispersión de la fase ε_r es la media de las diferencias de offset sobre las ultimas muestras usado como un estimador del error.

En una implementación muestras de (δ_i, θ_i) son desplazadas en una ventana de registros de desplazamiento desde la primera a la última, causando que viejas muestras sean desplazadas por las otras. Entonces, las muestras son puestas en una lista temporal ordenadas en orden por el aumento de δ.

La primera muestra en la lista (δ_0, θ_0) representa los estimadores (δ ^ ,θ ^), los cuales son almacenados para cada peer separadamente para después procesarlos por los algoritmos de combinación y selección.

La dispersión filtrada es interpretada como un indicador de cualidad.

Un buen estimado r es el peso de las diferencias de θ_i en una lista ordenada temporalmente relativa a θ_0.

Asumimos que la lista tiene $n > 1$ entradas$(n = 8$ normalmente$)$ con(δ_j, θ_j) $(j = 0, 1, \ldots, n-1)$muestras en orden incremental deδ_i. El filtro de la dispersión ε es definido como:

$$\varepsilon = \sum_{j=0}^{n-1} |\theta_j - \theta_0| \, v_j ,$$

La dispersión de la frecuencia ε_fRepresenta la lectura del reloj y la frecuencia de tolerancia a errores usado en métrica de distancia.

La distancia de sincronización $\lambda = \varepsilon f + \delta/2$ Usada como distancia métrica y máximo limite de error, hasta que la hora correcta θ_0 este en el rango:

$\theta - \lambda \leq \theta 0 \leq \theta + \lambda$

Este algoritmo reduce en el error estándar.

Algoritmo de intersección

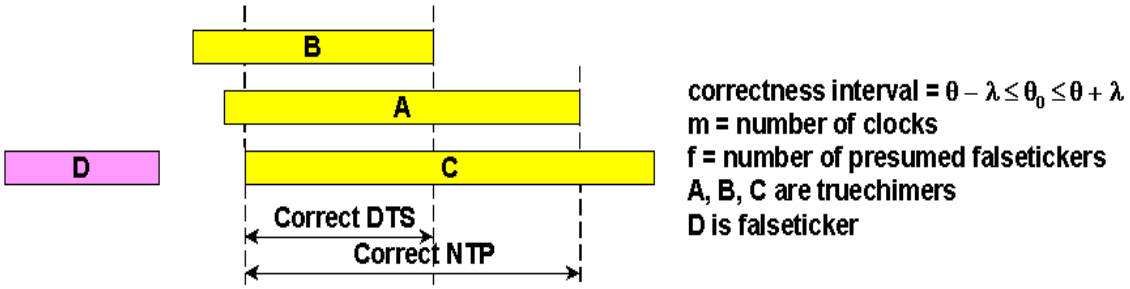


Ilustración 16 Algoritmo de intersección

El algoritmo NTS (de intersección) requiere que el punto medio del intervalo este en la intersección.

Inicialmente, pone a 0 f (falsotick), los contadores c y d.

Lee del mas lejano de la izquierda: añade uno a c para cada final de punto bajo, y le resta uno para cada final de punto alto, añade uno a d para cada punto medio

Si $c \geq m-f$ y $d \geq m-f$, declara éxito y sale del procedimiento.

Hace lo mismo empezando por la parte más lejana de la derecha, si no hay éxito, se incrementa f en uno y se intenta de nuevo:

Si $f^3 m/2$,Error

Algoritmo de agrupamiento

El algoritmo de agrupamiento nos va a servir para seleccionar el mejor subconjunto de servidores y combinar después sus diferencias para determinar el ajuste del reloj. Sin embargo, los diferentes servidores muestran diferentes diferencias sistemáticas, así que la mejor estadística no es obvia.

Varias clases de algoritmos de agrupamiento han sido encontrados en para este propósito. El que usa NTP se basa en ordenar las diferencia por una cualidad métrica, entonces calcula la varianza de todos los servidores en relación a cada servidor separadamente.

El algoritmo repetidamente deprecia al que tiene la más amplia varianza hasta que permanezcan el mínimo número de servidores. El ajuste final del reloj se computa como una media de los supervivientes.

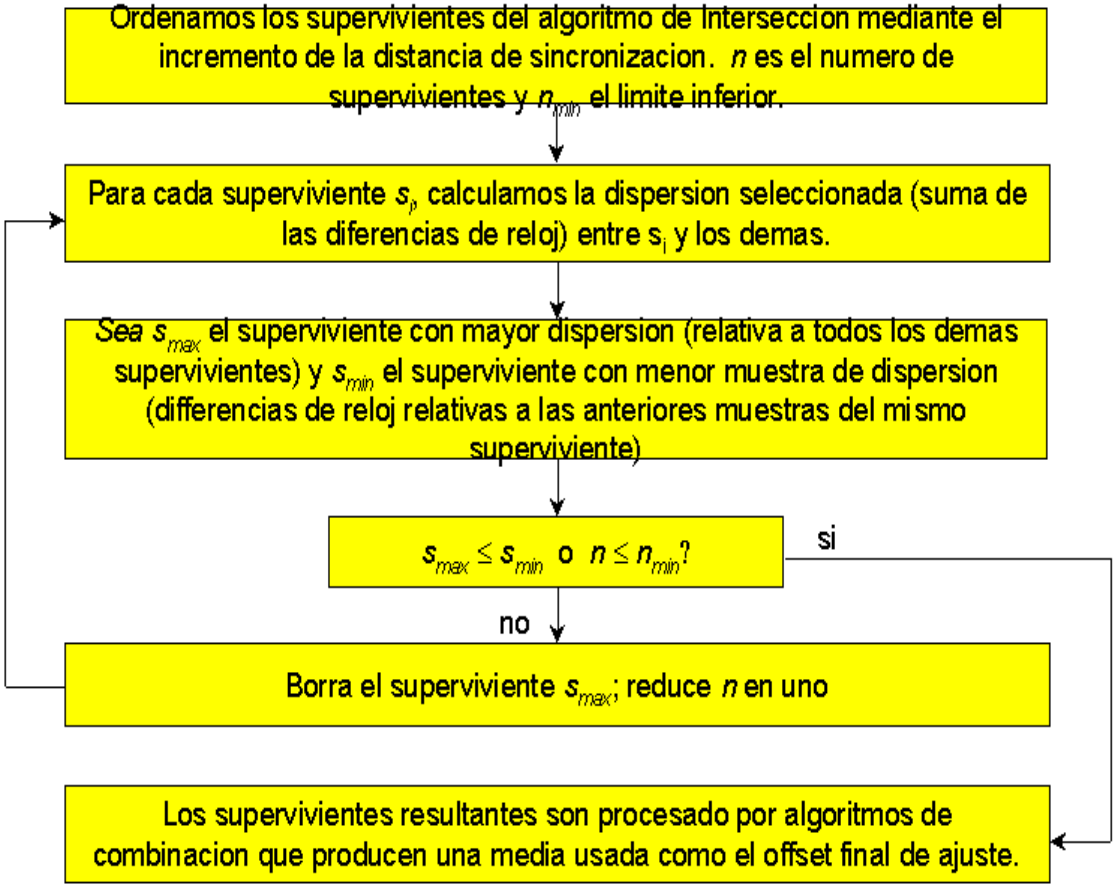


Ilustración 17 Algoritmo de agrupamiento

Algoritmos de selección y combinación

El algoritmo NTP de selección, esta basado en principios de la estadística y observaciones pragmáticas donde la mayor fiabilidad esta normalmente asociada con el stratummas bajo y dispersión de sincronización, mientras que la mayor precisión estaasociada a con el stratummas bajo y distancia de sincronización.

El algoritmo de selección de peer comienza construyendo una lista de candidatos peers ordenados primero por el stratum y después por la dispersión de sincronización. Para ser incluido en la lista de candidatos un peer debe pasar varios test diseñados para detectar errores descarados e implementaciones defectuosas.

Si ningún peer pasa los test la fuente de sincronizaciones existente, se cancela y el reloj local corre a su frecuencia intrínseca.

La lista puede ser recortada desde el final hasta un límite máximo y un stratum máximo.

El siguiente paso es diseñar el detector de falsetickers u otras condiciones que pudieran dar errores graves.

La lista de candidatos es reordenada primero por stratums y después por distancia de sincronización.

