UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

"SINCRONIZACIÓN DE RELOJES"

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS (COMPUTACIÓN)

P R E S E N T A :

ALEJANDRO VELÁZQUEZ MENA

DIRECTOR DE TESIS: SERGIO RAJSBAUM GORODESKY

México, D.F. **2008.**

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Índice de Figuras

INTRODUCCIÓN

La sincronización del reloj es uno de los problemas básicos del cómputo distribuido. En términos generales, la meta es asegurar que los procesadores físicamente dispersos busquen una idea en común del tiempo tomando como base los relojes físicos locales (donde las tasas de tiempo pueden variar) para el lograr el intercambio de mensajes sobre una red de comunicaciones con una latencia incierta. La discrepancia entre las distintas lecturas de los relojes se llama sincronización cercana.

Un ejemplo para explicar el problema de la sincronización del reloj en la computación es el funcionamiento del programa make en unix. Lo anterior se presenta cuando un programador termina de modificar todos los archivos fuentes, es entonces cuando inicia make, el cual examina la hora en que todos los archivos fuentes y objetos fueron modificados por última vez.

Si tiene el archivo fuente salida.c la hora 21:51 y el correspondiente archivo objeto salida.o muestra la hora 21:50, make reconoce que salida.c presenta modificaciones desde la creación de salida.o, por lo que entonces hay que volver a compilar salida.c.

Por otro lado si salida.c tiene la hora de 21:44 y salida.o presenta la hora 21:45, entonces no es necesario volver a compilar. Así el programa make revisa todos los archivos fuentes para determinar aquellos que deban volver a compilarse e invocar al compilador para que realice esta tarea.

¿Qué pasa si este mismo ejemplo sucede en un sistema distribuido donde no existe un acuerdo global en el tiempo?

Supongase que salida.o tiene la hora 21:44 y que poco después se modifica salida.c, pero se le asigna la hora 21:43 debido a que el reloj de esta máquina es lento, como se muestra en la figura I-1.

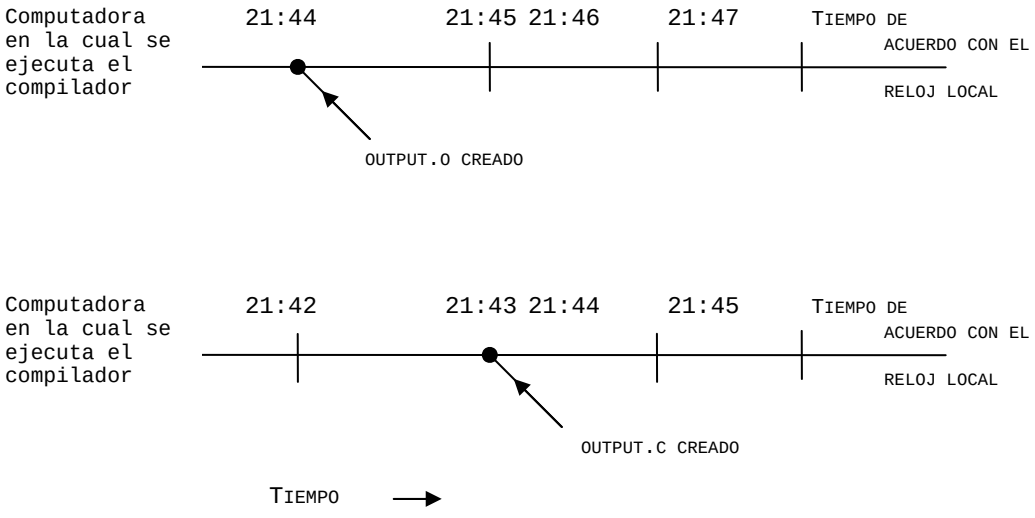


FIGURA I-1. DIAGRAMA DE TIEMPO DE CADA COMPUTADORA, EN ÉSTE LAS COMPUTADORAS ASIGNA EL TIEMPO A CADA EVENTO.

En este caso make no llamará al compilador. El programa en binario ejecutable contendrá una mezcla de archivos objetos de las fuentes anteriores y nuevas. Debido a la mezcla de archivos el programa no funciona y el programador no puede encontrar el error correspondiente, el cual no es por error en el codigo, sino por la mezcla de archivos. El programador tiene opciones para resolver el problema, si es sabe el origen del mismo, sin embargo, le llevará tiempo encontrarlo.

La solución más rápida, pero no la mejor, es “borrar” todos los archivos “.o” para que así, el programa make compile y cree todos los archivos “.o” nuevamente. Esta solución funciona para este tipo de problema pero el origen del mismo es la uniformidad de tiempo en las máquinas involucradas y en ese caso es difícil solucionarlo.

Los algoritmos de sincronización del reloj, se sincronizan con algún *desvío* denominado ε (skew). Garantizan que si r_1 (reloj 1) y r_2 (reloj 2) son dos relojes pertenecientes a los nodos de una red, entonces en cualquier instante de tiempo en r_1 difiere un cierto tiempo de r_2 pero no mayor a ε. La propiedad de sincronización del reloj se proporciona con una probabilidad muy alta más no con una precisión exacta [1].

En la figura I-2 existe la propagación de envíos de mensajes entre dos procesos. Las líneas verticales representan tiempo en el procesador (S y T respectivamente) y las líneas inclinadas representan el mensaje y cuánto tarda en llegar al otro procesador.

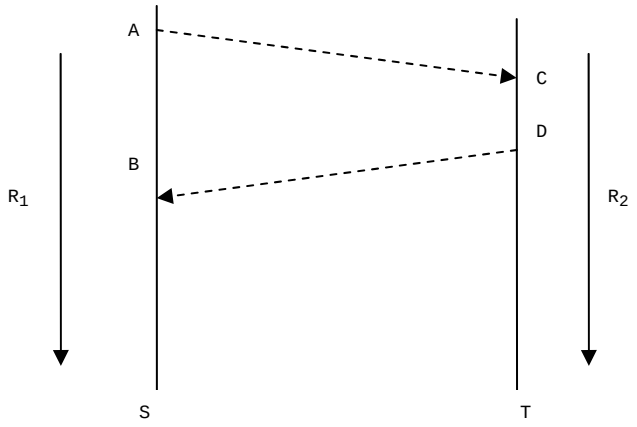


FIGURA I-2. EJEMPLO DE PROPAGACIÓN DE ENVÍO DE MENSAJES ENTRE DOS PROCESOS.

Hay numerosas aplicaciones que usan o dependen de la sincronización de reloj en redes de computadoras, por ejemplo:

1. Sistemas de Base de Datos.

2. Protocolos como el *SCMP* (Synchronized Clock Message Protocol) que garantiza “la mayoría de una vez” en la entrega de mensajes [2].

3. Autenticación con ticket en *Kerberos* [3], usando los esquemas:

   - Servidor generador de ticket (TGS siglas en inglés).
   - Vía cliente/servidor.

4. Conceptos de *atomicidad* y *commit* en Windows usando el sistema de archivos Harp [4].
5. Las versiones de administración y control de concurrencia usualmente dependen de la capacidad de asignar constantemente *marcas de reloj* (timestamp) a los objetos.

La dificultad básica de la sincronización del reloj se presenta, cuando la tasa de los relojes locales desconoce la precisión del avance, la sincronización cercana se va perdiendo con el paso del tiempo y cuando un procesador está comunicando la información sincronizada a los procesadores remotos, éstos tienen herencias acumuladas de una sincronización incierta, a menos que los tiempos de mensajes de transmisión sean precisamente conocidos por los procesadores que participan [5].

No existen los relojes y los enlaces de comunicación ideales. Sin embargo, siempre hay algo que los garantiza sobre el comportamiento sincronizado del sistema:

Se conoce la tasa de progreso de los relojes locales, se sabe de los límites inferiores y superiores en su tasa de progreso con respecto al tiempo real. A estos límites se les conocen como desvíos limitados (bounded_drift) [5].

Se asume que hay límites inferiores y superiores conocidos en el tiempo requerido para transmitir un mensaje, estos límites se llaman latencia del mensaje (message latency bounds).

La esencia de todos los problemas que hay en la sincronización del reloj es la falta de coincidencia (uniformidad) del tiempo y cómo utilizar estos límites para obtener la sincronización más cercana (tight).

Cuando se habla de sincronización local en los sistemas con un CPU, los problemas de las regiones relativas críticas (la exclusión mutua y la sincronización) se resuelven con los métodos de semáforos y monitores. Estos métodos no son adecuados para su uso en los sistemas distribuidos, puesto que siempre se basan (de manera implícita) en la existencia de la memoria compartida. Un ejemplo de ello es:

Si dos procesos que interactúan mediante un semáforo y debe tener acceso a éste, si se ejecutan en el mismo CPU, pueden compartir un semáforo al guardarlo en el kernel y realizar llamadas al sistema para tener acceso a él. Sin embargo, si se ejecutan en computadoras distintas, este método ya no funciona, por lo que necesitan otras técnicas.

Incluso en las cuestiones más sencillas, como el hecho de determinar si el evento *A* ocurrió antes o después que el evento *B* se requiere de una reflexión cuidadosa.

La sincronización es más compleja en los sistemas distribuidos que en los centralizados, puesto que los primeros deben utilizar algoritmos distribuidos. Por lo general, no es recomendable reunir toda la información relativa al sistema en un lugar y después dejar que un proceso la examine y tome una decisión, como se hace en el caso

centralizado. Si se tienen algoritmos distribuidos deben contar con lo siguiente:

1. La información relevante se distribuye entre varias máquinas.

2. Los procesos toman las decisiones sólo con base en la información disponible en forma local.

3. Debe evitarse un punto de falla en el sistema.

4. No existe un reloj común o alguna otra fuente precisa del tiempo global.

Los primeros tres puntos indican que es inaceptable reunir toda la información en un lugar para su procesamiento. Por ejemplo, para llevar a cabo la asignación de recursos (asignar los dispositivos E/S en una forma libre de bloqueos), generalmente no es aceptable enviar todas las solicitudes a un administrador de procesos, el cual los examina y otorga o niega las solicitudes con base en la información de tablas. En un sistema de gran tamaño, tal solución pone en un predicamento al proceso.

El punto 4 es también crucial. En un sistema centralizado, el tiempo no tiene ambigüedades. Cuando un proceso desea conocer la hora llama al sistema y el kernel se la da. Cuando en un sistema local, un proceso *A* pide la hora y poco después, el proceso *B* también la pide, el valor obtenido por *B* es mayor o igual que el valor obtenido por *A*. Ciertamente, no debe ser menor. En un sistema distribuido, es complejo unificar el tiempo de todas las computadoras.

Se cuentan con dos tipos de sincronización que se definen a continuación:

Sincronización Externa. Existe un procesador llamado origen distinguible en el sistema. La tarea de cada procesador diferente a éste, es obtener en cada tiempo, el intervalo más pequeño [a, b] de forma que la lectura actual del reloj origen esté dentro del intervalo [a, b]. Este tipo de sincronización es fundamental para la del reloj.

Sincronización Interna. Mantener todos los relojes del sistema tan cerca (en cuanto al valor del tiempo) uno del otro como sea posible, ejecutando la tasa de reloj de hardware, excepto en los puntos aislados donde los valores del reloj se reinician, la sincronización interna es la derivación de la sincronización externa, ya que es posible obtenerla a partir de la externa.

Objetivos

Se realiza la investigación en la sincronización del reloj y con énfasis especial en la sincronización interna, analizando algunos modelos y si es necesario mejorar los tiempos dependientes para obtener una precisión cercana a la realidad.

Se investiga y se comprede cómo es la solución de los algoritmos distribuidos ya conocidos y posteriormente en su caso, mejorar o se plantea casos diferentes para un enfoque en tiempo real.

Se implanta el algoritmo de sincronización del reloj en una tarjeta *FPGA* (F*ield Programmable Gate Array)* para disminuir los tiempos causados por la arquitectura de la computadora.

Trabajo Relacionado

En el estudio de la sincronización del reloj, existen diversas investigaciones, que se han clasificado según la perspectiva de cada una obteniendo los siguientes modelos: Modelo del Sistema Clásico y Modelo del Sistema en Tiempo Real.

Las siguientes referencias muestran los enfoques y soluciones al problema de la sincronización del reloj. Con esta base se inicia la presente tesis y se busca una línea de investigación para solucionar variables embebidas y las últimas investigaciones se concluyen con modelos de tiempo real. Por esta razón se requiere un algoritmo simple considerando cálculos simples, baja complejidad de tiempo y mensajes, los tiempos en cuanto a límites superior e inferior no excedan los valores establecidos por Liskov. Todo lo anterior han sido investigaciones que han encontrado resultados para acercarse a una solución óptima a este problema. A continuación se muestra una breve descripción de sus hallazgos:

- En [1] Liskov da una explicación de las ventajas e inconvenientes de la sincronización del reloj, así como de aplicaciones que usan directa e indirectamente el algoritmo en sistemas distribuidos y en posteriores trabajos que se ven en [2] y [4].

- En la tesis que se desarrolló en [18], se muestra la perspectiva de cómo minimizar la latencia de los mensajes sin importar qué tipo de sincronización sea (externa o interna), usando la técnica innovadora denominada “gráficas de sincronización” y basándose en las conclusiones de Lundelius y Lynch [7]. La base de [18] son las aplicaciones de Liskov [1], algoritmos investigados y explicados en el libro de Attiya [29], aplicaciones que usan como protocolo el NTP [8] y técnicas que emplean el algoritmo de Cristian [17]. La teoría de Autómatas de I/O [13] es una base para dividir el algoritmo de la computadora y recursos.

- En el artículo [6] se toma como base los resultados de Lundelius y Lynch [7], en [6] se muestra que los resultados no son del todo “reales” y señalan ventajas e inconvenientes para los sistemas en tiempo real, se hicieron mediciones en [19] y propuestas para solucionar en tiempo real.

- En el artículo [7] se hace el análisis de requerimientos para la propuesta de sincronización del reloj para cómputo distribuido, en este artículo se obtienen los límites inferiores y superiores.

- El libro de NTP [8] contiene un protocolo de aplicación probado y eficiente a lo largo de los años con respecto a la sincronización externa. Este libro también habla de la evolución del protocolo y la solución de la sincronización y cómo se inicia a partir de relojes lógicos de Lamport [9].

- El artículo inicial [9] que empieza toda la investigación del orden de los eventos, relojes y una perspectiva del tiempo, da una solución para sincronizar los relojes mediante la técnica de “relojes lógicos”.

Resultados y Organización de la tesis

La investigación para la presente tesis se organiza de la siguiente manera:

En la introducción se explica brevemente la importancia de la sincronización del reloj y aplicaciones en sistemas distribuidos.

En el capítulo 1, llamado marco teórico, se define el problema de la sincronización del reloj desde la lectura con respecto al sol, hasta los relojes atómicos y la relación entre sí. Se introducen conceptos básicos acerca del problema de sincronización del reloj desde del punto de vista de Lamport [9], Lundelius y Lynch [7], Liskov [1], y Mills [8].

En el capitulo 2, denominado modelo del sistema clásico (para la sincronización del reloj), se explica la propuesta del artículo de Boaz Patt-Shamir y Sergio Rajsbaum [5], donde se propone un algoritmo que utiliza gráficas de sincronización (A Theory Clock Synchronization) para resolver el problema de sincronización interna.

En el capítulo 3, denominado modelo del sistema en tiempo real, se explican los conceptos fundamentales de este modelo y se ve el algoritmo en tiempo real, propuesto por Moser y Ulrich [6], así como demostrar las propiedades para los algoritmos distribuidos y su complejidad.

En el capítulo 4 el análisis de requerimientos se inicia a partir de los dos modelos que se investigaron y se propone el modelo a utilizar para implementarlo en hardware y obtener resultados, el criterio de selección para estos dos modelos son las pruebas que sus autores han hecho en los algoritmos.

En el capítulo 5, la implementación del algoritmo de sincronización del reloj en un dispositivo FPGA consta de cinco secciones, las cuales son:

a) Entender el Ethernet MAC IP Core y sus componentes. Se habla con detalle del módulo para comprender y ampliar el protocolo CSMA/CD con el Algoritmo de Sincronización del Reloj (ASR).

b) Crear el módulo Algoritmo de Sincronización del reloj. La sección consiste en traducir el algoritmo propuesto al lenguaje Verilog, después se agrega al módulo Ethernet MAC IP Core.

c) Módulo de gestión de paquetes. El módulo trata de resolver la creación de una trama de Ethernet y envía los datos necesarios hacia los otros dispositivos. Los dispositivos reciben los frames necesarios que incluya la hora local de la computadora que lo está enviando, así como su contraparte para identificar la trama que es un paquete para ejecutar el ASR.

d) El módulo de función de reloj tiene la misión de comunicarse con el CPU, sus dos operaciones básicas son:

- Leer y
- Ajustar la hora local del dispositivo.

Este módulo es uno de los más complicados, aunque su función es relativamente sencilla. No se realiza a nivel aplicación por lo que se hace a nivel FPGA o hardware. Si no se puede realizar bajo este esquema se tiene que usar una solución a nivel aplicación, lo cual afecta a la solución ya que dependerá del calendarizador (scheduler) y el manejo de S.O. en cuestión.

e) El módulo de integración muestra cómo están interconectados los tres módulos al IP Core principal.

En el capítulo de conclusiones se describen los resultados encontrados, problemas actuales y posibles soluciones las cuales no fueron implantadas a lo largo de la tesis. Estas propuestas ayudarán a que versiones futuras del algoritmo funcionen de manera óptima en los dispositivos de hardware para que no dependan del CPU.