Sea m>0 el numero de candidatos que permanecen en la lista y sea$\theta_j$el offset de eljth candidato. Para cada j $(0 \le j < m)$ la dispersión seleccionada ε_j relativa al candidato jse define como:

$$\varepsilon_j = \sum_{k=0}^{m-1} |\theta_j - \theta_k| \, w^k$$

Donde w es un factor experimental ajustado para las características deseadas. Entonces descartamos el candidato con máxima ε_jo, en caso de empate el de máximo j, y repetimos el procedimiento. El procedimiento termina cuando la máxima dispersión seleccionada sobre todos los candidatos que quedan es menor que el mínimo filtro de dispersión de cada candidato o hasta que quede un solo candidato.

Los procedimientos están diseñados para favorecer aquellos peers que están cerca del principio de la lista, que son los de menor stratum y el más bajo retraso, y presumiblemente pueden proporcionar la hora con más precisión.

Con la apropiada selección de los factores de peso w, los alejados serán descartados del final de la lista, a menos que otras entradas tengan una discordancia significativa con respecto a las entradas que permanecen en la lista, en cuyo caso la entrada es descartada.

Los offsets de los peers que permanecen en la lista de candidatos son estáticamente equivalentes, así que cualquiera de ellos puede ser elegido para ajustar el reloj local.

Algunas implementaciones combinan esto usando un algoritmo de media, en el cual los offsets de los peers que permanecen en la lista son pesados por un estimador de error para producir una estimación combinada En esas implementaciones de estimación del error es tomada para ser la recíproca de la dispersión de sincronización

Para actualizar este procedimiento también ponemos el stratum local a uno mayor que el peer seleccionado. Además la distancia de sincronización del servidor y la suma de las dispersiones totales a la raíz de la subred de sincronización son calculadas y almacenadas en una variable de estado del sistema. Todo esto ese incluye en la cabecera del mensaje NTP.

Algoritmo de filtro de fase

En el corazón de los protocolos de sincronización esta el algoritmo usado para ajustar el reloj del sistema en concordancia con el ajuste final determinado por el resto de algoritmos.

Esto es llamado algoritmo defiltro de fase o disciplina del reloj. Este algoritmo puede ser clasificado dependiendo de si minimiza el tiempo de offset, la frecuencia de offset o ambos.

Por ejemplo, NTP minimiza ambos, tanto el offset de tiempo como el de frecuencia por lo que es más complicado y menos indulgente en cuanto a los errores de diseño y de implementación.

Todos los algoritmos de disciplina del reloj funcionan como un bucle realimentado, con offsets medidos usados para ajustar la fase y la frecuencia del oscilador. El comportamiento del bucle realimentado es bien comprendido y modelado mediante análisis matemático.

Los parámetros de diseño significativos son la constante de tiempo, o la sensibilidad a las variaciones externa o internas variaciones tanto en el tiempo como en la frecuencia. Una óptima selección de la constante de tiempo depende del intervalo entre los mensajes de actualización.

En general, cuanto mayor sea la longitud de estos intervalos, más amplia será la constante de tiempo y viceversa.

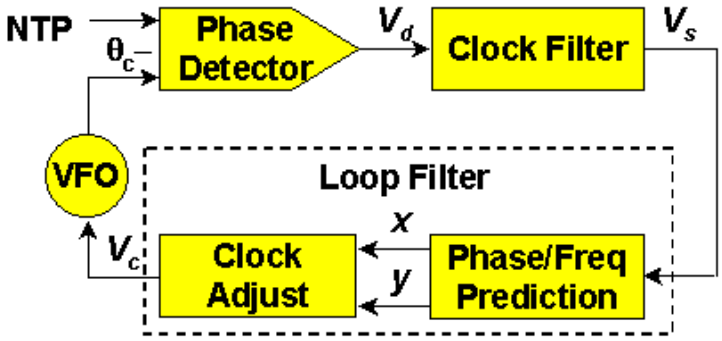


Ilustración 18 Algoritmo de filtro de fase de reloj

$V_d$es una función de las diferencias de fase entre NTP y el VFO.

V_S depende de la elegida en el registro de desplazamiento del filtro del reloj.

x e y son la fase y la frecuencia actualizadas, respectivamente, computadas por funciones de predicción

El proceso de ajuste del reloj corre una vez por segundo para computar V_C, que controla La frecuencia del oscilador de reloj local

La fase de VFO se compara a la fase NTP cerrando el bucle de realimentación.

Funciones de predicción FLL/PLL

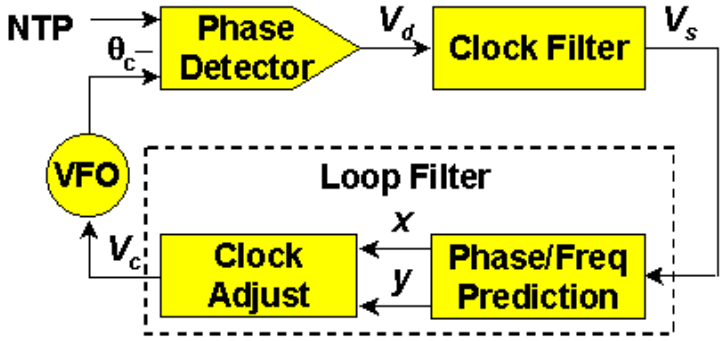


Ilustración 19 Funciones de predicción FLL/PLL

V_S es el offset de la fase producido por el algoritmo de filtrado del reloj.

X es la corrección de la fase calculada como una fracción de V_S yFLL es el ajuste de la frecuencia calculado como la media de los offsets de las frecuencias pasadas.

PLL es el ajuste de frecuencia calculado como la integral de los offsets de la fase pasada.

FLL y PLL son combinados de acuerdo a los factores de peso determinados por el intervalo de muestreo y la desviación de Allan.

Desarrollo del Sistema

Visualización y descripción general del sistema

El sistema cuenta con tres nodos:

1. Computadora portátil
2. Computadora portátil
3. Computadora portátil

La conexión de estos nodos se llevara acabo por medio de un dispositivo de red:

1. Switch

El sistema propuesto es el siguiente:

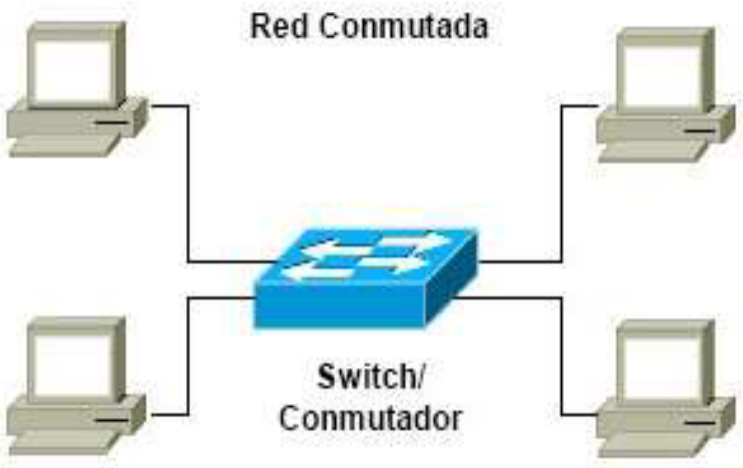


Ilustración 20 Sistema propuesto

Descripción de los equipos

Sistema de cómputo 1 Coordinador

- Laptop Hp dv5-2232la
- Procesador AMD Turion
- Disco duro de 500 GB
- 4 GB en memoria RAM
- Sistema Operativo Linux 14.04

Sistema de cómputo 2 Esclavo

- Laptop Acer Aspire AS5738
- Procesador Intel CoreDuo
- Disco duro de 350 GB
- 4 GB en memoria RAM
- Sistema Operativo Linux 14.04

Sistema de cómputo 3 Esclavo

- Laptop Gateway
- Procesador Intel B830
- Disco duro de 320 GB
- 4 GB en memoria RAM
- Sistema Operativo Linux 14.04

Dispositivo de red:

- Switch Kingston de 5 puertos

En el sistema hay tres equipos de cómputo que representan los tres nodos de información, todos ellos, laptops que tienen instalado como sistema operativo Ubuntu Linux 14.04 y en el cual la conexión entre los nodos del sistema se realiza a través de un Switch físicamente por medio de cable UTP categoría 5.

Características de la red

- Tipo de cable: UTP
- Categoría: 5e
- Tipo de red: Alámbrica
- Código de colores: Tipo B
- Topología de red: Estrella

Cable UTP por sus siglas en ingles (UnshieldedTwistedPair) es un medio de comunicación físico que consta de 8 hilos que están trenzados por pares y asilados por plástico, juntos el aislamiento y el trenzado disminuyen las interferencias de comunicación que pudieran existir.