Capítulo 1

Marco Teórico

En este capítulo se desarrolla el marco teórico y se describe el problema fundamental de la medición del tiempo y la sincronización del reloj. En las secciones 1.1 y 1.2 se lleva a cabo el análisis para medir el tiempo y explica la evolución de ésta hasta la medición mediante el reloj atómico. La sección 1.3 se muestra el componente principal del reloj: el cuarzo. Se muestran las características, inconvenientes y comportamiento de dicho elemento.

Una vez explicada la medición del tiempo y los componentes, en la secciones 1.5 y 1.6 se describe el problema de la sincronización del reloj y sus dos divisiones: externa e interna.

En las secciones 1.7 y 1.8 se menciona parte del artículo de Lamport y de los relojes lógicos, eventos y el orden parcial, artículo fundamental en esta investigación. Conociendo todo el entorno y problemática, en la sección 1.6 se mencionan algunas aplicaciones que dependen directa o indirectamente de la sincronización del reloj.

1.1 Medición del Tiempo

El concepto del tiempo es fundamental para cualquier actividad que realice. Se deriva desde un concepto básico del orden en el cual ocurren los eventos. Sin embargo, este concepto debe reexaminarse cuidadosamente cuando se considera en las computadoras como sistemas cliente/servidor, centralizados y más aún en sistemas distribuidos.

Para entender el concepto del tiempo, se debe explicar cómo se mide éste en realidad. No es tan sencillo como uno puede pensar, en particular cuando se requiere de gran precisión. Desde la invención de los relojes mecánicos en el siglo XVII, el tiempo se ha medido de manera astronómica. Cada día, el sol parece levantarse en el horizonte por el este, sube hasta una altura máxima en el cielo y desciende por el oeste. Al evento cuando el sol alcanza su punto aparentemente más alto en el cielo se le llama *tránsito del sol*. Éste ocurre diariamente alrededor de las doce del día. Al intervalo entre dos tránsitos consecutivos del sol se le llama *día solar*. Puesto que existen 24 horas en un día y cada una de ellas contiene 3600 segundos, el segundo solar se define exactamente como 1/86400 de un día solar (60 segundos x 60 minutos X 24 horas). La geometría del cálculo del día medio solar se muestra en la figura 1-1.

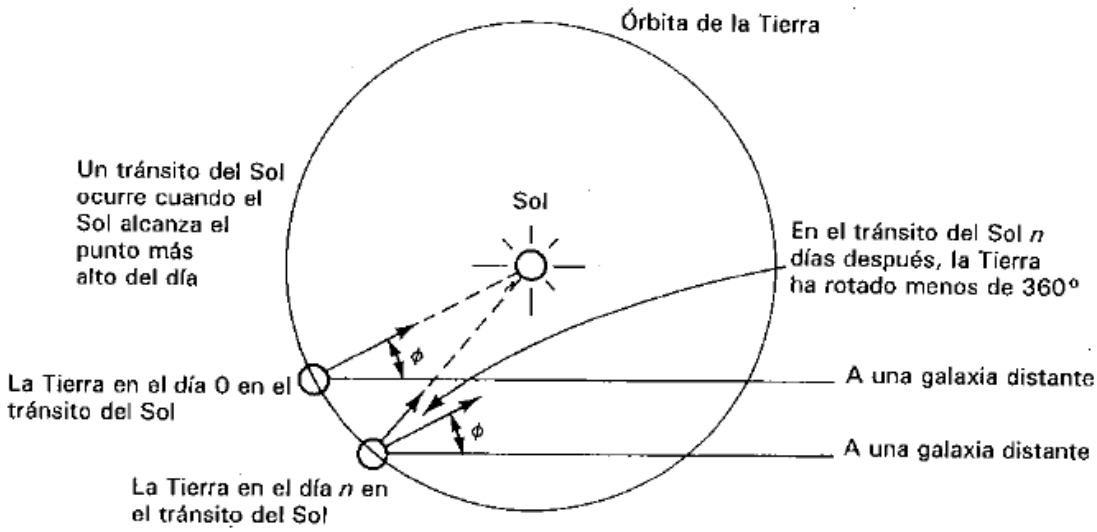


FIGURA 1-1. CÁLCULO DEL DÍA SOLAR PROMEDIO.

En la década de 1940, se estableció que el período de rotación de la tierra no es constante, esto se debe a que la tierra está disminuyendo su velocidad, debido a la fricción tidal y al arrastre atmosférico. Con base en estudios de los patrones de crecimiento del coral antiguo, los geólogos piensan hoy en dia, que hace 300 millones de años existían cerca de 400 días al año. Aunque la duración del año no ha cambiado considerando a esta como el tiempo que tarda la tierra en dar una vuelta alrededor del sol, los días son más largos. Además de la tendencia a largo plazo, existen pequeñas variaciones a corto plazo, de la duración del día, lo cual probablemente se deba a una turbulencia originada en el centro de acero fundido de la Tierra. Estos descubrimientos han llevado a los astrónomos a calcular la longitud del día mediante la medición de un gran número de días y considerar el su promedio antes de dividir entre 864000. A la cantidad resultante se llama segundo solar promedio.

1.2 Reloj Atómico

Con la invención del reloj atómico en 1948, fue posible medir el tiempo de manera mucho más exacta, independientemente del ir y venir de la tierra, cuando se cuentan las transiciones del átomo de cesio 133. Los físicos retomaron de los astrónomos la tarea de medir el tiempo y definieron el segundo como el tiempo que tarda el átomo de cesio 133 en hacer exactamente 9192631770 transiciones. La elección de 9192631770 se consideró para que el segundo atómico fuera igual al segundo solar promedio en el año de su introducción. Actualmente, cerca de 50 laboratorios en el mundo tienen relojes de cesio 133, en forma periódica, cada laboratorio le indica a la oficina internacional de la hora en París (BIH) el número de marcas de su reloj. La oficina hace un promedio de estos números para producir el tiempo atómico internacional (TAI). Así que el TAI es el promedio de las marcas de los relojes de cesio 133 a

partir de la medianoche del primero de enero de 1958 (principio del tiempo), dividido entre 9192631770.

TAI es muy estable y está disponible para todos los que quieran comprarse un reloj de cesio, pero existe un problema serio con él; 86400 segundos TAI equivalen a 3 milisegundos menos que un día solar medio (puesto que este día solar promedio es cada vez más grande). El uso de TAI para el registro del tiempo significaría que, con el paso de los años, el mediodía sería cada vez más temprano, hasta llegar al momento en que éste mediodía ocurriría en la mañana.

La BIH resolvió el problema mediante la introducción de segundos de salto, siempre que la discrepancia entre el TAI y el tiempo solar crezca hasta 800 milisegundos.

Los segundos TAI son de longitud constante, a diferencia de los segundos solares, los segundos de salto se introducen cuando es necesario mantenerse en fase con el sol como se muestra en la figura 1-2.

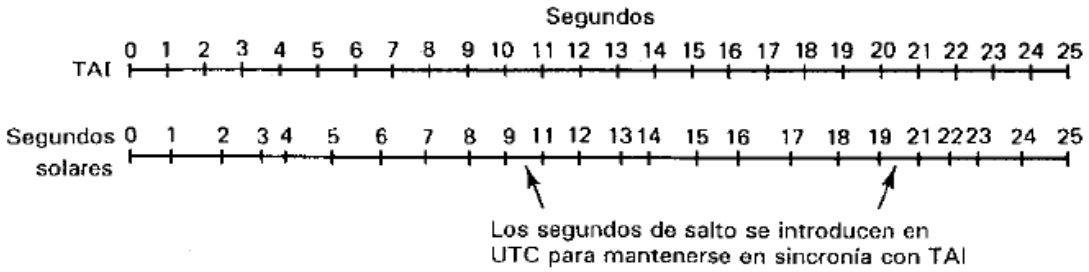


FIGURA 1-2. REPRESENTACIÓN DE LOS SEGUNDOS TAI, SOLARES Y DE SALTO.

Esta corrección da lugar a un sistema de tiempo basado en los segundos constantes de TAI, pero que permanece en fase con el movimiento aparente del Sol, a esto se le llama tiempo coordenado universal (UTC). UTC es la base de todo el sistema de mantenimiento moderno de la hora. En lo esencial, ha reemplazado al estándar anterior, el tiempo del meridiano de Greenwich, que es un tiempo astronómico.

La mayoría de las compañías que proporcionan la luz eléctrica basan sus relojes de 60 Hz o 50 Hz (América y Europa respectivamente) en el UTC, por lo que cuando BIH anuncia un segundo de salto, las compañías elevan su frecuencia a 61 o 51 Hz durante 60 o 50 segundos, para que avancen todos los relojes del área de distribución. Puesto que un segundo es un intervalo notable para una computadora, un sistema operativo que necesite mantener un tiempo exacto durante un periodo de algunos años debe tener un software especial para registrar los segundos de salto conforme sean anunciados (a menos que utilicen la línea de corriente eléctrica para medir su tiempo, lo cual es demasiado obsoleto). El número total de segundos de salto introducidos en UTC hasta ahora es cercano a 30.

1.3 Reloj Físico en la PC

Antes de hablar del reloj, primero se debe que hablar de los componentes que lo integran y el más importante en su caso es el cuarzo, a continuación se describe el mismo.

El cuarzo es un mineral muy común, de hecho forma parte del granito, es una roca muy utilizada en la construcción. Tiene una propiedad llamada piezo-electricidad, esto es, si se presionan las dos caras de un cristal de cuarzo, se obtiene entre ellas un voltaje eléctrico. Pero este efecto es reversible, ya que si se aplica un voltaje eléctrico con la misma polaridad entre ambas caras, el cristal se comprime de la misma forma que si se hubiese presionado físicamente.

Si se alterna la polaridad, el cristal vibrará (comprimiéndose y expandiéndose) al ritmo de la frecuencia del voltaje aplicado, y si esa frecuencia es similar a la de vibración natural del cristal, entonces entra a lo que se llama resonancia, momento en el cual alcanza su máxima intensidad de vibración.

Por lo tanto, un cristal de cuarzo es capaz de convertir una fuerza mecánica en una energía eléctrica y viceversa, una energía eléctrica en una fuerza mecánica. Tal propiedad ha sido aprovechada en el diseño de numerosos circuitos que se utiliza en la vida cotidiana: este componente es el corazón del sintonizador del televisor, del aparato de video, del receptor de radio, del teléfono móvil, del reloj de pulsera, la computadora, entre otros.

De hecho, la computadora puede compartir varios cristales de cuarzo, el del microprocesador y sus periféricos (impresora, escáner), todos ellos vibrando sincronizadamente para que no se produzca congestionamiento ni colisiones en las autopistas electrónicas por donde tienen que circular los bits.

Por las propiedades mecánicas, eléctricas y químicas, el cuarzo es el material más apropiado para fabricar dispositivos con frecuencia controlada. Para mayor información consultar el apéndice D.

Todas las computadoras cuentan con un circuito para el registro del tiempo. A pesar del uso generalizado de la palabra *reloj* para hacer referencia al dispositivo, en realidad no se trata de relojes en el sentido usual. Cronómetro es una palabra apropiada para hacer referencia al dispositivo. A cada cristal se le asocian dos registros, un contador y un registro mantenedor.

Cada oscilación del cristal disminuye en 1 al contador, cuando éste toma el valor de cero (0), se genera una interrupción y el contador se vuelve a cargar mediante el registro mantenedor. De esta forma, es posible programar un cronómetro para que genere una interrupción 60 veces por cada segundo o con cualquier frecuencia que se desee. Cada interrupción recibe el nombre de marca de reloj.

Los cronómetros reales no realizan interrupciones exactamente H veces por segundo. En teoría, un cronómetro con H=60 generaría 216000 marcas por hora. En la práctica, el error relativo que se puede obtener con los

circuitos de cronómetros modernos es de cerca de 10^{-5}, lo que significa que una computadora en particular puede obtener un valor del margen que va de 215998 a 216002 marcas por hora. Esto es, si existe una constante tal que

$$1-\rho \le \frac{dC}{dt} \le 1+\rho$$

Se dice que el cronómetro trabaja dentro de su especificación. La constante ρ es especificada por el fabricante y se conoce como la tasa máxima de desvío. En la figura 1-3 se muestra un reloj lento, otro perfecto y uno más rápido.

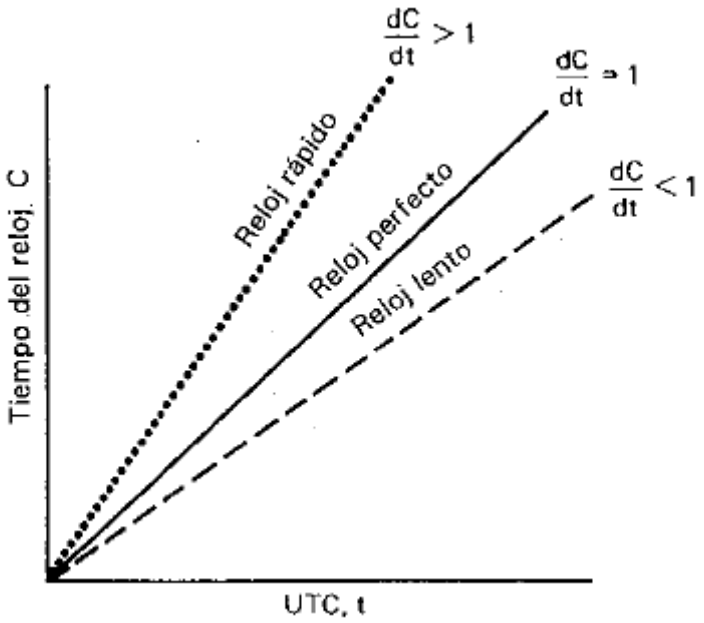


FIGURA 1-3. VALORES DEL DESVÍO DE TRES RELOJES.

Si dos relojes se alejan de UTC en la dirección opuesta, en un instante Δt después de que fueron sincronizados, podrían estar tan alejados como $2\rho\ \Delta t$. Si los diseñadores del sistema operativo desean garantizar que dos relojes cualesquiera no difieran más que δ, entonces los relojes deben volverse a sincronizar (en software) al menos cada $\delta/2\rho$ segundos. Los distintos algoritmos difieren en la forma precisa en la cual se realiza esta resincronización [17].

1.4 Formas para obtener el tiempo

Para proporcionar UTC a las personas que necesitan un tiempo preciso, el Instituto Nacional del Tiempo Estándar (NIST) opera una estación de radio de onda corta con las siglas WWV desde Fort Collins, Colorado. WWV transmite un pulso corto al inicio de cada segundo UTC. La precisión del propio WWV es de cerca de ±1 milisegundo, pero debido a la fluctuación atmosférica aleatoria que puede afectar la longitud de la trayectoria de la señal, en la práctica la precisión no es mejor que ±10 milisegundos. En Inglaterra, la estación MSF que opera desde Rugby, condado de Warwick, proporciona un servicio similar al de otras estaciones en varios países.

Varios satélites terrestres también ofrecen un servicio UTC. El satélite de ambiente operacional Geoestacionario (GEOS) puede proporcionar UTC con una precisión de hasta 0.5 milisegundos y algunos otros satélites lo hacen incluso mejor.

El uso del radio de onda corta o los servicios de satélite requiere un conocimiento preciso de la posición relativa al emisor y receptor para compensar el retraso de la propagación de la señal. Se dispone en forma comercial de esos radiorreceptores WWV, GEOS y otras fuentes de UTC. También existe vía telefónica desde el NIST en Fort Collins, pero hay que hacer una corrección por la ruta de la señal y la velocidad del modem. Esta corrección introduce por lo general cierta imprecisión, lo que dificulta la obtención del tiempo con una precisión extremadamente alta.

1.5 La sincronización del reloj

La sincronización del reloj es uno de los problemas básicos del cómputo distribuido. Se permite que las computadoras conectadas en red tengan algunos medios para fijar sus relojes a la hora nominal del día, incluso si esos medios ascienden al globo ocular y al reloj de pulsera. ¿Se puede hacer esto con precisión en la práctica? La mayoría de la gente que llega a trabajar a tiempo fija su reloj dentro de uno o dos minutos del tiempo mencionado usando el radio o la TV y espera que se desvíe al menos un minuto sobre el mes. Esto asciende a una tasa de error cerca de 23 partes por millón (ppm), no es malo para un reloj de pulsera. Los relojes reales de la computadora se pueden fijar con el cronómetro generalmente en un rango de uno a dos minutos, pero algunos tienen una tasa de error de diez veces más que los relojes de pulsera [8].

Cuando se habla de sincronización local en los sistemas con un CPU, los problemas en las regiones relativas críticas, la exclusión mutua y la sincronización se resuelven con los métodos de los semáforos y monitores. Estos métodos no son adecuados para emplearse en los sistemas distribuidos puesto que siempre se basan (de manera implícita) en la existencia de la memoria compartida. Un ejemplo de ello es:

Cuando dos procesos que interactúan mediante un semáforo y deben tener acceso a este, estos si se ejecutan en el mismo CPU pueden compartir un semáforo al guardarlo en el kernel y realizar llamadas al sistema para tener acceso a él. Sin embargo, si se ejecutan en computadoras distintas, este método ya no funciona, por lo que necesitan otras técnicas.

Incluso en las cuestiones más sencillas, como el hecho de determinar si el evento *A* ocurrió antes o después que el evento *B* se requiere de una reflexión cuidadosa.