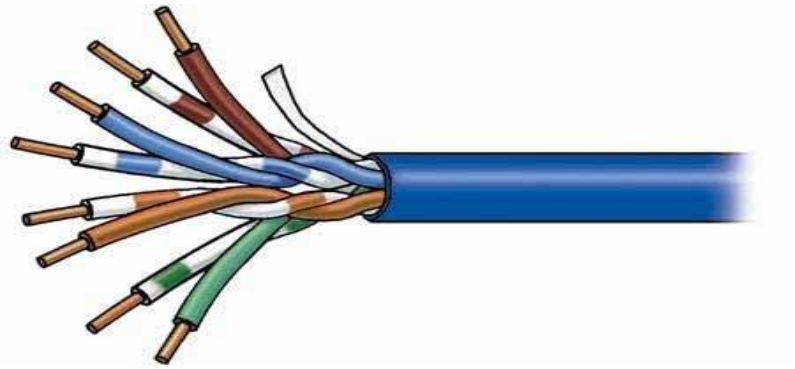


Ilustración 21 Cable UTP

Categoría 5e: permite procesar señales de alta integridad y alcanzar velocidades de 100bps hasta 100Mhz.

Código de colores B: Establece que la conexión haga un recorrido del Pin 1 al Pin 8 con el siguiente orden: Blanco/Naranja, Naranja, Blanco/Verde, Azul, Blanco/Azul, Verde, Blanco/Café, Café, y el otro extremo del cable será la conexión pin a pin debido a que la interconexión se esta haciendo con un switch y no equipo a equipo.

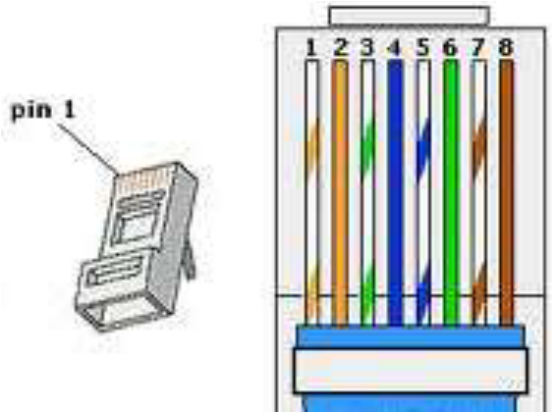


Ilustración 22 Código de colores B

Topología en Estrella: se define como tal debido a que el diseño de la red necesario y más conveniente para lo que se desea hacer demanda un nodo en medio representado por un switch que comunicará al resto de los nodos o equipos en la periferia.

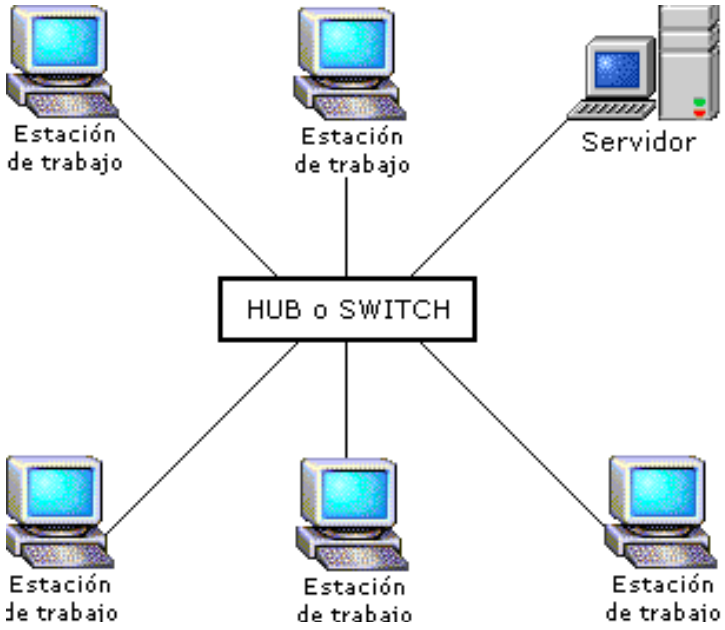


Ilustración 23 Topología en Estrella

Instalación de Software

Cabe mencionar que la realización de instalación de software se realizara con el acceso de usuario "root", es decir, posee todos los derechos en todos los modos y permite activar las funciones de los súper usuarios.

Por lo que para realizar las instalaciones de manera satisfactoria y sin ningún problema de autorización de usuario, abriremos una terminal en el sistema operativo Ubuntu Linux y ejecutaremos el siguiente comando:

EquipoDeTrabajo@equipo:~$**sudo -i**

Enseguida nos pedirá la contraseña y posteriormente estaremos como usuario "root"

EquipoDeTrabajo@equipo:~$**sudo -i**

Password:

root@equipo:~#

Instalar NTP

Network Time Protocol (NTP) es un protocolo de Internet para sincronizar los relojes de los sistemas informáticos a través de ruteo de paquetes en redes con latencia variable. NTP utiliza UDP como su capa de transporte, usando el puerto 123.

Para instalarlo podemos usar el Gestor de paquetes Synaptic o bien instalarlo a través de la consola [13], ej:

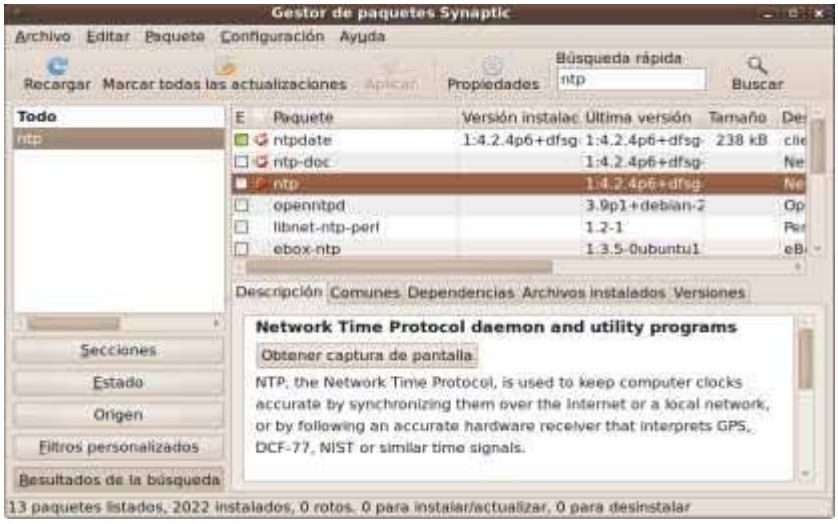


Ilustración 24 Gestor de paquetes Synaptic

Directamente se puede escribir en la consola:

root@equipo:~# apt-getinstallntp

Podemos comprobar que el servidor ya esta en marcha de esta manera:

root@equipo:~# pgrepntpd

Si nos devuelve un PID significará que toda la instalación ha ido bien.

Una vez instalado el servidor NTP [14] ya podemos configurarlo mediante el archivo **/etc/ntp.conf** en donde básicamente deberemos indicar los servidores con los que nuestro servidor NTP se va a sincronizar.

Se recomienda que se pongan al menos 2 servidores remotos con los que pueda sincronizarse. Uno actuará como servidor primario y el otro como copia de respaldo.

Antes de empezar a editar el archivo de configuración del NTP, debemos sincronizar la hora con el servidor externo que utilizaremos para sincronizar, éste paso se hace para garantizar que nuestro servidor tenga la hora exacta. Esta sincronización inicial la vamos hacer con el siguiente comando:

root@equipo:~# ntpdate cronos.cenam.mx

Nota**:** en ocasiones el bash no nos deja ejecutar este comando, para esto debemos instalarlo asi:

root@equipo:~# apt-getinstallntpdate

Cronos.cenam.mx es el nombre de uno de los servidores externos de tiempo con el cual estaremos sincronizados, (se configuran varios servidores externos por si alguno falla) como también podemos ver en el comando hay una letra que es la "-u" esta se utiliza cuando se tiene inconvenientes con el firewall.

También podemos observar cual es el comportamiento de la ejecución del comando, en la cual se puede ver que corrige la fecha y la hora, también la dirección del servidor y el tiempo que tarda en actualizar

Ilustración 25 Configuraciones NTP ntpdate

Ahora lo que vamos hacer es editar el archivo de configuración de nuestro servidor NTP. Este archivo lo vamos a editar con el siguiente comando:

root@equipo:~# gedit /etc/ntp.conf

Ilustración 26 Configuraciones NTP /etc/ntp.conf

A continuación nos aparecerá una ventana con unos parámetros que están configurados por defecto. Nosotros vamos a mostrar por pantallazos el archivo de configuración y subrayaremos lo que vamos a modificar.

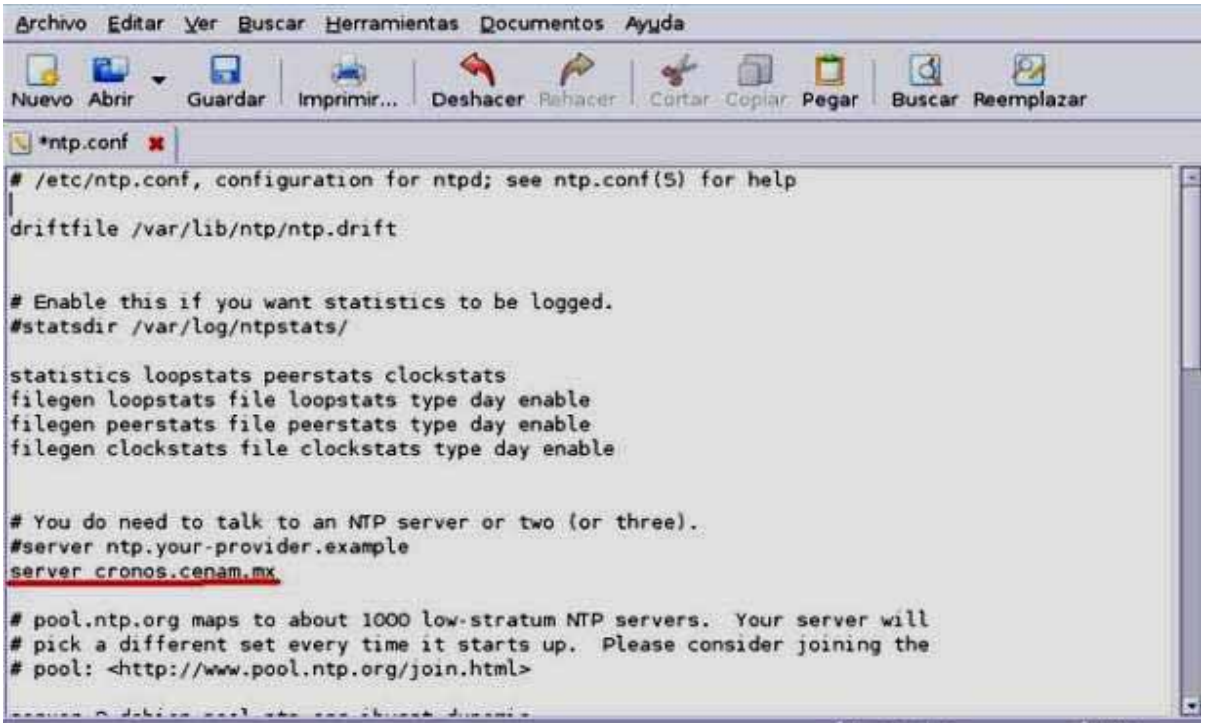


Ilustración 27 Servidor Cronos

En el primer pantallazo encontramos una linea subrayada de color rojo en esta línea estamos indicando que nuestro servidor NTP se sincronizara con el servidor externo [18],**cronos.cenam.mx**

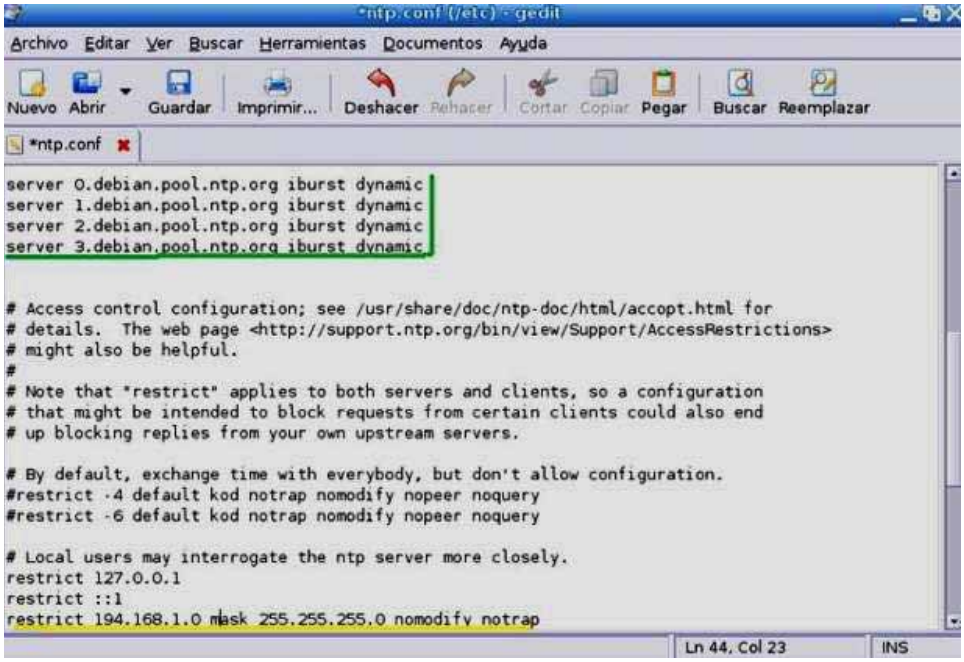


Ilustración 28 Otros Servidores NTP

En la parte de arriba encontramos subrayado con color verde otros servidores que trae configurado en archivo por default y en este ejemplo lo dejaremos también configurado con estos servidores de tiempo.

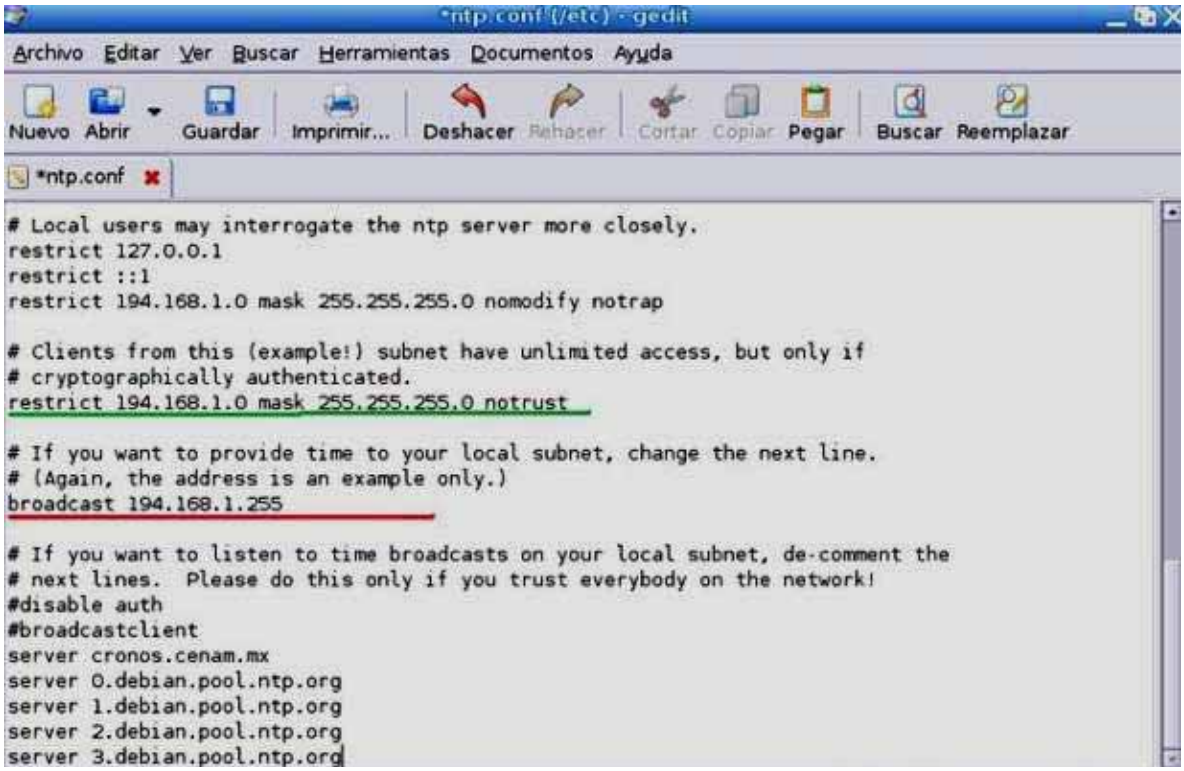


Ilustración 29 Host de la red

Y en la parte de abajo con color amarillo encontramos una línea que permite que todos los host de la red puedan sincronizarse con el servidor NTP que va hacer el equipo al cual le estamos configurando este archivo.