La sincronización es más compleja en los sistemas distribuidos que en los centralizados, puesto que los primeros deben utilizar algoritmos distribuidos. Por lo general, no es recomendable reunir toda la información relativa al sistema en un lugar y después dejar que un proceso la examine y tome una decisión, como se hace en el caso centralizado. Si se tienen algoritmos distribuidos se debe contar con lo siguiente:

1. La información relevante se distribuye entre varias máquinas.
2. Los procesos toman las decisiones sólo con base en la información disponible en forma local.
3. Debe evitarse un punto de falla en el sistema.
4. No existe un reloj común o alguna otra fuente precisa del tiempo global.

Los primeros tres puntos indican que es inaceptable reunir toda la información en un lugar para su procesamiento. Por ejemplo, para llevar a cabo la asignación de recursos (asignar los dispositivos E/S en una forma libre de bloqueos), generalmente no es aceptable enviar todas las solicitudes a un administrador de procesos, el cual examina a todos y otorga o niega las solicitudes con base en la información de tablas. En un sistema de gran tamaño, tal solución pone en predicamento al proceso.

El último punto (4) es también crucial. En un sistema centralizado, el tiempo no tiene ambigüedades. Cuando un proceso desea conocer la hora, llama al sistema y el kernel se la da. Cuando en un sistema local un proceso *A* pide la hora y poco después, el proceso *B* también la pide, el valor obtenido por *B* es mayor o igual que el valor obtenido por *A*. Ciertamente, no debe ser menor. En un sistema distribuido, no es común unificar el tiempo a todas las computadoras.

Un sistema distribuido consiste de una colección de distintos procesos los cuales están separados, y se comunican unos con otros por intercambio de mensajes.

Respecto a los errores de los relojes y siendo demasiado entusiastas, la única consecuencia seria es, al llegar un correo electrónico, no importa mucho si un reloj acumula el error de a lo más un minuto por mes.

El panorama cambia si en vez de un correo es un sistema distribuido de reservación de aerolíneas, donde un asiento no se vende o se vende dos veces, por el error de los relojes.

En los puntos que menciona Liskov [1] hay un número de aplicaciones que utiliza la sincronización del reloj y se realiza en misiones fáciles y de manera directa, sin embargo, en ellas no se observa de manera directa la aplicación de la sincronización del reloj.

Los investigadores en ciencias de la computación ven al modelo cronometrado en una red de computadoras distribuidas (sincronización del reloj) como una relación que ocurre antes de (conocida en inglés como happens-before); esto es, que todo evento que ocurre en una computadora debe pasar antes de que las nuevas noticias se obtengan por otra computadora.

En el esquema de Lamport [9], el tiempo nunca se pone hacia atrás, el cual viola la relación de “ocurre antes de”.

Los protocolos de sincronización que se usan hoy en día proveen diferentes significados, pero todos siguen el mismo modelo general. El cliente envía una petición al servidor y el servidor responde con la hora actual. Para mejor precisión, el cliente necesita medir el retardo de propagación del cliente/servidor para determinar el tiempo real compensado concerniente al servidor.

Los protocolos de sincronización trabajan en uno o más modos asociados, dependiendo del diseño del protocolo establecido. Existen tres clases de asociación: persistente (persistent), pre-hecho (preemtable) y efímero (ephimeral).

Las asociaciones persistentes (persistent) modifican según lo dirigido por el archivo de configuración y no cambian los parámetros.

Las asociaciones pre-hechas (preemtable) también se modifican como las persistentes, pero su modificación se cambian sí encuentran al servidor por varios minutos.

Las asociaciones efímeras (ephimeral), se modifican sobre la recepción de un paquete diseñado para ese propósito, por ejemplo un paquete de modo de difusión, y se cambia si el servidor no responde por varios minutos. El término “difusión” (broadcast) se debe interpretar tomando como base al Protocolo de Internet versión 4(IPv4) [8].

Mantener los tiempos locales de los procesos en un sistema distribuido sincronizado con presencia de fallas arbitrarias es importante para muchas aplicaciones y es un problema interesante. Tomando en cuenta los relojes con desvíos en tiempo real y que varía la expedición del mensaje en plazos, hace al problema más realista y desafiante. Para ser verdaderamente útil, una solución a éste debe permitir que los procesos con fallas se recuperen para ser reintegrados al sistema.

Se hace un modelo de un sistema distribuido que consiste en un conjunto de procesos que se comuniquen por envío de mensajes de uno a otro. Cada proceso tiene un reloj físico que no está bajo su control.

Un mensaje típico [7] consiste en texto y el nombre de los procesos enviados. Hay también dos mensajes especiales:

Mensaje START, es el mensaje que comienza desde un origen externo e indica que el recipiente deberá iniciar el algoritmo.

El mensaje TIMER activa al proceso para que reciba información cuando el reloj físico llegue al tiempo designado.

Desde un enfoque se observa que un proceso es un autómata con un conjunto de estados y una función de transición. La función de transición describe el nuevo estado donde el proceso entra, los mensajes se envían hacia afuera y los contadores de tiempo se fijan en el mismo, todos son en función del estado actual de los procesos, el mensaje recibido y el tiempo de reloj físico. Una aplicación de la función de transición constituye un proceso de paso [7].

Se define un reloj que sea monótonamente creciente, por todas partes de la función diferenciable de R(tiempo real) a R(tiempo de reloj). Un sistema de procesos consiste en un conjunto de procesos, un subconjunto de los procesos llamados los procesos que ellos mismos empiezan, y un sistema de los relojes (los relojes físicos), uno para cada proceso [7].

Cada proceso puede comunicarse directamente con otro, incluyéndose. El sistema de mensaje es modelado por un buffer de mensajes global. Cuando un proceso envía un mensaje en el tiempo real t a otro proceso, el mensaje se coloca en el buffer de mensajes junto con un tiempo t' mayor que t. En el tiempo real t', el mensaje es recibido por el recipiente apropiado y borrado desde el buffer. El mensaje se retrasa t' - t. Inicialmente el buffer de mensajes no contiene mensajes excepto los mensajes START, exactamente uno para cada proceso que empiece por sí mismo.

Todos los relojes limitados con ρ (desvío del reloj), incluyen los procesos con fallas. Los procesos con fallas se permiten tomar medidas arbitrarias, los relojes con fallas no aumentan su desempeño para afectar el comportamiento de procesos sin fallas.

Está a lo más f procesos con fallas, para una constante fija f, y el número total de procesos en el sistema n, son al menos 3f + 1 [15]. En [15] muestra que es imposible sin autentificación la sincronización del reloj a menos que las 2/3 partes de los procesos sean sin fallas.

El retardo de cada mensaje se encuentra dentro del rango $[\delta-\varepsilon, \delta+\varepsilon]$ para constantes no negativas δ y ε con $\delta>\varepsilon$.

Hay numerosas aplicaciones que usan o dependen de la sincronización del reloj en redes de computadoras. Por ejemplo:

1) Sistemas de base de datos.
2) Protocolos como el SCMP (Synchronized Clock Message Protocol) que garantiza "la mayoría de una vez" en la entrega de mensajes [2]
3) Autenticación con ticket en Kerberos [3], usando los esquemas

    - Servidor generador de ticket (TGS siglas en inglés).
    - Vía cliente/servidor.

4) Conceptos de *atomicidad* y *commit* en Windows usando el sistema de archivos Harp [4].
5) Las versiones de administración y control de concurrencia usualmente dependen de la capacidad de asignar constantemente *marcas de reloj* (timestamp) a los objetos.

Estos ejemplos se encuentran en el Apéndice E, aplicaciones que utilizan la sincronización del reloj.

1.6 El Orden Parcial

Mucha gente probablemente dice que un evento *a* pasó antes que un evento *b*, Si *a* pasó en un tiempo anterior que *b*. Si la especificación es en términos de tiempo físicos, entonces el sistema debe contener relojes reales. Incluso si contiene relojes reales, todavía hay el problema que tales relojes no fuesen perfectamente precisos y no guarden el tiempo físico exacto [9].

El sistema está compuesto de una colección de procesos. Cada proceso se encuentra conformado por una secuencia de eventos. La dependencia de la aplicación, la ejecución de un subprograma en una computadora, o la ejecución de una simple instrucción a nivel máquina es un evento.

Los eventos de un proceso forman una secuencia. Cuando *a* ocurre antes de *b* en esta secuencia, sucede que *a* se presenta antes que *b*. En otras palabras, un proceso simple está definido para ser un conjunto de eventos con un orden total a priori. Se asume que enviar o recibir un mensaje es un evento de un proceso [9].

Tan pronto se comienza a trabajar con varias máquinas, cada una con su propio reloj, la situación es distinta. Aunque la frecuencia de un oscilador de cristal es muy estable, es imposible garantizar que los cristales de las computadoras oscilen precisamente con la misma frecuencia. En la práctica, cuando un sistema tiene *n* computadoras, los *n* cristales correspondientes oscilarán con tasas distintas, lo que provoca una pérdida de sincronía en los relojes (de software) y que al leerlos tengan valores distintos. A la diferencia entre los valores de tiempo se le llama distorsión del reloj. Como consecuencia de esta distorsión, podrían fallar los programas que esperan que el tiempo asociado a un archivo, objeto, proceso o mensaje sea correcto o independiente del sitio donde haya sido generado, es decir, independiente del reloj utilizado.

En el artículo [9] se muestra lo siguiente. Definición 1. La relación "→" es el conjunto de eventos de un sistema de la relación más pequeña que satisface las 3 condiciones siguientes:

1) Si *a* y *b* son eventos de un mismo proceso, y *a* ocurre antes que *b*, entonces $a \rightarrow b$.
2) Si *a* envía un mensaje por un proceso y *b* es el receptor del mismo mensaje por otro proceso, entonces $a \rightarrow b$.
3) Si $a \rightarrow b$ y $b \rightarrow c$ entonces $a \rightarrow c$. Dos procesos distintos *a* y *b* son concurrentes si $a \not\Rightarrow b$ y $b \not\Rightarrow a$

1.7 Relojes Lógicos

Se introduce un concepto que inició a solucionar el problema del tiempo en el sistema, usando relojes lógicos (logical clocks). Se comenzó con un punto de vista abstracto, donde el reloj es un camino para asignar un número a cada evento, donde este se refiere al tiempo que se tarda el CPU en procesar tal.

Se define un reloj C_i para cada proceso y sea P_i una función que asigna un número $C_i(a)$, para cualquier evento *a* en este proceso. El sistema entero de relojes está representado por la función *C,* la cual asigna para

cualquier evento *b,* el número C(b). Donde $C(b)=C_j(b)$ si *b* es un evento en el proceso P_j. Desde ahora, no asume la relación de los números $C_i(a)$ para el tiempo físico, así podemos pensar que los relojes C_i son relojes lógicos más que relojes físicos. Deben estar implantados por contadores sin mecanismo actual de sincronización [9].

Se considera que el significado de un sistema de relojes es correcto. No se puede basar la definición de correcto (*correctness*) en un tiempo físico, desde que se desea requerir la introducción de relojes, los cuales guardan el tiempo físico. La definición se basa en el orden de cuales de los eventos ocurren. La condición razonable más fuerte es que si un evento *a* ocurre antes de que otro evento *b*, *a* debe suceder en un tiempo anterior que *b*.

Condiciones de Reloj. Para cualquiera de los eventos *a,b* :

Si a → b entonces C(a) < C(b)

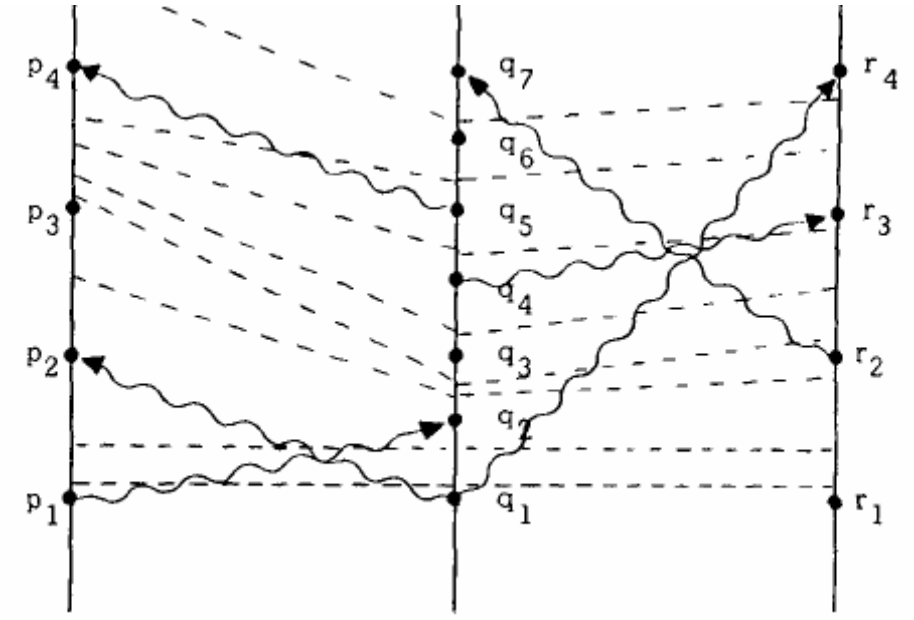


FIGURA 1-4. DIAGRAMA ESPACIO TIEMPO DE TRES PROCESOS DIFERENTES Y QUE SE COMUNICAN ENTRE SÍ.

Nótese que no se puede esperar la condición de conversión para detenerlo también, se desea que cualquiera de los dos eventos concurrentes debe ocurrir al mismo tiempo. En la figura 1-8, p_2 y p_3 son ambos concurrentes con q_3, significa que ambos deben ocurrir al mismo tiempo que q_3, lo cual se contradice con la condición del reloj $p_2 \rightarrow p_3$.

Es fácil ver la relación de "—>" desde la definición en [9] la condición del reloj se satisface con las dos condiciones que se describen a continuación:

Condición 1. Si *a* y *b* son eventos en el proceso P_i, y *a* ocurre antes de *b*, entonces $C_i(a)<C_i(b)$.

Condición 2. Si *a* es el envío de un mensaje por el proceso p_i y *b* es el receptor del mensaje por el proceso P_j, entonces $C_i(a) < C_j(b)$.

Ahora se asume que los procesos son algoritmos, y los eventos representan las acciones seguras durante sus ejecuciones. Se muestra como introducir los relojes dentro de los procesos, y los cuales satisfacen la condición del reloj. Los procesos P_i's de reloj están representados por un registro C_i, así que $C_i(a)$ es el valor que contiene por C_i durante el evento *a.* El valor de C_i cambiará entre los eventos, así que al cambiar C_i de sí mismo no constituye un evento [9].

1.8 Sincronización Externa e Interna

Sincronización Externa. Existe un procesador en el sistema distinguible llamado origen. La tarea de cada procesador diferente a éste, es obtener en cada tiempo, el intervalo más pequeño [a, b] de tal forma que la lectura actual del reloj origen esté entre [a, b]. El modelo de sincronización externa del reloj es como sigue. Existe un procesador distintivo llamado *s* o el origen, su reloj local muestra exactamente el tiempo real, ejemplo, para cualquier estado *x* tenemos local_time(x)=real_time(x).

El requerimiento correcto de un CSA (Algoritmo de Sincronización de Reloj) en cualquier procesador, donde el tiempo lógico es (T,x) para todo estado *x* se satisface: real_time(x) ε $[T-\varepsilon, T+\varepsilon]$. Se llaman CSA's externos [5].

Sincronización Interna. Sincronizar todos los relojes del sistema tan cerca uno del otro como sea posible, ejecutando la tasa de relojes de hardware físicos, excepto en los puntos aislados donde los valores del reloj son reiniciados.

Intuitivamente, la meta de la sincronización interna es mantener los relojes lógicos tan cercanos como sea posible. Para evitar el asunto de la *vivacidad* (liveness) al demostrar cuando se provee el límite inferior, se utiliza una definición llamada el "problema con un disparo" (one shot). La sincronización interna es la derivación de la sincronización externa, ya que es posible obtenerla a partir de ésta.

Concluyendo con este capítulo se adquiere el inicio del problema de la sincronización, cómo se ha resuelto desde los diferentes enfoques y cómo afecta en el campo de la computación, con estos conocimientos, en los capítulos posteriores se habla de las soluciones y el analisis de los mismos para obtener una terminación a este problema y que sea válida en tiempo real.

Capítulo 2

Modelo del Sistema Clásico

Este capítulo describe el algoritmo propuesto en [5] para reducir los tiempos en los envíos de mensajes (latencia mínima) usando “gráficas de sincronización”. En las secciones desde 2.1 hasta la 2.3 se definen los conceptos básicos de la teoría y de la gráfica de sincronización. Para comprender dichos conceptos se incluye un ejemplo en las secciones 2.4 y 2.5 y con profundidad en el anexo F, donde se muestra el comportamiento del mensaje y los tiempos obtenidos.

Desde la sección 2.6 hasta la 2.9 se mencionan y desarrollan los elementos que permiten generar una gráfica de sincronización y obtener la latencia para calcular el tiempo real.

Para estudiar los problemas de sincronización, inicialmente se define un modelo del sistema, se analiza desde un nivel abstracto en la teoría de gráficas. Se aplica el modelo del sistema que arroja resultados y éstos se emplean para realizar una gráfica que permiten analizar los problemas de sincronización de los relojes que son más prácticos que la variante elemental. Con lo anterior se obtienen dos tipos de tareas de sincronización de los relojes, al final de estos cálculos realizados llevados a cabo se plentea la solución para la sincronización del reloj.

2.1 Teoría de la Sincronización

Los modelos de sistemas se consideran cuando la incertidumbre proviene de una variante de retraso del mensaje, fallas y desvíos de relojes. Hoy en día reciben el nombre de modelos síncronos (non-lockstep). Al realizar investigaciones sobre los modelos clásicos, se encontraron varios, uno que se basa en gráficas dirigidas y libres de fallas. Otro en el que contempla la tolerancia a fallas simples. Al leer algunos modelos se decide por uno al considerarlo el mas completo, esta decisión se basa en que su algoritmo y funcionamiento pueda implementarse en hardware.

El modelo que se describe en esta sección se basa principalmente en el modelo de autómatas de tiempo de I/O (TIOA) de Lynch y Vaandrager [16], para ello considérese que un ambiente es la composición de todos los módulos de envío y los enlaces de comunicación. En la Figura 2-1 se muestra el modelo conceptual de autómatas de tiempo.

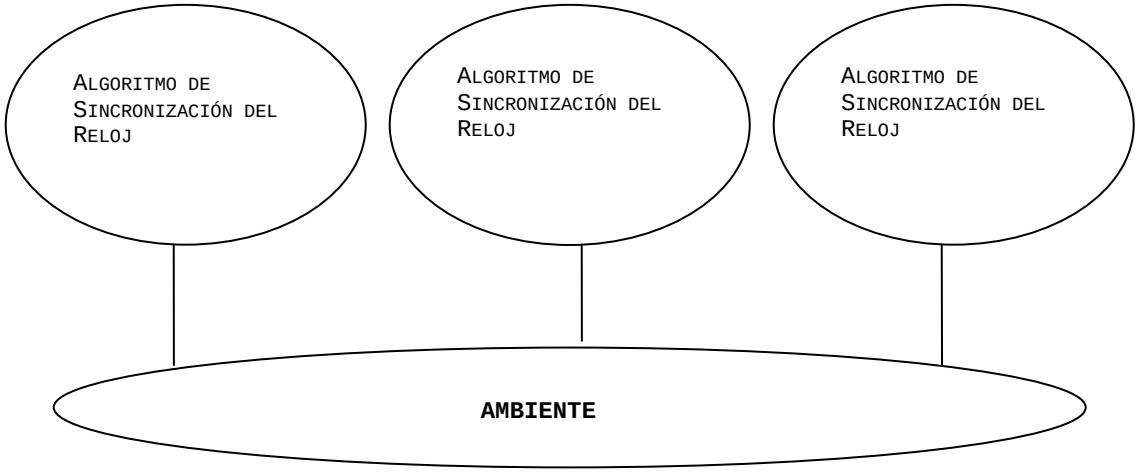


FIGURA 2-1. EL ARREGLO CONCEPTUAL DEL AMBIENTE EN UN SISTEMA DE SINCRONIZACIÓN DEL RELOJ.

2.2 Patrón

Un patrón es la ejecución de un ambiente, una gráfica dirigida que la describe representando cada evento como un punto *p*, y considerando que para cada punto existe un tiempo real y local de la ocurrencia. Un patrón puede describirse como (G, local_time, real_time), donde (G=(V,E), local_time) se refiere a una vista real_time muestra el tiempo real de p.

2.3 Vista

Una vista es un patrón que no considera el atributo del tiempo real para los puntos. La vista de las ejecuciones de los ambientes contiene información que usa cada ASR (Algoritmo de Sincronización del Reloj) para calcular la latencia, mientras que la información del tiempo real en los patrones se encuentra disponible únicamente para el análisis de los eventos que intervienen en la sincronización del reloj. Una vista se representa de la forma (G, local_time), donde G=(V,E) se trata de una gráfica dirigida. Cada punto $p \varepsilon V$ muestra cualquier acción de un automata o un punto nulo que se dice que ocurre en el mismo procesador. Local_time es un mapeo desde el punto de *V* hacia *R*, para cualquier punto $p \varepsilon V$, a local_time(p) se conoce como el tiempo local de *p*.

A continuación se presenta un ejemplo de un escenario donde se presentan eventos de envíos de mensajes, además es posible realizar un mapeo hacia una vista y posteriormente hacia un patrón.

2.4 Ejemplo de un escenario

Se tiene un sistema que consiste de dos procesadores *S* y *V* conectados en un enlace de comunicación bidireccional. En la figura 2-2 A), se cuenta con un diagrama de espacio-tiempo que representa al siguiente escenario. En el tiempo real 0, el procesador *V*, cuyo reloj local muestra -1, envía un mensaje *M* a *S*; el procesador *S* recibe *M* en el tiempo real 2, cuando éste muestra en su reloj local 1. Algunos eventos distinguibles ocurren en *S*, en el tiempo real 2.5, cuando el reloj local muestra 2. (A estos eventos distinguibles se les conoce como un evento interno). En el tiempo real 3.5, cuando el reloj local *S* está en 3, *S* envía un mensaje de *M'* a *V*; *M'* se recibe en *V* en el tiempo real 9 cuando la lectura del reloj local es 8.

En la figura 2-2 B), se ilustra el patrón basado en este escenario, con un punto nulo para los eventos distinguibles. Si se quitan todos los atributos llamados "tiempo real", entonces este patrón se convierte en vista.

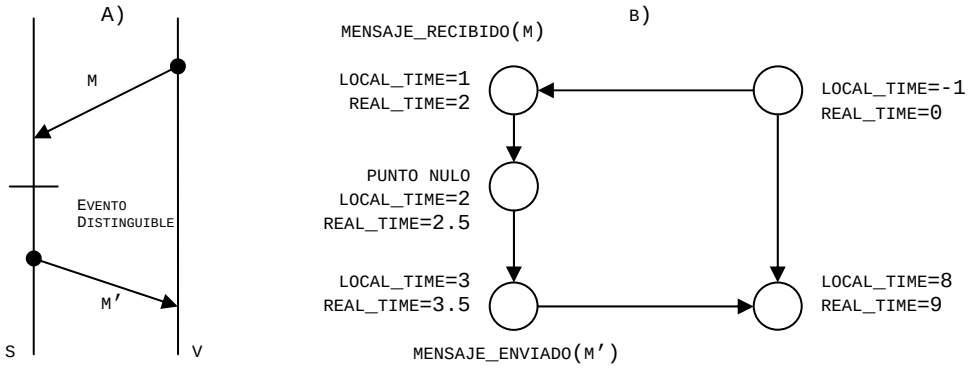


FIGURA 2-2. EJEMPLO DE UN ESCENARIO A) Y DE UN PATRÓN B)

Observando el escenario, existe una serie de variables que se deben considerar. En la línea del tiempo del procesador V, existen dos índices de progreso, llamados en la literatura límites de desvío del reloj, denominados $[\underline{\rho}, \overline{\rho}]$. Esto es análogo en la línea del tiempo del procesador *S,* si estos límites son iguales ($\underline{\rho} = \overline{\rho} = 1$), entonces se dice que es un reloj sin desvíos (perfecto). El índice inferior es el tiempo mínimo que debe de realizar la ejecución del proceso denominado $\underline{\rho}$. El índice superior es el tiempo máximo que debe realizar la ejecución del proceso denominado $\overline{\rho}$. Para mayor referencia a este valor, consultar la página 19.

Cuando se envía un mensaje, existen límites en el tiempo de tránsito, a estos límites se les denomina *límites de latencia del mensaje* y se representan como L(M) y H(M). El índice inferior de latencia del mensaje L(M) es el tiempo mínimo en que deben llegar los mensajes M y M'. El índice superior de latencia del mensaje H(M) es el tiempo máximo en el que puede llegar el mensaje.

Considerando el ejemplo anterior se agregan ahora los requerimientos para los límites de desvío y la latencia del mensaje.

El procesador *V* no tiene límite de desvío ($\rho[1,1]$) y *S* tiene un desvío ρ de [0.5, 1.5]. El mensaje *M* se tarda no menos de 2 unidades y no más de 3 unidades [2,3]. El mensaje *M'* se tarda no menos de 5 unidades y no hay límite de transmisión [$5, \infty$).

El nuevo diagrama de espacio-tiempo con los límites considerados, se encuentran en la figura 2-3, en el se muestran algunos tiempos en los límites de desvío y en la latencia del mensaje, también se observa el tiempo de un viaje redondo vía *M* y *M'*.

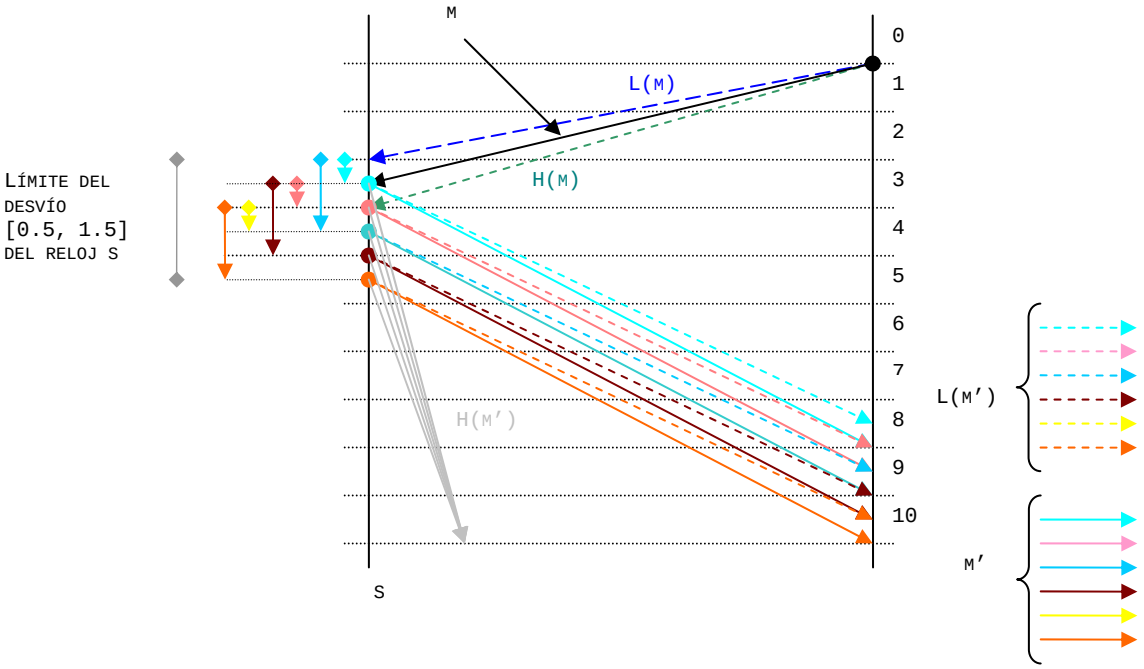


FIGURA 2-3. DIAGRAMA ESPACIO-TIEMPO DE UN VIAJE REDONDO

Observando la figura 2-3, se aprecia que hay una serie de intervalos a considerar, incluyendo el punto nulo que no se toma en cuenta para la ejecución del escenario, por esta razón, se realizan dos ejecuciones. La primera es el mejor caso en todos los intervalos (límites de desvío y latencia del mensaje) y la segunda es el peor caso, con estas dos ejecuciones se sabe cuál es el mejor y peor tiempo que se puede obtener.

Se observa en la figura 2-4, el rango de tiempo de ejecución que usa el escenario y éste se encuentra dentro del intervalo $[8, \infty)$. En el patrón de la figura 2-2, el tiempo de ejecución es de 9, un valor dentro del rango obtenido, lo cual proporciona un resultado veraz.

Una vez explicados los eventos que representan en una ejecución, se mencionan más conceptos para su aplicación en las gráficas de sincronización.

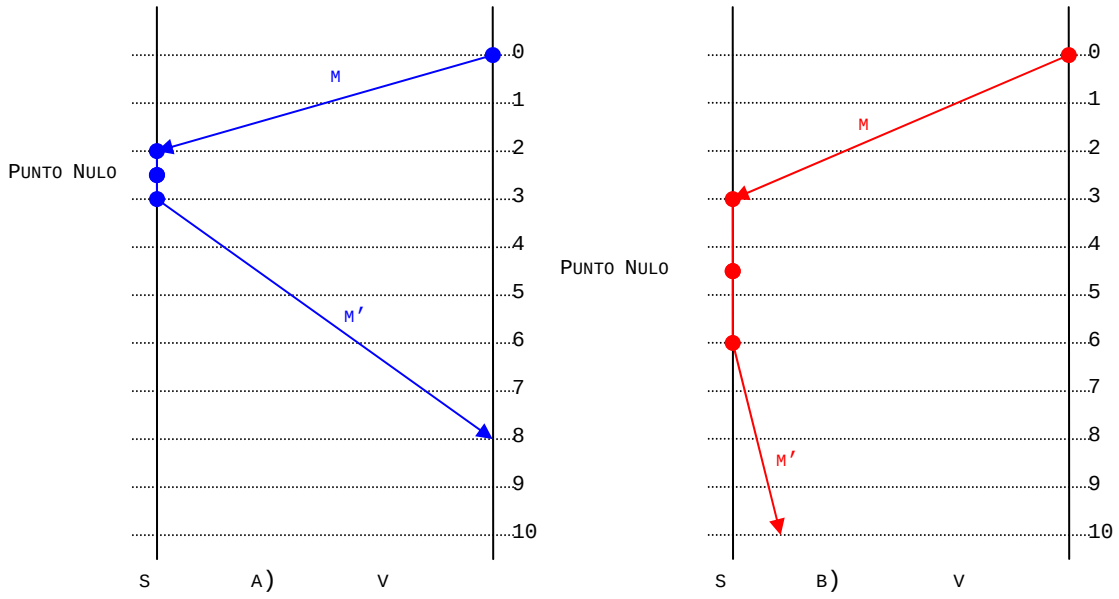


FIGURA 2-4. ESCENARIO, A)EN EL MEJOR Y B) PEOR CASO

2.5 Retraso Actual y Virtual

Sean *p* y *q* dos puntos de un patrón *P*, el retraso actual en *P* que es relativo de *p* a *q* y el retraso virtual en *P* que es relativo de *p* a *q* se definen como:

$$act\_del_P(p,q)=real\_time_P(p)-real\_time_P(q)$$
$$virt\_del_P(p,q)=local\_time_P(p)-local\_time_P(q)$$

Dos puntos *p*, *q* en una vista dada V=(G,local_time) son llamados puntos adyacentes si hay un arco dirigido entre ellos en *G*.

Un mapeo de límites para una vista V es una función B que mapea cada par de puntos *p,q* en puntos adyacentes en V, en un número tal que $-\infty < B(p,q) \leq \infty$. Un patrón con una vista V se dice que satisface B si para todo par de puntos adyacentes *p,q* se tiene $act\_del(p,q) \leq B(p,q)$.

Para un mensaje *m* con punto de envío p, punto de recepción q, donde los límites de latencia son L(m) y H(m) se tiene:

$$B(q,p)=H(m) \quad y \quad B(p,q)=-L(m)$$

Dado que *p* sea el predecesor inmediato de *q* en un procesador con reloj ($\underline{\rho}$, $\overline{\rho}$). Entonces:

$$B(q,p)=virt\_del(q,p)/\underline{\rho} \quad y \quad B(p,q)=virt\_del(p,q)/\overline{\rho}$$

Con este concepto todos los eventos que se ejecuten en un escenario se van a normalizar y la función a utilizar dependerá en dónde se encuentra el mensaje en la ejecución de la sincronización del reloj.

Ahora se tomará el problema fundamental para resolverlo con la gráfica de sincronización. Dados dos puntos en una ejecución del sistema de sincronización del reloj, encontrar el límite más estrecho en el tiempo real que pasa entre sus ocurrencias.

2.6 Gráfica de Sincronización

La gráfica de sincronización $\Gamma_{\beta B}=(V,E,w)$ de una vista y un mapeo de límites de B que se define por la gráfica $\Gamma_{\beta}=(V,E)$ con una función de peso w, donde para cada par de puntos $(p,q) \varepsilon E$

$$w(p,q)=B(p,q)- virt\_del(p,q)$$

La gráfica de sincronización se puede obtener mediante gráficas de tipo V más la función de mapeo de límites *B*. Las gráficas tipo *V* y las gráficas tipo *P* son las abstracciones de las vistas y patrones respectivamente.

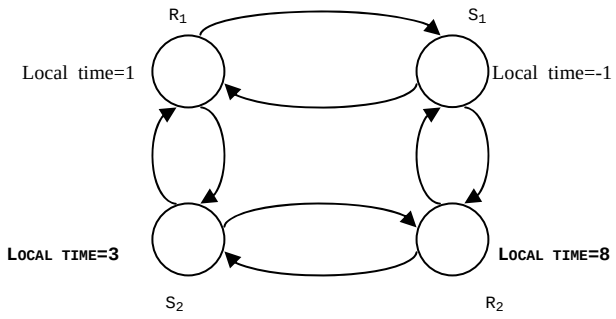


FIGURA 2-5. UN EJEMPLO DE GRÁFICA TIPO V

2.7 Generación de la Gráfica de Sincronización

Para generar una gráfica de sincronización se deben considerar varios puntos:

- Crear las gráficas de tipo V.
- Mapeo de límites.

Cuando se obtienen esos elementos se realizan las gráficas:

- Tipo P.
- Compensaciones (offset).

Con todos esos elementos se puede obtener la gráfica de sincronización y su interpretación. Para que el lector se familiarice con los conceptos y las gráficas, se agrega un ejemplo que presenta paso a paso la forma para obtener la gráfica de sincronización. Este ejemplo se encuentra en el

apéndice F y las pruebas y demostraciones de los teoremas mencionados se pueden consultar en [5].

Gráfica de Tipo P. Se compone de una *vista* más los valores de cada uno de los puntos de p $\in$ V, éstos son los valores en tiempo real. En las gráficas de tipo P, se define el concepto clave denominado compensación (offset).