La línea marcada de color verde le indica a los host de la red que cuando vayan a tener acceso al servidor de tiempo lo hagan con una autentificación y la línea con color rojo se coloca la dirección de broadcast de nuestra red.

Ahora el paso a seguir es iniciar el demonio del servidor NTP, este lo haremos con el siguiente comando:

root@equipo:~# /etc/init.d/ntprestart

Ilustración 30 Inicio de demonio del servidor NTP

Con la ejecución de este ultimo comando terminamos de configurar nuestro servidor de tiempo NTP ahora lo que procedemos hacer es a configurar nuestro cliente NTP.

Como instalar y configurar un cliente NTP

Ahora vamos a instalar y configurar un cliente NTP [14] para nuestro servidor. La distribución que utilizaremos para configurar el cliente es Ubuntu.

Lo primero que vamos a hacer es probar si nuestro servidor quedo bien configurado y vamos a sincronizar la hora con el servidor interno que acabamos de configurar. Esta sincronización inicial la vamos hacer con el siguiente comando:

root@equipo:~#ntpdate –u 194.168.1.4

194.168.1.4 es la dirección IP del servidor NTP que acabamos de configurar y que desde ahora es el que nos sincronizara la hora mientras que el se sincroniza con uno externo.

Como podemos observar la hora y la fecha se sincronizan con la que tiene el servidor NTP.

Ilustración 31 Prueba del Servidor NTP

Luego procedemos a instalar el NTP con el comando:

root@equipo:~#apt-getinstallntp

Ilustración 32 Instalación NTP cliente

Como podemos ver es el mismo software que utilizamos para configurar en servidor NTP, es que este software nos sirve tanto para configurarlo como servidor o cliente.

Ahora lo que vamos hacer es editar el archivo de configuración de nuestro cliente NTP. Este archivo lo vamos a editar con el siguiente comando:

root@equipo:~#gedit /etc/ntp.conf

Ilustración 33Solicituda del Servidor NTP

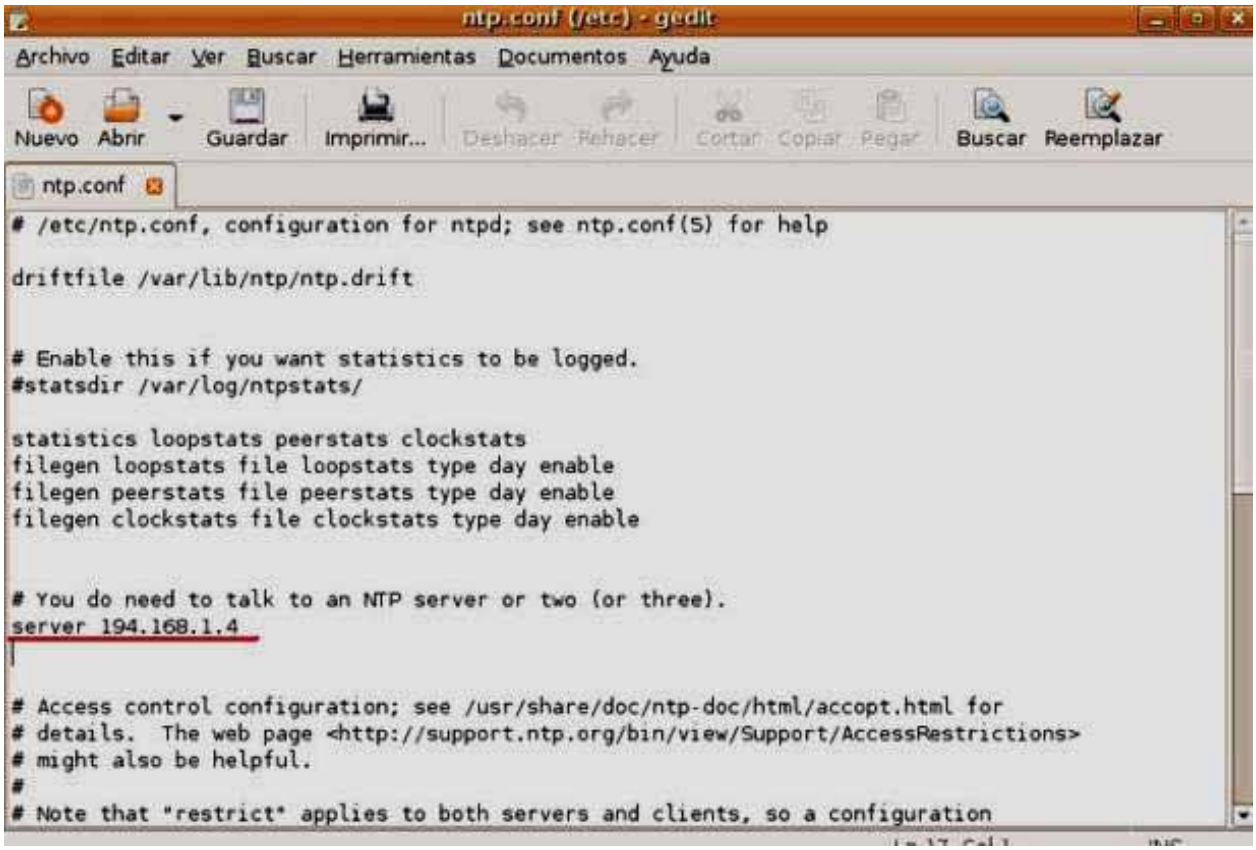


Ilustración 34 Configuración NTP /etc/ntp.conf

A continuación nos aparecerá una ventana con unos parámetros que están configurados por defecto. Nosotros vamos a mostrar lo único que hay que configura en el archivo de configuración del NTP.

Podemos ver en la imagen subrayado con rojo hay una linea donde se especifica el servidor NTP con el cual este cliente quiere sincronizar la hora y la fecha.

Ahora solo queda reiniciar nuestro demonio y esto lo hacemos con el siguiente comando:

root@equipo:~#/etc/init.d/ntpstart

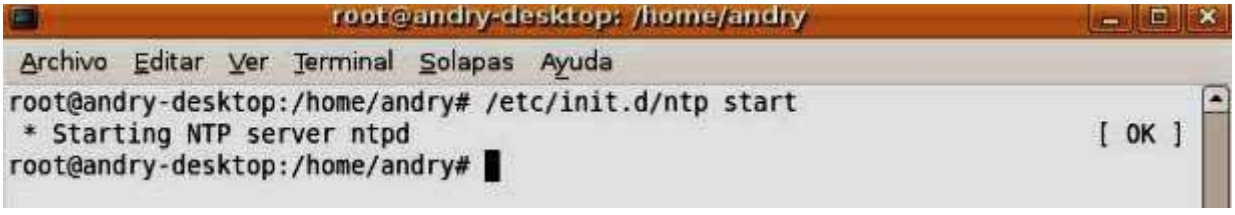


Ilustración 35 Reinicio del Servicio

Ahora ya esta nuestro cliente sincronizado con nuestro servidor local. Una forma de de saber que este enlace esta bien hecho es mediante la ejecución del comando:

root@equipo:~#ntpdate 194.168.1.4

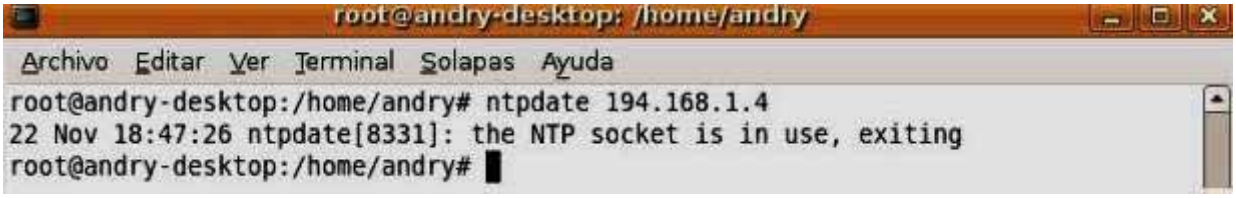


Ilustración 36 Prueba ntpdate

Como podemos ver en la imagen hay un error y dice que este enlace ya se está usando, lo que significa que nuestra conexión con el servidor NTP esta bien hecha y que cuando intentamos utilizar esta conexión de nuevo, nos arroja el mensaje que vemos en la imagen.

Pero si en cambio ejecutamos el comando con "–u" ahí si nos sincroniza por que como explique anteriormente este comando tiene privilegios frente al firewall.

Para comprobar que estamos sincronizando correctamente podemos usar el comando: ntpq y peer para poder ver los servidores [15].

root@equipo:~# ntpq

ntpq> peer

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 LOCAL(0)        LOCAL(0)        10 l    2   64  377    0.000    0.000   0.008
-ntp1-rz.rrze.un .DCFp.           1 u  446  512  377  110.762   -9.304   3.525
-ntp2-rz.rrze.un .GPS.            1 u  242  512  377  112.863   -9.615   0.526
-hora.cs.tu-berl .PPS.            1 u  456  512  377  123.149   -2.476   0.479
*canon.inria.fr  .GPS.            1 u  444  512  377  100.236    1.824   0.455
-ntp2.ien.it     .IEN.            1 u  313  512  377  141.008   -3.922   0.386
-swisstime.ee.et .DCFa.           1 u  445  512  377  105.025   -5.507   4.360
+ntp0.NL.net     .GPS.            1 u  495  512  377  267.048    3.765   0.254
+ntp2.gbg.netnod .PPS.            1 u  314  512  377  145.764    4.568   1.417
```

Ilustración 37 Lista de Servidores disponibles

En este caso, en el momento de hacer la llamada a **ntpq** el servidor elegido era **canon.inria.fr** lo que se indica con un * al principio de la línea. Los que comienzan con + también son candidatos. Si comienzan con – se

considera que el error es demasiado alto. Si comienza con un blanco entonces no esta sincronizado o no es accesible.

También se indica en el listado el stratus de cada peer, si está activo (t), cuántos segundos hace que se llamó (when), cuantos segundos deben pasar entre cada llamada (poll), la máscara de llamadas con exito (en octal, cada bit corresponde a una llamada, el bit 0 es la última y el bit 7 la de hace 7 llamadas, un '1' corresponde a éxito). (delay) se refiere al tiempo estimado, en milisegundos, que tarda el paquete desde el servidor hasta nuestro host. (offset) es la diferencia estimada entre lo que marca nuestro reloj y el de referencia. (jitter) se refiere a la dispersion de los valores de referencia obtenidos con ese peer, es una medida de la calidad de la referencia.

Por último, algunas recetas para monitorizar qué servidores, cuantos y con qué frecuencia están utilizando nuestro servidor [15]. De ello se encarga ntpdc. Si se quiere una lista detallada:

```
root@equipo:~# ntpdc -c monlist
```

Puede tardar un poco porque DNS tiene que resolver todas las entradas. Si queremos una respuesta rápida pero sin los nombre de lugar: ntpdc -n -c monlist, podemos encadenar esto para saber cuántos clientes/servidores hemos contactado.

```
root@equipo:~# ntpdc -ncmonlist | wc–l
```

Comprobar que el demonio NTP se está ejecutando

El protocolo de tiempo de red (NTP) demonio debe estar en ejecución en todos los hosts del clúster para asegurarse de que sus relojes están sincronizados. El demonio de propagación se basa en todos los nodos en los que tienen sus relojes sincronizados con fines de sincronización. Si los nodos no tienen NTP en marcha, la instalación puede fallar con un error de configuración propagación, así como otros errores posibles.

Para comprobar si los hosts están configurados para ejecutar el demonio NTP en el arranque, ejecute el siguiente comando:

```
root@equipo:~# chkconfig –listntpd
```

La salida indica los niveles de ejecución en el que el demonio ejecuta. Verifique que el nivel de ejecución actual del sistema (generalmente 2 o

5) el demonio NTP ha activado. Si usted no sabe el nivel de ejecución actual, puede encontrarlo mediante el comando runlevel:

root@equipo:~# runlevel

Esto configura el demonio NTP para ejecutarse en el nivel de ejecución actual. A continuación, deberá o bien reiniciar el host, o manualmente iniciar el demonio NTP para continuar el proceso de instalación. Puede iniciar el demonio de forma manual mediante el comando:

root@equipo:~# /etc/init.d/ntpstart

root@equipo:~# /etc/init.d/ntp stop

root@equipo:~# /etc/init.d/ntprestart

ó a través de los siguientes comandos:

root@equipo:~# servicentpstart

root@equipo:~# servicentpstotp

root@equipo:~# servicentprestart

Tras activar el servicio o al arrancar el ordenador, durante los primeros minutos, el reloj estará desincronizado. Una llamada a ntptrace como root puede dar como resultado lo siguiente:

root@equipo:~# ntptrace

localhost: stratum 16, offset 0.000073,

synchdistance 0.00000,0.0.0.0: *No sincronizado*

Tras algunos minutos, por fin se sincroniza. Ahora el listado es:

root@equipo:~# `ntptrace`

localhost: stratum 3, offset 0.000064, synchdistance 0.50545

offset 0.001512

Una vez instalado simplemente deberemos configurar su archivo de configuración [13],/etc/default/ntpdate:

root@equipo:~# :/etc/default# gedit ntpdate

Cron y Crontab

El nombre cron viene del griego chronos que significa “tiempo”. Cron es un administrador regular de procesos en segundo plano (demonio) que ejecuta procesos o guiones a intervalos regulares (por ejemplo, cada minuto, día, semana o mes). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero crontab [16].

Cron se ejecuta en el fondo, revisa cada minuto la tabla de tareas crontab/etc/crontab o en /var/spool/cron en búsqueda de tareas que se deban cumplir. Como usuario podemos agregar comandos o scripts con tareas a cron para automatizar algunos procesos. Esto es útil por ejemplo para automatizar la actualizacion de un sistema.

Crontab es un simple archivo de texto que guarda una lista de comandos a ejecutar en un tiempo especificado por el usuario. Crontab verificará la fecha y hora en que se debe ejecutar el script o el comando, los permisos de ejecución y lo realizará en el fondo.

Cada usuario puede tener su propio archivo crontab, de hecho el /etc/crontab se asume que es el archivo crontab del usuario root.

Primero que nada haremos un script.

Este script será llamado por cron y contendrá todas las instrucciones que queremos que haga, por lo tanto es necesario probarlo en varios casos y de varias formas antes de incluirlo a cron, un sencillo script de actualizacion como este:

```
#!/bin/bash
#script ejemplo de actualizacion
#elija su distribucion
#debian-ubuntu
#apt-getupdate&apt-get -y upgrade
#fedora
#yum -y update
#Arch
#pacman --noconfirm –Syu
```

Quitaremos el # a la línea de nuestra distro. En nuestro caso como es Ubuntu/Debian, a la que empieza con apt-get.

Guardamos el script como actualizacion.sh (ej. directorio scripts tu home). Cambiamos los permisos de ejecucion del dichoso script con:

```
root@equipo:~# chmoda+x ~/scripts/actualizacion.sh
```

Agregar tareas a Crontab

Ejecutamos la edición del crontab con crontab -e, en algunas distros (como ubuntu) nos da la opcion de elegir el editor de textos que deseemos, los demás nos quedamos con vi. El archivo crontab lucirá algo así [16].

```
root@equipo:~#  m h dom dow user command
```

Donde:

- **m** corresponde al minuto en que se va a ejecutar el script, el valor va de 0 a 59.
- **h** la hora exacta, se maneja el formato de 24 horas, los valores van de 0 a 23, siendo 0 las 12:00 de la medianoche.
- **dom** hace referencia al día del mes, por ejemplo se puede especificar 15 si se quiere ejecutar cada dia 15
- **dow** significa el día de la semana, puede ser numérico (0 a 7, donde 0 y 7 son domingo) o las 3 primeras letras del día en inglés: mon, tue, wed, thu, fri, sat, sun.
- **user** define el usuario que va a ejecutar el comando, puede ser root, u otro usuario diferente siempre y cuando tenga permisos de ejecución del script.
- **command** refiere al comando o a la ruta absoluta del script a ejecutar, ejemplo: /home/usuario/scripts/actualizar.sh, si acaso llama a un script este debe ser ejecutable.

Para que quedara claro unos cuantos ejemplos de tareas de cron explicados:

```
15 10 * * * usuario /home/usuario/scripts/actualizar.sh
```

Ejecutará el script actualizar.sh a las 10:15 a.m. todos los días.

```
15 22 * * * usuario /home/usuario/scripts/actualizar.sh
```

Ejecutará el script actualizar.sh a las 10:15 p.m. todos los días.