Sea *p* un punto en la gráfica tipo *P* donde se encuentra definida como α = (G, local_time, real_time). La compensación absoluta de p es:

$$\delta\alpha(p)=real\_time\alpha(p)-local\_time\alpha(p)$$

Además la compensación relativa de p hacia q es:

$$\delta\alpha(p,q)=\delta\alpha(p)-\delta\alpha(q)$$

Definición de la gráfica de sincronización. Está se genera a partir de las gráficas de tipo V y de los mapeos de límite B Se trata de una gráfica dirigida con peso Γ=(V,E,w), donde G=(V,E) y el peso se representa por:

$$w(p,q)=B(p,q)- virt\_del(p,q) \text{ para todo } (p,q) \in E.$$

Al obtener la gráfica de sincronización y de la compensación (anexo F), se aprecia la relación entre las gráficas de sincronización y las gráficas de tipo P. Todo esto se basa en el siguiente lema obtenido de [5].

Lema 2.1 Si una gráfica de tipo P con una gráfica de tipo V satisface B, entonces $\delta(p,q) \in [-w(p,q), w(p,q)]$.

El concepto de distancia entre los puntos de las gráficas de sincronización. Se puede ver utilizando la siguiente definición.

Definición 2.1 El peso de una ruta $\theta=p_0,p_1,...,p_k$ en una gráfica pesada Γ=(V,E,w) es $w(\theta)=\sum_{i=1}^{k} w(p_{i-1},p_i)$. Una ruta de *p* a *q* es la ruta mas corta si este peso es mínimo entre todas las rutas de *p* hacia *q*. La distancia de *p* a *q*, denotada como d(p,q), es el peso de una ruta más corta de *p* hacia *q* o ∞ si éste no tiene una ruta de *p* a *q*.

Nótese que las distancias no están bien definidas, si Γ tiene ciclos con pesos negativos da una suficiente condición para que Γ no tenga ningún ciclo con peso negativo.

Lema 2.2 Si existe una gráfica de tipo P con el gráfico de tipo V tal que satisface B, entonces Γ tiene ciclos con peso no negativo.

Teorema 2.1 Una gráfica de tipo P con una gráfica de tipo V satisfacen B sí y sólo sí para cualquier par de puntos p,q ε V en la gráfica de sincronización $\delta(p,q) \leq d(p,q)$.

Con esto, se puede obtener la distancia más corta entre un evento y otro y así conocer los arcos que hay que considerar.

Éste es el punto de vista clásico, se asumen o se consideran conceptos que se ejecutan de manera inmediata, como son:

- Cuando un mensaje llega, se procesa inmediatamente.
- Cuando un mensaje es procesado, se envía inmediatamente.
- No existen procesos que interfieran con el procesamiento de los eventos.
- El calendarizador (scheduler) funciona adecuadamente.

En las investigaciones se modelan estos problemas con más variables internas del sistema operativo y se muestra cómo se comportan en la realidad. El resultado de esto se le llama modelo del sistema en tiempo real.

Al conocer una de las soluciones de manera abstracta, la solución es viable y teóricamente funciona bien, pero al realizar el mapeo a tiempo real se cuentan con variables y restricciones que no se han analizado en el caso abstacto (o modelo clásico) y repercute en los tiempos ya estipulados en otras investigaciones.

.

Capítulo 3

Modelo del Sistema en Tiempo Real

En este capítulo se realiza una breve explicación de los modelos del sistema en tiempo real. En la sección 3.1 se realizan comparaciones entre modelos clásicos y modelos en tiempo real presentando sus características, ventajas y consideraciones.

La sección 3.2 muestra el modelo teórico y formal del modelo computacional en tiempo real, tambien se describe un ejemplo que indica cómo se efectúa un envío de mensaje y cuáles son las regiones por donde pasa, además se agregan definiciones. La sección 3.3 efectúa una ejecución y presenta las variables que la integran, así como las condiciones que debe cumplir el algoritmo.

En la sección 3.4 se describe un ejemplo de la ejecución en tiempo real y las condiciones que deben cumplirse en el modelo teórico. Las secciones 3.5 y 3.6 muestran el problema de la sincronización del reloj para el cómputo distribuido y cómo se comporta con las tres propiedades (terminación, acuerdo y validez) para demostrar que dicho algoritmo cumple para ser distribuido; junto con los teoremas también se describen lemas y demostraciones formales del algoritmo de sincronización del reloj mediante envío de mensajes broadcast y multicast, encontrando un límite superior. Por último se lista la relación que tienen los sistemas en tiempo real y los sistemas clásicos mediante el uso de transformaciones para ambos.

3.1 Sistemas en Tiempo Real

Las ejecuciones de los algoritmos distribuidos son típicamente un modelo de secuencias, pasos atómicos computacionales que se ejecutan en tiempo cero (inmediatamente). Con esta asunción, no se hace la diferencia, por ejemplo, si los mensajes llegan a un procesador simultáneamente o en un tiempo considerable.

Los mensajes se procesan instantáneamente cuando éstos llegan. La abstracción denominada el paso del tiempo por cero es por lo tanto muy conveniente para el análisis y existe una abundancia de algoritmos distribuidos. El inconveniente que imposibilita el resultado se debe a que los límites inferiores estuvieron pensados para modelos que emplean la suposición de mensajes procesados instantáneamente.

En sistemas reales, sin embargo, los pasos computacionales no son instantáneos ni arbitrarios (preemptable) [página 8], ya que un paso computacional se acciona por un mensaje que llega a la mitad de una ejecución de algún otro paso computacional y usualmente se retrasa hasta que el actual cálculo se termina. Estos resultados se presentan debido al fenómeno de colas, el cual depende no tanto del actual mensaje de patrón recibido como de la disciplina de cola/calendarización (scheduler).

La comunidad en sistemas de tiempo real estableció técnicas de gran alcance para analizar dichos efectos [10], es tal el resultado del peor

caso en tiempo de respuesta y retrasos en los extremos (*end-to-end*) que puede calcularse.

Los modelos en cómputo el paso del tiempo por cero tienen buena cobertura en los sistemas donde el retardo de los mensajes es mucho más alto que el tiempo de procesamiento del mensaje. Ejemplos de algunos usos para estos modelos son las redes de alta velocidad, sin embargo, no es el caso en este estudio.

Es más importante asumir el paso de los mensajes por el tiempo cero son inevitables y se ignora la cola del receptor. Es posible, incluso en el caso donde el mensaje tiene un gran retardo y que múltiples mensajes lleguen al receptor al mismo tiempo. Esto causa que los procesos de algunos de estos mensajes se retrasen hasta que el procesador esté disponible (*idle)* otra vez.

La práctica común para medir el retraso de cola es contar con un contador que se incrementa hasta límite superior $\underline{\delta}^{+}$ en el mensaje retrasado. Este acercamiento, sin embargo, tiene dos desventajas:

Primera, la información conocida por el patrón del mensaje del algoritmo es necesaria para determinar un parámetro del modelo del sistema, que crea dependencias cíclicas.

Segunda, en pruebas de un límite inferior, el usuario puede elegir un mensaje de retardo arbitrario dentro del intervalo $[\underline{\delta}^{-}, \underline{\delta}^{+}]$, incluso si esta opción no está de acuerdo, es decir, no es posible, con el patrón actual de la llegada del mensaje. Esto podía conducir a los límites más bajos excesivamente pesimistas.

El modelo subyacente asume etapas de tiempo diferentes de cero, y se considera suficientemente pequeña la etapa para ignorarse completamente por los efectos de cola. Por otra parte, en contraste, se restringe su atención a los "problemas internos" (esencialmente que corresponden a las propiedades de trazo) solamente.

Existe otro ejemplo del modelo del paso de tiempo diferente de cero llamado “referencia de memoria remota” (RMR) para los sistemas de memoria compartida de Anderson [11,12]. Se asume tiempo de cómputo, que depende del número de conflictos de accesos en la memoria compartida. El modelo RMR se emplea como base para derivar varios algoritmos.

Existen otras ramas de investigación donde el cómputo distribuido y los sistemas en tiempo real se presentan y están combinados en ciertos modelos [13], tales marcos permiten modelar formalmente un análisis de complejidad de los sistemas en tiempo real distribuidos.

Un ejemplo representativo son los autómatas de tiempo de I/O (TIOA) [14], el cual puede cambiar el estado vía transiciones ordinarias discretas y trayectorias continuas. TIOAs facilita la composición jerárquica, la abstracción y las pruebas de las siguientes propiedades: seguridad y viveza.

Sin embargo, ninguno de los marcos modelados soporta tiempos diferentes de cero y análisis de calendarización en tiempo real de algoritmos distribuidos.

3.2 Modelo Computacional en Tiempo Real

El modelo computacional en tiempo real es igual al modelo computacional clásico, a excepción del siguiente cambio:

Un paso de cómputo en un sistema en tiempo real se ejecuta dentro de un límite a lo ancho del sistema, estos límites son el inferior μ^- y el superior μ^+. Se permite el tiempo de procesamiento y por lo tanto los límites $[\mu^-,\mu^+]$. Dependen del número de mensajes enviados en un paso de cómputo. Para distinguir claramente un paso computacional en tiempo real (llamado acción) del modelo clásico, se utiliza el término trabajo para hacer referencia en [6].

En particular, los efectos que hacen la cola y la calendarización deben considerar lo siguiente:

- Ahora se deben distinguir dos modalidades del procesador en cualquier punto durante el tiempo real *t*: *idle* y *ocupado* (es decir, cuando se encuentran ejecutando un trabajo). Puesto que los pasos que se computan no pueden interrumpirse, es necesaria una cola para almacenar los mensajes ordinarios y de tiempo que llegan mientras el procesador está ocupado. Se asume que los mensajes se almacenan en la cola en el orden en el cual van llegado.

- Cuándo y cuál es el orden de los mensajes que se toman en la cola donde son procesados y están especificados por una cierta política de calendarización, que es en general, independiente al algoritmo. Formalmente, una política de calendarización se especifica como un mapeo arbitrario del estado actual de la cola (igual a una secuencia de mensajes), de la lectura del reloj en hardware y del estado del procesador local sobre un solo mensaje de esa secuencia. La política de calendarización se utiliza para seleccionar un nuevo mensaje de la cola que haya terminado un proceso de el trabajo [6].

Se asume en la tesis que la política de calendarización es *nonidling*:

Cuando el procesador está en *idle*, el proceso de un mensaje entrante comienza inmediatamente. Cuando el procesador acaba un trabajo y la cola no está vacía, un mensaje de la cola se toma y el proceso correspondiente al trabajo inicia sin mayor retraso. Se ejecuta lo siguiente:

- El retraso de un mensaje se mide en tiempo real desde el comienzo del trabajo que envía el mensaje en tiempo real hasta la llegada del procesador destino (donde el mensaje queda en la cola o, si el procesador está en *idle*, inicia inmediatamente el trabajo correspondiente). Como en el modelo de cómputo clásico, el retraso del mensaje por parte de los mensajes ordinarios, deben de estar dentro de un límite inferior al ancho sistema $\underline{\delta}^-$ y de un límite superior $\underline{\delta}^+$ si el mensaje se retrasa y por lo tanto los límites $[\underline{\delta}^-, \underline{\delta}^+]$ pueden depender otra vez del número de mensajes enviados y el trabajo que se envía.

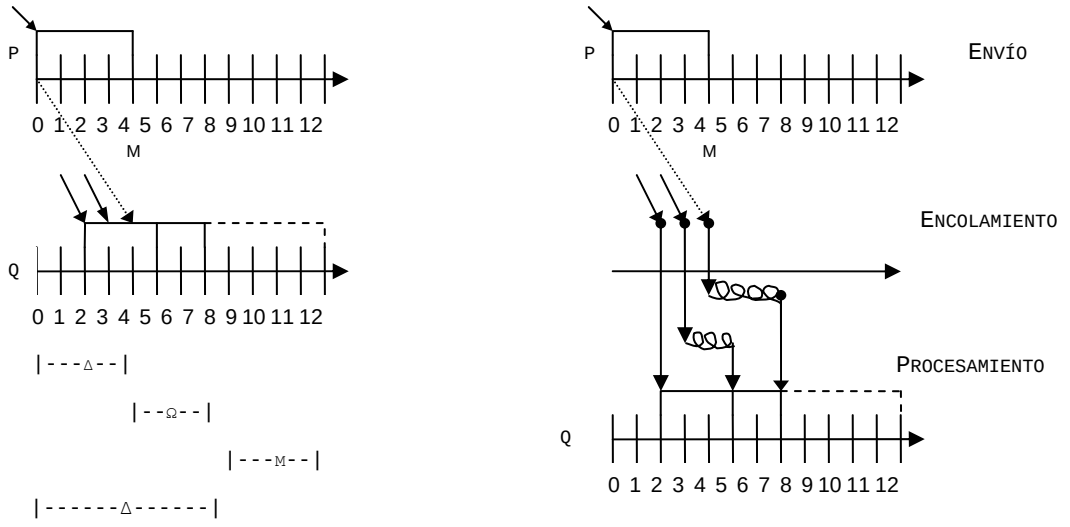
- Se asume que el reloj en hardware se puede leer solamente al principio de un trabajo. Esta restricción conjunta con la definición de mensajes retrasados, permitirá que se defina funciones de transición exactamente de la misma manera que en el modelo de cómputo clásico. Después de todo, la función de transición apenas define la semántica lógica de una transición, pero no su sincronización.

- Contrario al modelo de cómputo clásico, las transiciones del estado de la forma *nuevoEstado* → , ... , → *viejoEstado* en un solo paso de cómputo no necesitan suceder al mismo tiempo, ya que Típicamente, ocurren en diferentes tiempos en el mismo trabajo, permitiendo que un estado intermedio sea válido en un procesador para una cierta duración diferente a cero [6].

La figura 3-1 representa el ejemplo de un solo trabajo en el procesador *p* que envía un mensaje *m* al receptor *q* quién está actualmente ocupado en el proceso de otro mensaje.

La parte A) muestra los parámetros de sincronización relacionados con el modelo de cómputo en tiempo real, se sabe el retraso del mensaje (δ), el retraso de la cola (w), el retraso end-to-end ($\Delta = \delta + w$) y el retraso por procesamiento (μ) para un mensaje m que está representado por la flecha rayada. Los límites en el retraso del mensaje (δ) y el retraso por procesamiento (μ) son parte del modelo del sistema, lo que permite que no necesariamente se conozca el algoritmo.

Los límites en la cola del retardo (w) y el retardo end-to-end (Δ) no son parámetros del modelo del sistema en contraste como modelo de cómputo clásico, donde el retraso end-to-end iguala siempre al mensaje de retraso. Estos límites (si existen) se deben derivar de los parámetros del sistema ($n,[\delta^-, \delta^+],[\mu^-, \mu^+]$) y del patrón del mensaje del algoritmo, realizando un análisis a la calendarización de tiempo real. En B) se muestra la relación detallada entre la llegada del mensaje (la cola) y el procesamiento del mensaje actual.



A) PARÁMETROS EN TIEMPO PARA ALGÚN MENSAJE M

B) RELACIÓN DEL MENSAJE QUE LLEGA Y EL MENSAJE PROCESADO, CON ENCOLAMIENTO MOSTRADO EXPLÍCITAMENTE

FIGURA 3-1. MODELO COMPUTACIONAL EN TIEMPO REAL

3.3 Ejecuciones en tiempo real

Se formaliza la notación de una ejecución en tiempo real (rt-run), que corresponde a una ejecución en el modelo computacional clásico. Un rt-run es justo una secuencia que reciben eventos y trabajos.

Un evento de recepción *R* para un mensaje que llega en *p* en tiempo real *t* es una triple consistencia con base en el índice del procesador proc(R) = *p*, el mensaje msg(R) y la llegada en tiempo real time(R) = t. Es *t* el tiempo que está en cola en la figura 3-1(B).

Un trabajo *J* que inicia en el tiempo *t* en *p, que* es una 6-tupla conformada por:

- El índice del procesador proc(J) = p.
- El mensaje procesado msg(J).
- El tiempo de inicio begin(J) = t.
- El tiempo de procesamiento del trabajo d(J).
- La lectura del reloj en hardware HC(J) = HCp(t), y
- La secuencia de transición del estado trans(J) = (oldstate ... newstate).

States(J), sent(J), oldstate(J) y newstate(J) son abreviaturas para especificar las distintas partes de trans(J) y se encuentran definidas para hacerlas análogas al modelo del cómputo clásico. Dado end(J) = begin(J) + d(J).

La figura 3-1 proporciona un ejemplo de rt-run, conteniendo tres receptores de eventos y tres trabajos en el segundo procesador. Por ejemplo, el trabajo con líneas punteadas en el segundo procesador q se encuentra conformada por (q, m, 7, 5, HCq(7), [*oldstate*, … … , newstate]), con *m* siendo el mensaje recibido durante el evento de la recepción (q, m, 4). Hay que notar que ni los actuales tiempos de transición de estado ni los actuales tiempos de los envíos de mensajes actuales están registrados en un trabajo. Medir todo el retardo del mensaje de inicio del trabajo y saber que los estados de transición y los mensajes enviados ocurren en el orden mencionado en tiempos arbitrarios, es suficiente para probar que rt-run satisface un sistema dado y un conjunto de propiedades durante el trabajo, y que éstos son los adecuados para realizar el análisis en cuanto a la complejidad del tiempo [6].