```
00 10 * * 0 rootapt-get -y update Usuario root
```

Ejecutará una actualización todos los domingos a las 10:00 a.m

```
45 10 * * sunrootapt-get -y update
```

Usuario root ejecutará una actualización todos los domingos (sun) a las 10:45 a.m

```
30 7 20 11 * usuario /home/usuario/scripts/actualizar.sh
```

El día 20 de noviembre a las 7:30 el usuario correra el script.

Igual se pueden manejar rangos especiales:

30 17 * * 1,2,3,4,5

A las 5:30 de la tarde todos los días de lunes a viernes.

00 12 1,15,28 * *

A las 12 del día todos los días primero, quince y 28 de cada mes (ideal para nóminas).

Administracion de trabajos en Cron

root@equipo:~# crontab archivo

Remplaza el existente archivo crontab con un archivo definido por el usuario

root@equipo:~# crontab -e

Editar el archivo crontab del usuario, cada linea nueva sera una nueva tarea de crontab.

root@equipo:~# crontab -l

Lista todas las tareas de crontab del usuario

root@equipo:~# crontab -d

Borra el crontab del usuario

root@equipo:~# crontab -c dir

Define el directoriod e crontab del usuario (este debe tener permisos de escritura y ejecucion del usuario)

root@equipo:~# crontab -u usuario

prefijo para manejar el crontab de otro usuario, ejemplos:

root@equipo:~# crontab -l -u root

root@equipo:~# crontab -e usuario2

root@equipo:~# crontab -d -u usuario

Análisis de Resultados

Durante las pruebas realizadas se encontraron algunos inconvenientes en la sincronzación delos relojes, dentro de los cuales podemos mencionar queuno de los equipo de cómputo, el encargado de tener la función de servidor interno, no mantenía una comunicación al exterior a través de internet con los otros servidores de estrato 1 a los que hace referencia para verificar su hora, y de ser necesario corregirla para que este a su vez permita la correcta sincronización de hora con los clientes que le hacen peticiones. Así, se verificóen cada uno de los pasos la correcta instalación del NTP y el de las configuraciones, probando algunos comando para verficar su funcionamiento como en las siguientes capturas de la terminal:

```
root@arthwr-HP-Pavilion-dv5-Notebook-PC:~# ntpq
ntpq> pee
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 ciclope.azc.uam .INIT.          16 u    -   64    0    0.000    0.000   0.000
 xalli.cie.unam. .INIT.          16 u    -   64    0    0.000    0.000   0.000
 ratir.astrosen. .INIT.          16 u    -   64    0    0.000    0.000   0.000
 juniperberry.ca .INIT.          16 u    -   64    0    0.000    0.000   0.000
 172.20.10.255   .BCST.          16 u    -   64    0    0.000    0.000   0.000
 pbx.denvercolo. .INIT.          16 u    -   64    0    0.000    0.000   0.000
 vimo.dorui.net  .INIT.          16 u    -   64    0    0.000    0.000   0.000
 pool-test.ntp.o .INIT.          16 u    -   64    0    0.000    0.000   0.000
 time.netspectru .INIT.          16 u    -   64    0    0.000    0.000   0.000
ntpq> pee
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 ciclope.azc.uam .INIT.          16 u    -   64    0    0.000    0.000   0.000
 xalli.cie.unam. .INIT.          16 u    -   64    0    0.000    0.000   0.000
 ratir.astrosen. .INIT.          16 u    -   64    0    0.000    0.000   0.000
 juniperberry.ca .INIT.          16 u    -   64    0    0.000    0.000   0.000
 172.20.10.255   .BCST.          16 u    -   64    0    0.000    0.000   0.000
 pbx.denvercolo. .INIT.          16 u    -   64    0    0.000    0.000   0.000
 vimo.dorui.net  .INIT.          16 u    -   64    0    0.000    0.000   0.000
 pool-test.ntp.o .INIT.          16 u    -   64    0    0.000    0.000   0.000
 time.netspectru .INIT.          16 u    -   64    0    0.000    0.000   0.000
ntpq> exit
```

Ilustración 38 Comunicación entre servidores

```
root@arthwr-HP-Pavilion-dv5-Notebook-PC:~# ntpdate -u 148.206.80.16
11 Nov 13:56:12 ntpdate[2861]: no server suitable for synchronization found
root@arthwr-HP-Pavilion-dv5-Notebook-PC:~# ntpdate -u ciclope.azc.uam.mx
11 Nov 13:56:38 ntpdate[2926]: no server suitable for synchronization found
root@arthwr-HP-Pavilion-dv5-Notebook-PC:~# nano /etc/ntp.conf
root@arthwr-HP-Pavilion-dv5-Notebook-PC:~# service ntp restart
 * Stopping NTP server ntpd                                          [ OK ]
 * Starting NTP server ntpd                                          [ OK ]
root@arthwr-HP-Pavilion-dv5-Notebook-PC:~# nano /etc/ntp.conf
root@arthwr-HP-Pavilion-dv5-Notebook-PC:~# nano /etc/ntp.conf
root@arthwr-HP-Pavilion-dv5-Notebook-PC:~# ntpdate -u ciclope.azc.uam.mx
11 Nov 14:14:52 ntpdate[3645]: no server suitable for synchronization found
```

Ilustración 39 Prueba de sincronización

Como se puede observar, en ningún momento se establece la comunicación hacia el servidor antes mencionado, por lo que solamente aparecen como inicializados pero sin ninguna respuesta.

Ahora bien, al momento de realizar la prueba de sincronización hacia un servidor de estrato 1 vía IP, se llevó a cabo la petición. La figura nos permite apreciar que se realiza la petición pero no se sincroniza, por lo que el mensaje que envía es de error al no encontrar ningún servidor adecuado para la sincronización. Por otra parte, se revisaron de nuevo las configuraciones y se llevó a cabo la prueba del reinicio del servidor NTP para garantizar el funcionamiento.

```
root@arthwr-HP-Pavilion-dv5-Notebook-PC:~# nmap -p123 -sU -P0 localhost

Starting Nmap 6.40 ( http://nmap.org ) at 2014-11-11 14:23 CST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00018s latency).
PORT    STATE SERVICE
123/udp open  ntp
```

Ilustración 40 Comando Nmap

Nmap es un nombre abreviado para “Red Mapper" y este comando a través del puerto interno en uso, permitió verificar si se encontra abierto o cerrado y en que servicio estaba siendo utilizado, por lo que el puerto interno no era la afectación.

```
root@arthwr-HP-Pavilion-dv5-Notebook-PC:~# ntpdate -dv ntp.ubuntu.com
11 Nov 14:24:15 ntpdate[4280]: ntpdate 4.2.6p5@1.2349-o Wed Oct  9 19:08:07 UTC 2013
Looking for host ntp.ubuntu.com and service ntp
host found : golem.canonical.com
transmit(91.189.89.199)
transmit(91.189.94.4)
transmit(91.189.89.199)
transmit(91.189.94.4)
transmit(91.189.89.199)
transmit(91.189.94.4)
transmit(91.189.89.199)
transmit(91.189.94.4)
transmit(91.189.89.199)
transmit(91.189.94.4)
91.189.89.199: Server dropped: no data
91.189.94.4: Server dropped: no data
server 91.189.89.199, port 123
stratum 0, precision 0, leap 00, trust 000
refid [91.189.89.199], delay 0.00000, dispersion 64.00000
transmitted 4, in filter 4
reference time:    00000000.00000000  Sun, Dec 31 1899 17:23:24.000
originate timestamp: 00000000.00000000  Sun, Dec 31 1899 17:23:24.000
transmit timestamp:  d80ceef5.64ea9f1a  Tue, Nov 11 2014 14:24:21.394
filter delay:  0.00000  0.00000  0.00000  0.00000
         0.00000  0.00000  0.00000  0.00000
filter offset: 0.000000 0.000000 0.000000 0.000000
         0.000000 0.000000 0.000000 0.000000
delay 0.00000, dispersion 64.00000
offset 0.000000

server 91.189.94.4, port 123
stratum 0, precision 0, leap 00, trust 000
refid [91.189.94.4], delay 0.00000, dispersion 64.00000
transmitted 4, in filter 4
reference time:    00000000.00000000  Sun, Dec 31 1899 17:23:24.000
originate timestamp: 00000000.00000000  Sun, Dec 31 1899 17:23:24.000
transmit timestamp:  d80ceef5.981b7664  Tue, Nov 11 2014 14:24:21.594
filter delay:  0.00000  0.00000  0.00000  0.00000
         0.00000  0.00000  0.00000  0.00000
filter offset: 0.000000 0.000000 0.000000 0.000000
         0.000000 0.000000 0.000000 0.000000
delay 0.00000, dispersion 64.00000
offset 0.000000

11 Nov 14:24:23 ntpdate[4280]: no server suitable for synchronization found
```

Ilustración 41 Contacto servidor NTP

Otra manera de ver cómo está trabajando NTP es mediante el empleo del comando ntpdate -d. Este comando permite contactar con un servidor NTP y determinar la diferencia de tiempos, pero no modificará el reloj de su sistema. En la figura se puede apreciar que encuentra al servidor y posteriomenteempieza a transmitir pero sin respuesta de conexión, apareciendo como mensaje que el servidor esta caído y sin datos.