Rt-run de un cierto algoritmo *A* debe satisfacer las siguientes propiedades:

- Consistencia Local:

  Todos los estados de transición y los mensajes enviados deben ser constantes con la función de la transición definida en *A*. Como en el modelo de cómputo clásico, el oldstate del primer trabajo en cada procesador debe ser un cierto estado inicial $Istate^{ru}_{p.}$

  El reloj en hardware debe ejecutarse con la tasa idéntica en tiempo real. Ejemplo. HC(j)- begin(j) debe ser equivalente al de todos los trabajos J en el mismo procesador.

  Los trabajos en el mismo procesador no debe traslaparse esto es, no deben ser dos trabajos J, J’ con proc(J) = proc(J’) y begin(J) ≤ begin(J’) < end(J).

  Si un mensaje de tiempo (TIMER) se envía para la recepción del tiempo T y este llega cuando el reloj en hardware lee T, ejemplo: esto es un evento de recepción R con $HC_p(time(R))=T$.

  Calendarización (scheduling) no debe ser idle (ociosa).

- Consistencia Global: Cada mensaje se envía, recibe y procesa solamente una vez (excepto para mensajes init).

Un procesador p está ocupado en el tiempo t si se encuentra realizando algún trabajo J tal que begin(J) ≤ t < end(J); en cualquier otro caso está desocupado [6].

3.4 Sistemas y ejecuciones admisibles en tiempo real

Un sistema en tiempo real s está definido por un entero n y dos intervalos [δ^-, δ^+] y [μ^-, μ^+]. Considerando a δ-, δ+ μ-, μ^+ como constantes y dando una ventaja injusta a los algoritmos basados en difusión (broadcast) cuando se comparen algunos algoritmos en tiempos de complejidad donde los pasos computacionales toman un tiempo entre μ^- y μ^+, independientemente del número de mensajes enviados. Esto hace imposible derivar un límite inferior de la complejidad significativa del tiempo para que los sistemas en los cuales usen difusión (broadcast) en tiempo constante no esté disponible.

Por lo tanto, los límites del intervalo δ^-, δ^+ y μ^-, μ^+ pueden ser cualquier constante o función no decrecientes, considerando que {0,...,n-1} → R^+, representando un mapeo al número de procesadores destino donde los mensajes ordinarios son enviados durante los pasos computacionales hacia el mensaje actual o al límite del retraso del procesamiento.

Ejemplo: Durante algún trabajo, un solo mensaje es enviado a tres procesadores. La duración de este trabajo miente dentro del intervalo $[\mu_{(3)}^-, \mu_{(3)}^+]$. Cada uno de estos mensajes tiene un retraso del mensaje entre $\delta_{(3)}^-$ y $\delta_{(3)}^+$, los retrasos de los tres mensajes no necesitan ser los mismos.

Enviar L mensajes inmediatamente no debe ser más costoso que enviar esos mensajes en múltiples pasos. Además, se asume que el retraso del mensaje tiene incertidumbre $\varepsilon_{(\ell)} := \delta_{(\ell)}^- - \delta_{(\ell)}^+$ y que está debe ser también no decreciente y, por lo tanto, $\varepsilon_{(1)}$ es la mínima incertidumbre. Esta asunción es razonable porque generalmente se envían más mensajes aumentando la incertidumbre en lugar de disminuirla.

Definición 3.1. Dado que S=(n, [δ^-,δ^+],[μ^-, μ^+]) es un sistema en tiempo real. rt-run es s-admisible, si rt-run contiene exactamente n procesadores y satisface las siguientes características de la sincronización. Si L es el número de mensajes enviados durante un cierto trabajo J:

1. El retraso del mensaje (medido para begin(J) en el correspondiente evento receptor) en sent(J) debe estar dentro del intervalo [δ^-(l), δ^+(l)].

2. La duración del trabajo d(J) debe estar dentro del rango $[\mu_{(\ell)}^-, \mu_{(\ell)}^+]$

3. El orden de recepción de eventos y trabajos capturan causalidad.

Capturar causalidad significa que:

a) Si un trabajo J’ ocurre después del trabajo J en el mismo procesador => J<J’.

b) Si un trabajo J se autoenvía un mensaje recibido por algún trabajo J'=> J<J' y

c) Si recibe el evento R algún mensaje procesado por algún trabajo J => R < J.

También se requiere contar con rt-run, es decir, la secuencia de eventos recibidos y trabajos, se ordene por el tiempo de ocurrencia de eventos recibidos y tiempos de inicio de los trabajos [6]. Considerando el siguiente ejemplo:

- Para cada procesador, se elige un estado inicial del conjunto proporcionado de alguna manera, un valor inicial del reloj en hardware y el tiempo en que cual llegará el mensaje init.

- Para cada procesador, se escoge un valor $[\delta^-,\delta^+]$ representando la suma de:

  a) El tiempo entre el inicio del trabajo cuando se envían los mensajes y el tiempo actual del envío de mensajes

  b) El retardo actual de transmisión de los mensajes (incluyendo los eventos ocurridos que se recibieron).

  c) Para cada trabajo, escoge un valor en el intervalo $[\mu^-, \mu^+]$ para el tiempo de procesamiento y cualquier gasto indirecto asociado (calendarización, entre otros).

  d) Define la política de calendarización.

3.5 Problemas, Algoritmos y Pruebas

Un problema se representa como un juego de propiedades *trace* que deben satisfacer la ejecución o rt-run (real-time run) y un conjunto de propiedades de estado, para todas las trayectorias de estado posibles a la ejecución o rt-run. De las siguientes definiciones se asegurarán aquellos problemas específicos para el modelo computacional clásico.

3.5.1 Eventos y Propiedades de Estado

Definición 3.2. El evento *e* corresponde a una acción *ac* o a un trabajo J, es 4-tupla que consiste en un índice del procesador *proc(e) = proc(ac)/proc(J),* la ocurrencia de inicio del tiempo real *begin(e) = time(ac)=begin(J),* el valor del reloj en hardware *HC(e) = HC(ac)=HC(J),* y el antiguo valor local *oldstate(e) = oldstate(ac)/oldstate(J)*.

El evento trace de alguna ejecución rt-run es apenas la secuencia de eventos correspondientes a algún subconjunto relevante (las acciones externas) de las acciones o trabajos. Una propiedad *trace* usualmente se caracteriza por un predicado *E* accionado por eventos *traces*. En algunos casos, es más conveniente especificar una propiedad como un predicado en los estados globales preferentes que en los eventos. Una propiedad de estado es un conjunto de trayectorias de estado.

Un problema es una colección de propiedades *trace* y/o propiedades de estado. Se dice que una ejecución/rt-run satisface un problema si todas las propiedades *trace* satisfacen a toda la correspondencia de trayectoria de estados globales.

Definición 3.3. (El problema de la terminación de la sincronización del reloj). Dado el tiempo de ajuste en tiempo real *t* en el procesador *p,* se define como $HC_p(T) + state_p(t).adj$, donde el valor de ajuste local *adj* se puede cambiar en el procesador (ya que forma parte del estado local). $state_p(t).terminated$ es cierta (trace) cuando el algoritmo ha terminado en el procesador *p* por tiempo *t*. El problema de sincronización del reloj que termina con precisión γ está especificado por las siguientes dos propiedades del estado [7]:

Terminación. $\exists t_{term}$: para todo $t \geq e_{term}$ para todo p: $state_p(t).terminated$ = true, es decir, todos los procesadores terminan en algún tiempo real finito t_{term}.

Acuerdo. Para todo t: (para todo p: $state_p(t).terminated$ = true) => para todo p,q: $| HC_p(t) + state_p(t).adj - (HC_q(t) + state_p(t).adj)| \leq \gamma$, es decir, después de t_{term}, todos los procesadores han ajustado sus relojes dentro del rango γ.

Desafortunadamente, en el modelo de cómputo en tiempo real, el tratamiento de propiedades de estado es considerablemente más complejo que el tratamiento de propiedades *trace*, al tomar en cuenta el problema de la terminación de la sincronización del reloj, por ejemplo:

Una vez que $state_p(T).terminated$=true, se termina (en tiempo cero) la acción, la cual ocurre en el modelo de cómputo clásico, pero no en el modelo de tiempo real. Para asegurar la garantía de terminación del trabajo en el modelo computacional en tiempo real es determinar los conjuntos de las trayectorias de estado y de ahí el t_{term} del tiempo real y con esto se demuestra que son diferentes ambos modelos.

Afortunadamente, si se asumen las ejecuciones/rt-run como muchas propiedades pueden especificarse como propiedades de eventos o estados, es decir, cada procesador ejecuta un número de no operaciones de acciones/trabajos después de la finalización y el problema de la terminación de sincronización del reloj puede representarse unicamente como propiedades vía *trace [6]*.

Definición 3.4. El problema de terminación de sincronización de reloj, propiedades *trace*.

Terminación

e_{term}: para todo $e \geq e_{term}$: oldstate(e).terminated = true

Acuerdo

$$\forall e_1, e_2 : oldstate(e).ter\min ated = true) => | HC(e_1) + oldstate(e_1).adj - begin(e_1) - (HC(e_2) + oldstate(e_2).adj - begin(e_2)) | \leq \gamma$$

La definición 3.3 requiere que después de la terminación se realicen las compensaciones del tiempo real (= adj.time – tiempo real) de dos eventos que estén dentro de γ.

3.5.2 Transformaciones

Se describirá como en el modelo computacional clásico y el modelo computacional en tiempo real que son claramente equivalentes desde la perspectiva de la solución de problemas: Un sistema en tiempo real puede simular algún sistema clásico en particular (y viceversa) y las condiciones para transformar un algoritmo A de un modelo computacional clásico en un algoritmo S_A del modelo computacional en tiempo real (y viceversa). Como consecuencia, se puede traducir una cierta imposibilidad y los resultados de los límites inferiores.

Se simula un sistema en tiempo real ($n,[\delta^+ = \underline{\delta}^-, \delta^+ = \underline{\delta}^+],[\mu^- = \underline{\mu}, \mu^+ = \underline{\mu}]$) sobre de un sistema clásico ($n,[\underline{\delta}^-, \underline{\delta}^+]$), esto se logra de manera directa y es suficiente para implementar un procesamiento de retraso artificial μ, la cola de mensajes se manipula durante una simulación de trabajo y el programador de calendarización (scheduler). Estas simulaciones permiten ejecutar cualquier algoritmo A en el modelo computacional en tiempo real diseñado para un sistema ($n,[\delta^-, \delta^+],[\mu^-, \mu^+]$) con $\delta^- \leq \underline{\delta}^-$, $\delta^+ \geq \underline{\delta}^+$ y $\mu^- \leq \underline{\mu} \leq \mu^+$ sobre él, esto da como resultado un algoritmo correcto en el modelo clásico computacional [6].

La otra simulación es de un sistema clásico ($n,[\underline{\delta}^- = \Delta^-, \underline{\delta}^+ = \Delta^+]$) y está sobre un sistema en tiempo real ($n,[\delta^-, \delta^+],[\mu^-, \mu^+]$), aquí es más engañoso. Primero la clase del algoritmo A del modelo clásico computacional que se puede transformar en un algoritmo S_A del modelo computacional en tiempo real debe estar restringido. Segundo y más importante, un análisis programado en tiempo real debe conducirse en orden para romper la dependencia circular del algoritmo A y simular el retraso end-to-end $\Delta \in [\Delta^-, \Delta^+]$, esta normalmente se encuentra oculta en el modelo clásico computacional pero aparece cuando uno trata de instanciar en el modelo de un sistema real.

Se describe cómo simular un sistema clásico ($n,[\underline{\delta}^- = \Delta^-, \underline{\delta}^+ = \Delta^+]$) sobre un sistema de tiempo real ($n,[\delta^-, \delta^+],[\mu^-, \mu^+]$) proporcionando transformación de un contador libre del modelo de cómputo clásico al algoritmo A para resolver algún problema P basado en trace y resolver al modelo computacional en tiempo real del algoritmo S_A en P.

Intuitivamente, la restricción del algoritmo de tiempo libre se necesita porque el modelo computacional en tiempo real, no se puede garantizar que

un conjunto del reloj en algún tiempo de reloj en hardware se procesará para entonces (el sistema quizá esté ocupado). Este problema no aparece en el modelo computacional clásico donde se computan pasos que toman tiempo cero. La restricción de los problemas basado en trace, es decir, los problemas que contienen únicamente propiedades de trace, son apenas una consideración simplificada por la decisión de omitir el tratamiento de propiedades de estado.

El mayor problema en la transformación del modelo computacional clásico en uno de tiempo real, es que en el modelo clásico computacional existe la dependencia circular del algoritmo A en los retardos reales end-to-end y viceversa, por un lado, el algoritmo A del modelo clásico computacional se ejecuta sobre la simulación y quizás necesite saber que el limite $[\underline{\delta}^{-},\underline{\delta}^{+}]$ del retraso de la simulación del mensaje, es apenas el límite del retraso end-to-end $[\Delta^{-},\Delta^{+}]$ de la simulación subrayada, por otro lado esos retrasos end-to-end, implican el retraso de la cola w y es dependiente de (el patrón del mensaje) A y en $[\underline{\delta}^{-},\underline{\delta}^{+}]$ [6].

Esta dependencia circular se puede romper de la siguiente manera:

Dado algún algoritmo A del modelo clásico computacional se asume un retraso de mensajes con límites en el intervalo $[\underline{\delta^{-}},\underline{\delta^{+}}]$, considerado parámetros sin valor, en el algoritmo S_A del análisis programado en tiempo real. Esto proporciona una ecuación del retraso end-to-end en el límite del análisis programado en tiempo real $[\Delta^{-},\Delta^{+}]$ en términos de los parámetros de sistemas de tiempo real $n,[\delta^{-},\delta^{+}],[\mu^{-},\mu^{+}]$ y los parámetros del algoritmo $[\underline{\delta}^{-}=\Delta^{-},\underline{\delta}^{+}=\Delta^{+}]$, es decir, una función F que satisface

$$[\Delta^{-},\Delta^{+}]=F(n,[\delta^{-},\delta^{+}],[\mu^{-},\mu^{+}],[\Delta^{-},\Delta^{+}])$$ ecuación 3.1

La clave para esta transformación es una simulación muy sencilla:

Al recordar que un algoritmo está especificado como un mapeo de los índices de procesadores a un conjunto de estados iniciales y una función de transición, donde esta función de transición es idénticamente para el modelo de cómputo clásico y el modelo de tiempo real. Dado que S_A sea un algoritmo para el modelo computacional en tiempo real, comprendiendo exactamente la misma función de estados iniciales y la transición dado un algoritmo *A* del modelo clásico computacional.

1. <VARIABLES GLOBALES DE *A*>
2.
3. FUNCIÓN S_A PROCESO_DEL_MENSAJE(MSG,TIME)
4. A-PROCESO_DEL_MENSAJE(MSG,TIME)

FIGURA 3-2. SIMULACIÓN DEL MODELO DE CÓMPUTO CLÁSICO SOBRE EL CÓMPUTO EN TIEMPO REAL

3.5.3 Sincronización del reloj

En el modelo clásico computacional, un límite estrecho es $(1-\frac{1}{n})\underline{\varepsilon}$ y está probado en [7] como la mejor sincronización del reloj confiable en precisión. Además, un algoritmo $A([n,\delta^-,\delta^+])$ se garantiza con una precisión óptima en cada sistema clásico $([n,\delta^-,\delta^+])$ con $\varepsilon=\underline{\delta}^+-\underline{\delta}^-$. El algoritmo trabaja para enviar a cada procesador un mensaje de tiempo con un sello de tiempo (*timestamp*) a los otros procesadores, y después calcular el promedio estimado de las diferencias del reloj como valor de corrección. Cualquier procesador difunde su mensaje tan pronto como lleguen los mensajes *init*.

Las transformaciones proporcionadas en la sección anterior se usan para generalizar estos resultados al modelo computacional del tiempo real, arrojando un límite superior de

$$(1-\frac{1}{n})(\varepsilon_{(n-1)}+\mu^+_{(n-1)}+(n-2)\mu^+_{(0)})$$

y un límite inferior $(1-\frac{1}{n})\varepsilon_{(1)}$:

En el modelo computacional en tiempo real, la sincronización de los relojes está dentro de $(1-\frac{1}{n})(\varepsilon_{(n-1)}+\mu^+_{(n-1)}+(n-2)\mu^+_{(0)})$ y esto es posible.

El siguiente algoritmo alcanza la precisión óptima en difusión (broadcast)

```
1.  var estimates ← {}
2.  var adj
3.
4.  function process_message(msg,time)
5.    /* start alg. by sending (SEND) to proc. 0 */
6.    if msg=(SEND)
7.       send(TIME,time) to all other processors
8.    elseif msg=(TIME,remote_time)
9.       estimates.add(remote_time-time+ (δ⁻ + δ⁺)/2 )
10.   if estimates.count=ID
11.      send timer(SEND) for time + max( ε − δ⁻ + μ⁺ , μ⁺ )
12.   if estimate.count=n-1
13.      adj ← (Σ estimates)/n
```

FIGURA 3-3. ALGORITMO DE LA SINCRONIZACIÓN DEL RELOJ PARA SISTEMAS QUE UTILIZAN BROADCAST

El algoritmo modificado, representado en la figura 3-3, funciona como sigue:

Los n procesadores conectados tienen ID's de 0 a n-1, el primer procesador (ID=0) envía su valor de reloj al resto de los procesadores.

El procesador i espera hasta que haya recibido el mensaje del procesador (i-1), también espera el valor de la función max($\varepsilon - \delta^- + \mu^+, \mu^+$) en unidades de tiempo, y entonces difunde su propio valor de reloj en hardware. De esa manera, cada procesador recibe el valor de reloj en hardware de todos los otros procesadores con incertidumbre ε, y provee que no hubo encolamiento (que será demostrado abajo). Esta sincronización es suficiente para sincronizar los relojes dentro de $(1-\frac{1}{n})\varepsilon$. Se asume solamente se envía un mensaje del tipo init (el procesador 0), como mensajes adicionales del tipo init no causan efectos de cola y por lo tanto no es necesario una segunda ronda de intercambios de mensajes.

Algoritmo para mensajes unicast.

El algoritmo de la sección anterior proporciona la sincronización del reloj dentro de $(1-\frac{1}{n})\varepsilon_{(n-1)}$. Sin embargo, a menos que el tiempo constante de la difusión esté disponible, $\varepsilon_{(1)}$ será generalmente más pequeña que $\varepsilon_{(n-1)}$.El algoritmo se puede adaptar para enviar unicast según las indicaciones de la figura 3-4, sin embargo, más que el enviar todos los n-1 mensajes inmediatamente, se envían en n-1 trabajos subsecuentes conectados por mensajes de tipo send, cada envío del mensaje es solamente uno. Estos mensajes son sellos de tiempo (timestamp) con su valor correspondiente a HC.

El siguiente algoritmo alcanza la precisión óptima en unicast. El de la figura 3-4 logra una precisión de $(1-\frac{1}{n})\varepsilon_{(1)}$. Manda exactamente n(n-1) = $O(n^2)$ mensajes a lo largo del sistema y tiene O(n) en tiempo de complejidad

```
1.  var estimates ← {}
2.  var adj
3.
4.  function process_message(msg,time)
5.    /* start alg. by sending (SEND ,1) to proc. 0 */
6.    if msg=(SEND,1)
7.       send(TIME,time) to target
8.      if target +1 mod n ≠ ID
9.           send timer (SEND, target +1 mod n) for time +μ+
10.   elseif msg=(TIME,remote_time)
11.      estimates.add(remote_time-time+ (δ⁻ + δ⁺)/2 )
12.      if estimates.count=ID
13.           send timer(SEND) for time + max(ε − δ⁻ + 2μ⁺, μ⁺)
14.      if estimate.count=n-1
15.           adj ← (Σ estimates)/n
```

FIGURA 3-4. ALGORITMO PARA LA SINCRONIZACIÓN DEL RELOJ PARA SISTEMAS QUE UTILIZAN UNICAST

3.6 Limite Inferior

En particular, para la precisión óptima, se demuestra en [7] que por lo menos $\frac{1}{2}n(n-1)=\Omega(n^2)$ mensajes se deben intercambiar, es decir, cada conexión debe mandar al menos un mensaje. Este límite es asintóticamente cercano al teórico demostrado por Lynch. En el modelo clásico computacional, una red completamente conectada con la incertidumbre igual a la conexión no puede lograr mejor precisión que $\frac{1}{2}\varepsilon$, mientras que al quitar una conexión rinde un límite inferior de ε, así, después de quitarla, la precisión óptima de $(1-\frac{1}{n})\varepsilon$ mostrada por [7] no puede ser mayor al que se ha logrado [6].

3.6.1 Complejidad del Mensaje

Para la sincronización del reloj dentro de algunos $\gamma < \epsilon(1)$. Implica que esto es un rt-run ya que son mensajes dirigidos y tienen un diámetro < 2, por decir, el mensaje gráfico está conectado completamente y tiene $\frac{n(n-1)}{2}$ bordes. Esto lleva al siguiente teorema.

Teorema 3.1. La sincronización de relojes de $\gamma < \epsilon(1)$ tiene un mensaje en el peor de los casos con complejidad menor a $\frac{n(n-1)}{2}=\Omega(n^2)$.

Se presenta un algoritmo que alcanza una precisión óptima de $(1-\frac{1}{n})\varepsilon_{(1)}$ con $n(n-1)=\Omega(n^2)$ mensajes.

Obviamente, un pequeño límite inferior puede obtener una subóptima sincronización del reloj. Se emplea el siguiente lema grafico-teórico simple:

Lema 3.1. En un gráfica no dirigida con n > 2 nodos y diámetro D o menos, hay por lo menos un nodo con grado > $\sqrt[D+1]{n}$.

Prueba. Se asume por contradicción que todos los nodos tienen un grado máximo de algún entero no-negativo D < $\sqrt[D+1]{n}$. Como n > 2, d=0 o d=1 causa que la gráfica esté desconectada, con lo cual se contradice la suposición del límite del diámetro. Así, se puede asumir que d> 1.

Fijar algún nodo P. Claramente, después de D saltos, el número máximo de nodos accesibles de P (inclusive P en la distancia 0) es $\sum_{i=0}^{D} d^i = \frac{d^{D+1}-1}{d-1} \le d^{D+1} < \sqrt[D+1]{n}^{\,D+1} = n$, cuando no se pueden alcanzar n nodos después de D saltos, así queda demostrado [6].

3.6.2 Complejidad en el Tiempo

En el caso de la precisión óptima, el siguiente teorema tiene un estado de complejidad en tiempo Ω(n) como límite inferior. Este límite es asintóticamente cercano al calculado teoricamente.

Teorema 3.2. La sincronización del reloj dentro de $\gamma<\varepsilon_{(1)}$ tiene una complejidad de tiempo en el peor caso de por lo menos Ω(n).

Prueba. Los n procesadores necesitan intercambiar por lo menos $\frac{n(n-1)}{2}$ mensajes. Por lo tanto, ningún algoritmo puede lograr ejecutarse en un mejor tiempo max($\frac{n(n-1)}{2}\mu^{+}_{(0)}, \delta^{+}_{(\frac{n-1}{2})}$) asumiendo concurrencia óptima. Esto implica una complejidad en tiempo de Ω(n) como límite inferior.

A diferencia del capítulo anterior, este nuevo concepto ayuda a entender las diferentes variables en tiempo real y los tiempos “máximos” que se deben considerar para que el algoritmo de la sincronización del reloj sea lo más óptimo posible. Este capítulo es la base para realizar el análisis y contar con una solución a este problema separándolo de variables no controlables en el dispositivo a considerar (en este caso las computadoras).

Capítulo 4

Análisis de Requerimientos

En los dos capítulos anteriores se realizó una breve introducción a las dos investigaciones para resolver la sincronización del reloj en cómputo distribuido. El capítulo 2 considera un autómata (modelo clásico) y el capítulo 3 considera más variables y tiempo de procesamiento diferente de cero (modelo en tiempo real), pero su base es el algoritmo usado y probado teóricamente [7].

Durante esta investigación se considera al algoritmo que se implantará, éste debe ser más rápido y/o más simple con respecto a las otras investigaciones que se han estudiado [6] y [7]. Considerando la eliminación de variables no controladas o tratado de hacerlas casi nulas.

El objetivo fundamental de este capítulo es demostrar que se puede introducir un algoritmo en el protocolo Ethernet, que es una una capa más baja donde se encuentran actualmente los algoritmos funcionando, con el fin de que el ASR no se ejecute en el procesador de la computadora correspondiente y se cuente con el control de las otras variables a considerar, de no ser el caso, mediante estadísticas y pruebas estocásticas tener los tiempos de las variables no controladas.

En los artículos que se han investigado y estudiado (Mills, Lamport, Liskov, entre otros) se toman en cuenta y demuestran dos requerimientos fundamentales:

- La sincronización del reloj debe utilizarse cuando las computadoras interactúan entre sí.

- La solución del algoritmo de sincronización del reloj debe ser una solución simple y con poco cálculo (cómputo).

Antes de implantar el algoritmo analizado y evaluado para el hardware, este capítulo contendrá lo siguiente:

La primera sección hablará de las ventajas y desventajas de los dos algoritmos analizados, las características para saber cuál es el mejor para implantarlo en hardware considerando:

- Solución Simple
- No realizar operaciones de alta complejidad.
- No depender de otros algoritmos.

Se han descartado las pruebas con simulaciones y equipos virtuales ya que no se pueden medir los tiempos, además el comportamiento de este algoritmo se tiene que mostrarse con tiempo real. Por esta razón las simulaciones quedan fuera del alcance de la tesis.

Además se debe entender que el análisis demostrado por Daniel Albeseder [19] acerca de los tiempos que utiliza el algoritmo de sincronización es la base de la presente tesis para su solución.

La sección 4.2 investiga y analiza la capa de enlace de datos, una de las más bajas en el protocolo TCP/IP, para conocer y comprender las cabeceras de los paquetes correspondientes, esta capa es de gran utilidad para invocar al algoritmo desde la interfaz de red (NIC) para su implantación en hardware.

En la sección 4.3 se tienen como objetivo los fundamentos y las características del lenguaje de programación verilog para usarlo en el algoritmo seleccionado e implantarlo en un dispositivo FPGA. En ningún momento se pretende programar el protocolo CSMA/CD en verilog, por lo que se investigará el código existente para el dispositivo FPGA y así implementarlo agregando la funcionalidad para el algoritmo de la sincronización del reloj.

4.1 Ventajas y Desventajas de los Algoritmos Analizados

4.1.1 Teoría de la Sincronización

En el caso de la investigación de Boaz Patt-Shamir (La Teoría de Sincronización [18]) tiene una forma muy práctica de encontrar las rutas mínimas, tanto para la latencia del mensaje como para los límites de desvío de los procesos.

La visualización de la solución para la sincronización del reloj sin considerarlo por medio de autómatas es la siguiente:

- Tiempo de procesamiento en P2.
- Tiempo de envío del mensaje de P2 hacia P1.
- Tiempo de procesamiento en P1.
- Tiempo de envío del mensaje de P1 hacia P2.

Estos tiempos se ven en la figura 4-1 A) Vista Conceptual y 4-1 b) Boaz-Patt con la representación de mensajes y envíos de los mismos.

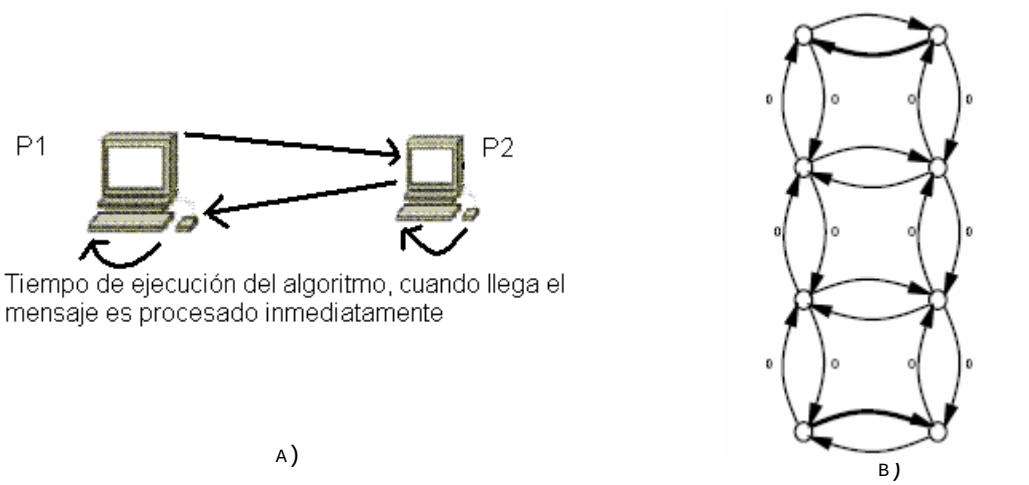


FIGURA 4-1. A) VISTA CONCEPTUAL Y B) PARADIGMA DE BOAZ-PATT

El algoritmo de Boaz-Patt es robusto debido a que para encontrar rutas mínimas emplea el algoritmo de Bellman-Ford (algoritmo RIP). Boaz utiliza el algoritmo mencionado y no el de Djisktra debido a que Bellman-Ford usa pesos negativos para la solución [18].

La desventaja considerable de la teoría de sincronización es la complejidad para encontrar las rutas ya que depende de la complejidad de tiempo y los mensajes en el algoritmo de Bellman-Ford, un algoritmo bastante robusto dada su complejidad en tiempo de ejecución la cual es $O(VE)$.

Implementarlo en hardware será demasiado costoso debido a las limitaciones físicas. Esto conduce a que serán dos algoritmos que se ejecutarán en el hardware, los cuales son: el algoritmo de sincronización del reloj más el algoritmo de Bellman-Ford.

En la presente tesis se ha descrito uno de los principales requerimientos para la solución:

- **El algoritmo debe ser simple en cuanto al cálculo.**

La teoría de sincronización no cumple con los requerimientos para la solución, existen variables que no están consideradas en la teoría de sincronización porque su nivel de abstracción es mayor, ya que la solución se encuentra en una capa que no se considera.

4.1.2 Modelo del Sistema en Tiempo Real

En el algoritmo utilizado por Ulrich y Moser [6], su base es el algoritmo propuesto por N. Lynch. Se realizan las consideraciones con más variables y en tiempo real, para ello usan el modelo Teta [21].

Este modelo es gran avance debido a que propone analizar el tiempo de procesamiento y el calendarizador de tareas (scheduler). Además está

comprobado para el algoritmo de sincronización interna del reloj. Han realizado pruebas estocásticas del algoritmo dentro del modelo Teta, donde a continuación se muestran los resultados de los mismos.

Los criterios que se toman en cuenta en [19] para conocer los tiempos de las variables a considerar son los siguientes:

- Carga en el CPU.
- Carga en la red.
- Carga en los dos puntos anteriores.

Los algoritmos analizados (broadcast y unicast) son sencillos y de cálculo simple. El modelo considera variables de tiempo real o no sería "tan abstracto" como los modelos anteriores.

Estas pruebas han hecho que los algoritmos propuestos hasta el momento puedan alcanzar los tiempos garantizados en los documentos de Lynch [7].

Una de las principales desventajas del modelo es que los tiempos han sido calculados únicamente con el modelo Teta, este modelo aunque prueba ser robusto y confiable, tiene la limitante de que se desarrolla en un ambiente controlado, es decir, se hizo una propuesta de computadoras con características similares: el mismo sistema operativo (linux), calendarizador (scheduler), enlace con otras variables para asegurar el control y/o la medición de las variables en el modelo Teta [19].

Por otro lado, al ser controlado no es un modelo heterogéneo. Aunque el modelo muestra valores más reales, éstos pueden cambiar considerablemente al modificar cualquier característica del ambiente. Al realizar pruebas estocásticas, su objetivo no es encontrar rutas mínimas sino el límite mínimo y máximo de las variables $\delta^{-}, \delta^{+}, \mu^{-} y \mu^{+}$, y así lograr un enfoque diferente a la Teoría de la Sincronización.

La Teoría de la Sincronización busca "tiempos mínimos" para ejecutarse, mientras el sistema de tiempo real evalúa y obtiene los límites de las variables $\delta^{-}, \delta^{+}, \mu^{-} y \mu^{+}$ en el algoritmo. El algoritmo a utilizar es simple y es una pieza fundamental para el problema de la sincronización del reloj se puede adaptarse al hardware.

Aunque los dos algoritmos se pueden programar en hardware, el algoritmo de Moser y Ulrich es más sencillo y cuenta con un análisis de pruebas en dicho modelo, además no depende de otro algoritmo para su funcionamiento (lo que reduce tiempo y procesamiento). Se decide utilizar el algoritmo que propone Moser para implantarlo en hardware.

Con los análisis realizados en las dos investigaciones se han considerado otras líneas. La primera línea de investigación fusiona estas dos teorías situándolas al mismo nivel, debido a que se encuentran en diferentes niveles de abstracción.

La segunda línea de investigación adopta un algoritmo y lo implanta en hardware para reducir aún más los tiempos no controlables, manipularlos y si es posible eliminarlos.

La tercera línea de investigación adopta un algoritmo en hardware y cuando esté “robusto”, combinarlo con otro algoritmo.

Se decide adoptar la segunda línea de investigación por los motivos, que se enumeran a continuación:

- Punto de vista diferente a las dos investigaciones analizadas.
- Se puede comprobar la tercera línea de investigación a partir de la segunda. Sin embargo, ésta última necesita más análisis y reducir en ambas investigaciones su nivel de abstracción.
- Utilizando las primeras capas del protocolo TCP/IP se ahorra tiempo de procesamiento, empleando como base los resultados de las gráficas anteriores.

Para fortalecer está la hipótesis donde se ocupa más tiempo de cómputo, de esta manera se realiza la propuesta de programación sobre el dispositivo FPGA.

Se utiliza el algoritmo de Ulrich y Moser por ser el más completo y porque considera más variables en el entorno de computadoras en tiempo real.

El algoritmo elegido se programa en un dispositivo FPGA y después se implanta en una tarjeta de red (NIC), con esto se reducen los tiempos $\mu^{-} y \mu^{+}$, debido a las soluciones anteriores, se encuentran en las capas superiores (cuadros grises) del modelo TCP/IP, como se aprecia en la figura 4-2.

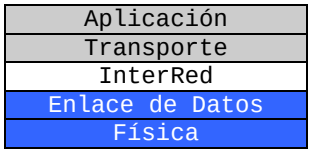


FIGURA 4-2. PILA DEL PROTOCOLO TCP/IP

Cambiando la programación a niveles inferiores se pueden ahorrar de 2 a 3 capas del protocolo y tiempo de procesamiento, el cual es realizado por el dispositivo de red, en este caso el FGPA.

La solución del algoritmo de sincronización del reloj para que sea más rapido se observa desde el punto de vista de procesamiento y no desde el punto de vista del protocolo TCP/IP.

En la figura 4-3 a) se muestran los dispositivos donde se ejecuta el algoritmo de sincronización del reloj y en la figura 4-3 b) el dispositivo que ahora realiza el procesamiento y la iteracción con el ASR.