```
root@arthwr-HP-Pavilion-dv5-Notebook-PC:~# timedatectl status
      Local time: mar 2014-11-11 14:35:19 CST
  Universal time: mar 2014-11-11 20:35:19 UTC
        RTC time: mar 2014-11-11 20:35:19
        Timezone: America/Mexico_City (CST, -0600)
     NTP enabled: yes
NTP synchronized: yes
 RTC in local TZ: no
      DST active: no
 Last DST change: DST ended at
                  dom 2014-10-26 01:59:59 CDT
                  dom 2014-10-26 01:00:00 CST
 Next DST change: DST begins (the clock jumps one hour forward) at
                  dom 2015-04-05 01:59:59 CST
                  dom 2015-04-05 03:00:00 CDT
```

Ilustración 42 Uso del comando timedatectl

Por último se empleó el comando: timedatectl que permite comprobar la hora actual del hardware y la hora del reloj del sistema, algunos parámetros como son la zona horaria de México, así como también, el encontrar habilitado el NTP y la sincronización.

Así, se detectó que la falla realmente se debe a los mecanismos de seguridad implementados dentro de la Universidad. Esto es,está prohibido el empleo de ciertos protocolos para el desarrollo de algunos sistemas debido aque por su fácil acceso pueden ser utilizados por hackers.

Debido a ello, el sistema fue implementado posteriormente fuera de la Universidad, en donde al realizar las pruebas de conexión y sincronización, permitió hacer uso del NTP en el sistema y establecer una conexión externa a través de internet. Éstos mecanismos de conexión utilizados a través de los puertos de entrada y de salida vía IP, y conectado al equipo con la función de servidor interno, permitieron proporcionar no solo la conexión remota hacia los servidores de estrato 1, sino a demás, la sincronización directa hacia el servidor interno y éste a su vez hacia los clientes, proporcionándoles la corrección exacta de la hora en tiempo real.

De esta forma, al realizar las peticiones de los clientes de forma simultánea hacia el servidor interno, se logró corroborar que la hora que mantenía cada uno de los relojes se encontraba desfasada.

Por lo que en cada petición del cliente al servidor, la corrección de la hora se ajusta al reloj principal y a través de la utilización de comandos de verificación de sincronización, permitieron determinar la diferencia de tiempo de cada uno de los clientes con el servidor interno y éste corregirles la hora correcta proporcionada de los servidores NTP externos.

En cuanto a los resultados, al hacer un análisis en cada uno de ellos, se logra identificar las mejoras que proporciona los algoritmos de Mills en relación a los anteriores, corrigiendo éste los problemas que presenta el algoritmo de Lamport, la relación "sucede antes", es decir, un mensaje no puede ser recibido sin antes ser enviado. Existe una exigencia de resolución de reloj y en adición, resuelve los problemas de escalamiento, ya que solo existe un mensaje entrante y uno saliente para sincronizar cada computadora (ver sección algoritmo de Lamport).

En el caso del algoritmo de Cristian, no solo resuelve el problema de la hora grupal interna, sino que además garantiza que un procesador pueda leer un reloj remoto con más de una precisión específica. Incluye también el tiempo máximo de envió de un mensaje entre más de dos computadoras y mejora la hora al ajuste del reloj; esto es, si la hora de referencia recibida desde el master era menor a la que se desea ajustar, dicho ajuste ahora ya no implica un retraso y por lo tanto también mejora la escalabilidad en la arquitectura cliente-servidor (ver sección algoritmo de Cristian).

Por último con el algoritmo de Berkeley se corrigen las demoras de los paquetes de datos que circulan por la red de interconexión entre computadoras que se desean sincronizar y evita demorar tiempo al preguntar a cada cliente por su hora, calculando sus promedios e informándoles como deben de cambiar su hora, esto es debido a que los algoritmos de Mills realizan todas estas operaciones de forma eficiente y autónoma (ver sección algoritmo de Berkeley).

Conclusiones

Como resultado de la implementación de los algoritmos de Mills para la sincronización de relojes físicos, es posibleverificar el gran avance de la sincronización en los sistemas distribuidos mostrando como solución los inconvenientes que cada uno de éstos algoritmos de sincronización presentan en el paso de mensajes, así como también en el modelo cliente-servidor.

Es posible denotar que los algoritmos de Mills proporcionan un ambiente amigable, seguro, confiable y de fácil utilización, sobre todo alpoder tener más de una fuente de precisión externa y un libre acceso hacia más de una conexión remota a los servidores NTP externos. Esto incluye también una gran variedad de manuales para su correcto funcionamiento y futuras modificaciones a las que se pretendan llegar.

De esa manera, se puede concluir que los algoritmos de Mills, en conjunto, se destacan por contemplar gran cantidad de requerimientos en cuanto a seguridad, escalamiento, sobrecarga y tolerancia a fallos, a diferencia de aquellos que trabajan individualmente. Sin embargo, el hecho de que contemple varios objetivos no permite la optimización para alguno de sus algoritmos de manera particular.

Como posibles trabajos futuros, éste proyecto terminal se podría mejorar al desarrollar dos bibliotecas, una para consulta y actualización de la hora que funcione en forma local en cada nodo, y otra de sincronización de la hora que se ejecute en forma distribuida.

Bibliografía

[1] Coulouris G., Dollimore J., Kinberg T., “Sistemas Distribuidos. Conceptos y Diseño”, 3a edición. Pearson Educación, 2001. ISBN: 8478290494.

[2] Cristian F., Fetzer C., "The Time AsynchronousDistributedSystemModel", IEEE TransactionsonParallelSystems, June 1999, pp. 603618.

[3] Leslie Lamport, "Time, Clocks, and theOrdering of Events in a DistributedSystem," Communications of the ACM, July 1978, 21(7):558-565.

[4] F. Cristian. "ProbabilisticClockSynchronization".Distributed Computing, 3: 146–158, 1989.

[5]RiccardoGusella , Stefano Zatti, The Berkeley UNIX 4.3BSD Time SynchronizationProtocol, University of California at Berkeley, Berkeley, CA, 1985

[6] Mills D.L., "Internet time Synchronization: the Network Time Protocol", IEEE trans. Communications COM39, October 1991, pp. 14821493.

[7] Mills D.L., "Network Time Protocol (Version 3) specification, implementation and analysis", DARPA NetworkingGroupReport RFC1305, Universityod Delaware, March 1992.

[8] Mills D. L., "A BriefHistory of NTP Time: Confessions of an Internet Timekeeper". ACM ComputerCommunicationsReview 33, 2 (April 2003), pp 922.

[9]http://www.uv.es/~montanan/redes/trabajos/ntp.doc

[10]TheScience of Timekeeping.pdf (Allan) en from/mas/tin

[11] Fetzer C., Christian F., "IntegratingExternal and InternalClockSynchronization", June 1996. Real-Time Systems.SpringerNetherlands.ISSN 0922-6443 (Print) 1573-1383 (Online). Vol. 12, Number 2 / marzo de 1997.DOI 10.1023/A:1007905917490. Páginas 123-171. SubjectCollectionInformática.

[12]http://www.elo.utfsm.cl/~mineducagv/docs/ListaDetalladadeModulos/tutorial_ps.pdf

[13]http://eithel-inside.blogspot.mx/2010/01/instalar-y-configurar-un-servidor-y-un.html

[14]http://humanliks.wordpress.com/active-directory-windows-server-2003/servidor-de-tiempo-ntp/

[15]http://www.oxixares.com/~gbv/hora.html#serv_avan

[16]http://blog.desdelinux.net/cron-crontab-explicados/

[17]www.ntp.org

[18]http://support.ntp.org/bin/view/Servers/StratumTwoTimeServers

[19]http://www.ntp.org/ntpfaq/NTP-s-config-adv.htm#AEN3181