FIGURA 4-3. VISTA DEL MODELO ANTERIOR Y ACTUAL

4.2 Capa de Enlace de Datos del Protocolo TCP/IP

Actualmente uno de los protocolos más difundido alrededor del mundo para la comunicación entre computadoras es el protocolo TCP/IP (Transmision Control Protocol/Internet Protocol RFC 1180). Este protocolo en realidad es una familia de protocolos de Internet, donde ambos (TCP e IP) son los más utilizados. TCP/IP es la base de Internet para comunicar todo tipo de computadoras y se encuentran implantados en diferentes sistemas operativos.

En la extensa literatura se podrá encontrar la analogía del modelo TCP/IP con el modelo de referencia OSI y cómo las siete capas de este último permiten un mayor entendimiento en la comunicación entre computadoras, pero el modelo TCP/IP es el que realmente se usa. Por lo que a continuación se muestran los niveles del modelo TCP/IP y una muy breve introducción de los cinco niveles.

El nivel físico describe las características físicas de la comunicación, por ejemplo las convenciones sobre la naturaleza del medio usado para la comunicación (como las comunicaciones por cable, fibra óptica o radio), y todo lo relativo a los detalles de las interfaces, cómo son los conectores, código de canales, modulación, potencias de señal, longitudes de onda, sincronización, temporización, distancias máximas, entre otros.

La capa de enlace de datos especifica cómo son transportados los frames sobre el nivel físico, incluyendo los delimitadores (patrones de bits concretos que marcan el comienzo y el fin de cada frame). Esta capa se subdivide en dos: control de enlace lógico (logical link control) y control de acceso al medio (media access control).

El nivel de interred y en especial el protocolo IP realiza las tareas básicas para conseguir transportar datos desde un origen a un destino. IP puede pasar los datos a una serie de protocolos superiores; cada uno de ellos se identifica con un único "Número de protocolo IP". *ICMP* e *IGMP* son los protocolos 1 y 2, respectivamente. Algunos de los protocolos por

encima de IP como ICMP e IGMP van en niveles superiores a IP pero realizan funciones del nivel de red.

El nivel de transporte puede solucionar problemas como la fiabilidad y la seguridad de los datos y que éstos lleguen en el orden correcto. En el conjunto de protocolos TCP/IP, los de transporte también determinan a qué aplicación van destinados los datos. Los dos protocolos más utilizados son TCP y UDP.

En el nivel de aplicación los programas de usuario final más comunes se encuentran a través de la red con otros programas. Los procesos que acontecen en este nivel son aplicaciones específicas que pasan los datos al nivel de aplicación en el formato que internamente comprenda el programa y es codificado de acuerdo con un protocolo estándar.

Algunos programas específicos se ejecutan en este nivel. Proporcionan servicios que directamente trabajan con las aplicaciones del usuario. Estos programas y sus correspondientes protocolos incluyen http, ftp, smtp, ssh, dns y muchos otros.

Una vez realizada la breve descripción del modelo TCP/IP, lo siguiente es el análisis de la segunda capa del protocolo TCP/IP (enlace de datos). Toda la explicación de la capa de enlace de datos se encuentra en el Apéndice G.

4.3 Selección del lenguaje de programación para el dispositivo FPGA

A continuación se describen los lenguajes de programación y se selecciona el adecuado para el desarrollo del algoritmo de sincronización del reloj en el dispositivo FPGA, se cuentan con varias opciones las cuales son:

- VHDL
- ABEL y
- Verilog

Para elegir un lenguaje de éstos que sea conveniente para el propósito de esta tesis, primero se tiene que considerar e investigar el código (denominado IP Core) en estos lenguajes. El código debe ser el protocolo CSMA/CD o Ethernet, debido a que el propósito de la tesis no es crear el código sino modificarlo agregándole el algoritmo de sincronización y adaptarlo como un módulo.

El dispositivo que se usa para la programación es un FPGA de la compañía Xilinx. El modelo de la tarjeta es una Virtex II-PRO Xc2VP30 con 30816 celdas lógicas. A partir del dispositivo se investigaron los lenguajes compatibles con él, los cuales son VHDL y Verilog [28].

Buscando e investigando en la WEB, se encontraron varios IP Cores, pero muchos de ellos son de código objeto y no se pueden modificar para agregar más módulos. Un sitio que se encontró y el cual está muy completo en documentación es *www.opencores.org*. El sitio cuenta con varios

proyectos y uno de los cuales es el denominado Ethernet IP Core, dicho código se encuentra en verilog [26].

Dado este requerimiento, el lenguaje para desarrollar el módulo ASR será verilog.

Al seleccionar el lenguaje verilog se hace una breve descripción con sus propiedades más importantes, se concluye este capítulo con la solución para la implementación.

Verilog fue inventado por Phil Moorby en 1985 mientras trabajaba en Automated Integrated Design Systems, más tarde renombrada Gateway Design Automation. El objetivo de Verilog era ser un lenguaje de modelado de hardware. Gateway Design Automation fue comprada por Cadence Design Systems en 1990. Cadence ahora tiene todos los derechos sobre los simuladores lógicos de Verilog y Verilog-XL hechos por Gateway.

Verilog es un lenguaje de descripción de hardware (HDL, Hardware Description Language) usado para modelar sistemas electrónicos. El lenguaje, algunas veces llamado Verilog HDL, soporta el diseño, prueba e implementación de circuitos analógicos, digitales y de señal mixta a diferentes niveles de abstracción.

El diseño primordial de Verilog es que fuera un lenguaje con una sintaxis similar a la del lenguaje de programación C, para que fuera rápidamente aceptada. El lenguaje tiene un preprocesador como C.

Con el éxito de VHDL, Cadence decidió hacer el lenguaje abierto y disponible para la estandarización. Cadence transfirió Verilog al dominio público a través de Open Verilog International, actualmente se conoce como Accellera. Verilog fue después enviado a la IEEE, quien lo convirtió en el estándar IEEE 1364-1995, habitualmente referido como Verilog 95.

Extensiones a Verilog 95 se enviaron a la IEEE para cubrir las deficiencias que se habían encontrado en el estándar original de Verilog. Estas extensiones se volvieron el estándar IEEE 1364-2001 conocido como Verilog 2001.

El advenimiento de los lenguajes de verificación de alto nivel como OpenVera y el lenguaje E de Verisity impulsó el desarrollo de Superlog, por Co-Design Automation Inc. Co-Design fue comprada más tarde por Synopsis. Las bases de Superlog y Vera fueron donadas a Accellera, se transformaron y actualizaron para quedar como SystemVerilog, que probablemente se convierta en el próximo estándar de la IEEE.

Verilog contiene una basta serie de instrucciones construidas, incluyendo compuertas lógicas, primitivas definidas por usuarios, switches y lógica alambrada. También tiene los dispositivos de retardos de pin a pin y verificadores de tiempo.

Los diferentes niveles de abstracción son en esencia, proveídos por la semántica de dos tipos de datos: variables y nets. La asignación continua de las expresiones de ambas variables y nets puede conducir a valores continuos dentro de las nets, generando la estructura básica de construcción. Un diseño puede ser estructural, de comportamiento o una

mezcla y consiste de un conjunto de módulos, cada uno de los cuales tienen una interfaz de I/O y una descripción de estas funciones.

Estos módulos están formados por una jerarquía y están interconectados con redes (nets en FPGA). El lenguaje Verilog es extensible vía las subrutinas de la Interfaz de Lenguaje de Programación (PLI) y la Interfaz de Procedimientos Verilog (VPI). El PLI/VPI es una colección de rutinas que permite funciones externas para acceder al contenido de la información del diseño descriptor en Verilog HDL y facilidades para la iteración dinámica con las aplicaciones de simulación de PLI/VPI. Además interactúa con otros simuladores y sistemas CAD.

Un diseño típico del flujo para circuitos VLSI se muestra en la figura 4-4. Los bloques no sombreados muestran el nivel de representación del diseño; los bloques sombreados muestran procesos en el diseño de flujo.

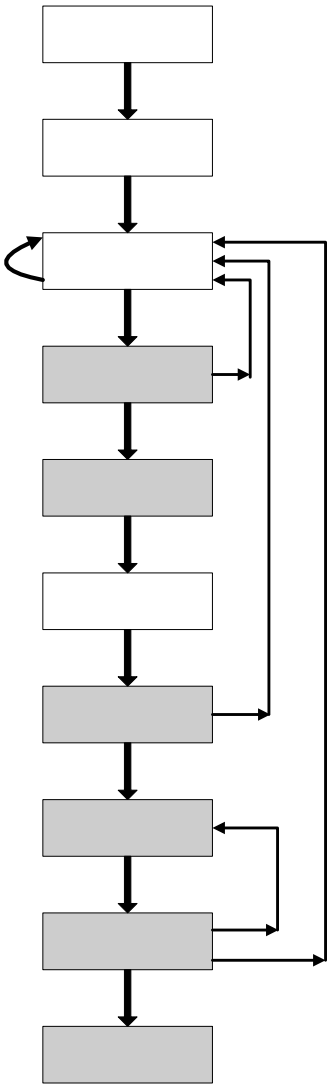


FIGURA 4-4. DIAGRAMA DE UN DISEÑO DE FLUJO

El diseño de flujo se usa frecuentemente por los diseñadores de HDL. En cualquier diseño, las especificaciones son descritas primero ya que éstas describen la funcionalidad abstracta, la interfaz y todo lo relacionado con la arquitectura del circuito para ser diseñado. En este punto no se necesita conocer cómo se implementará el circuito. Una descripción del comportamiento se crea para analizar el diseño en términos de funcionalidad, rendimiento, conformidad con los estándares, entre otras cosas.

En HDL se cuentan con dos tipos básicos de metodologías de diseño digital: la metodología de diseño top-down y la metodología de diseño bottom-up. En la metodología top-down, se define los bloques del nivel más alto y se identifican los sub-bloques necesarios para construir el bloque más alto. Se subdividen los sub-bloques tantas veces sea necesario hasta llegar a las hojas, las cuales ya no pueden ser divididas. En la figura 4-5 se muestra el diseño de proceso top-down.

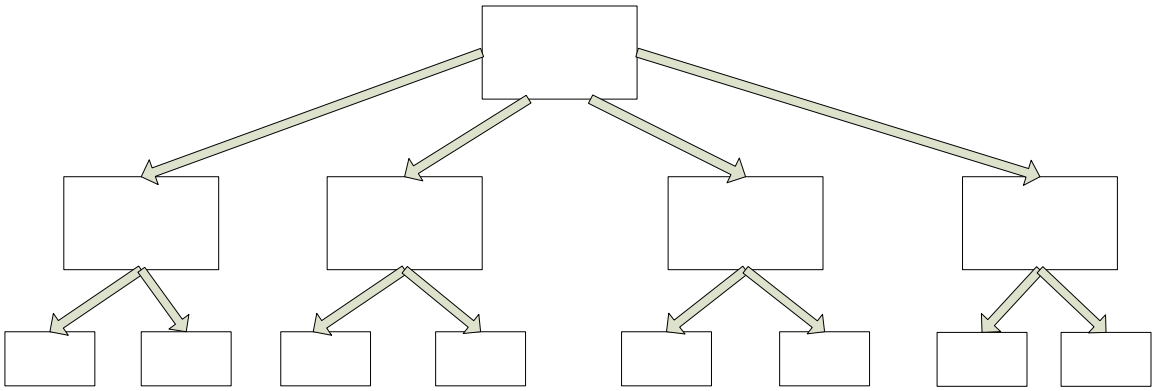


FIGURA 4-5. METODOLOGÍA DE DISEÑO TOP-DOWN

En la metodología de diseño bottom-up, primero se identifican los bloques construidos que se tienen disponibles. Después se construyen grandes celdas, usando estos bloques construidos. Estas celdas son empleadas entonces para crear otros bloques de mayor nivel hasta que se construya el bloque más alto en el diseño. La figura 4-6 muestra el proceso bottom-up

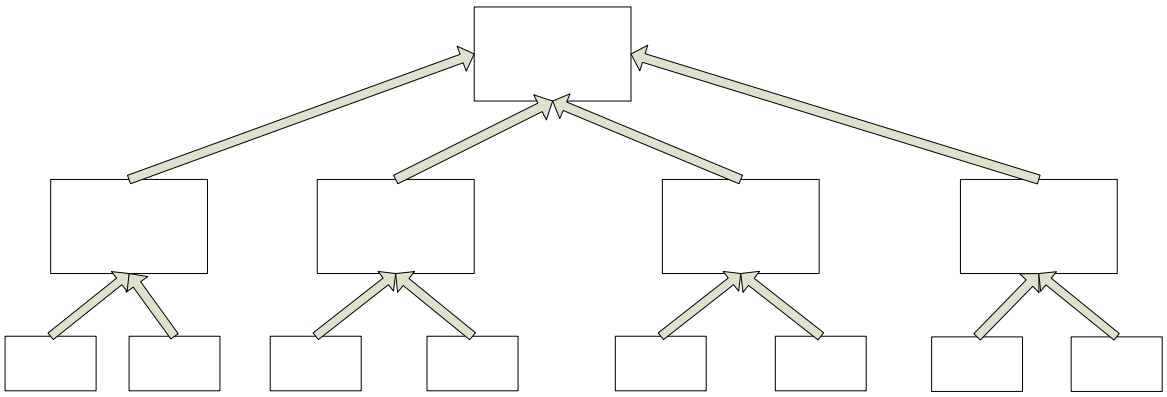


FIGURA 4-6. METODOLOGÍA DE DISEÑO BOTTOM-UP

Existe una combinación de ambos diseños. Los arquitectos del diseño definen las especificaciones de los bloques en el nivel superior. Los diseñadores lógicos deciden que el diseño debe ser estructurado para romper la funcionalidad dentro de los bloques o sub-bloques. Al mismo tiempo, los diseñadores de circuitos están desarollando circuitos óptimos para las celdas en el nivel hoja.

Capítulo 5

Implementación del Algoritmo de Sincronización del Reloj en un FPGA

Este capítulo tiene la misión de generar una alternativa a la solución de sincronización del reloj desde hardware, no a nivel de un nuevo algoritmo, sino para mejorar los tiempos de procesamiento y latencia de los mensajes ya que estas dos variables son críticas para el comportamiento del tiempo real. Para lograr los tiempos se analizó el protocolo MAC, el cual es uno de los protocolos de TCP/IP, se creó un módulo para una FPGA y se generaron nuevas funciones para la realización del mismo. Para este problema estudiado existen varias soluciones. Aunque la investigación se diversificó en diferentes soluciones, la siguiente propuesta no se delcaró como la más óptima para el problema, pero sí cp,p un avance para el mejoramiento del mismo.

En la sección 5.1 se explican los componentes del MAC Ethernet IP Core que se utilizó para comprender el funcionamiento de los módulos, se debe conocer la relación entre ellos pues con esta información se puede agregar y acoplar el nuevo módulo del algoritmo de sincronización del reloj.

La sección 5.2 explica el algoritmo de sincronización del reloj, los pasos que realiza, el pseudocódigo (en código verilog) y los módulos que lo integran. En la sección 5.3 se explica el algoritmo de ordenamiento para las direcciones físicas, éste se usa para decidir cuál será el orden en el que los dispositivos enviarán los mensajes de “tiempo local” en modo broadcast y su criterio con respecto a las direcciones físicas. Se describe un ejemplo del mismo, el pseudocódigo y por último el código en verilog con los resultados de salida.

La sección 5.4 explica la importancia de obtener los valores 2δ en el tiempo para la sincronización del reloj y cómo se relaciona con las tres propiedades del cómputo distribuido (Acuerdo, Validez y Terminación).

En la sección 5.5 se describe el módulo de gestión de paquetes. La función principal del módulo es el reconocimiento de paquetes de tipo ASR (Algoritmo de Sincronización del Reloj) para que el dispositivo FPGA realice el procesamiento y el cálculo para que los paquetes de tipo ASR se procesen en esta capa y no pasen a las capas superiores de TCP/IP. En la sección 5.6 se explica brevemente el módulo de función de tiempo que hace la iteración dispositivo FPGA y CPU.

La última sección (5.7) describe brevemente la integración del módulo ASR con el Ethernet MAC IP Core de Ethernet, así como su operación.

5.1 Ethernet MAC IP Core

La aplicación denominada Ethernet MAC IP Core realiza las funciones de control de acceso al medio (MAC por sus siglas en inglés) Ethernet de 10/100. Consiste de un core sintetizado *RTL* de verilog que provee todas las características necesarias para implementar la capa 2 del protocolo de Ethernet. Este se encuentra diseñado para ejecutar las especificaciones de IEEE 802.3 y 802.3u que definen los estándares 10 Mbps y 100 Mbps respectivamente [28]. El tipo de características que provee el core son las siguientes:

- Control de flujo y generación automática de las tramas en modo full duplex (IEEE 802.3μ).
- Detección de colisiones y auto retransmisión de colisiones en modo half duplex (protocolo CSMA/CD).
- Generación automática del CRC de 32 bits y verificación, además de la interfaz independiente del medio (MII) IEEE 802.3.
- Generación del preámbulo y su eliminación, respondiendo a un estatus completo de los paquetes TX/RX.

El Ethernet MAC IP Core desarrollado por Igor Mohor, donde el código reside en [28], consiste en 7 unidades principales: interfaz wishbone, módulo de transmisión, módulo de recepción, módulo de control, módulo interfaz independiente del medio (MII), módulo de registro y módulo de estatus. Los módulos tienen a su vez submódulos, se muestra el diagrama de algunos y mas adelante se describen los mismos. A continuación se presenta el esquema de las siete unidades principales en la figura 5-1:

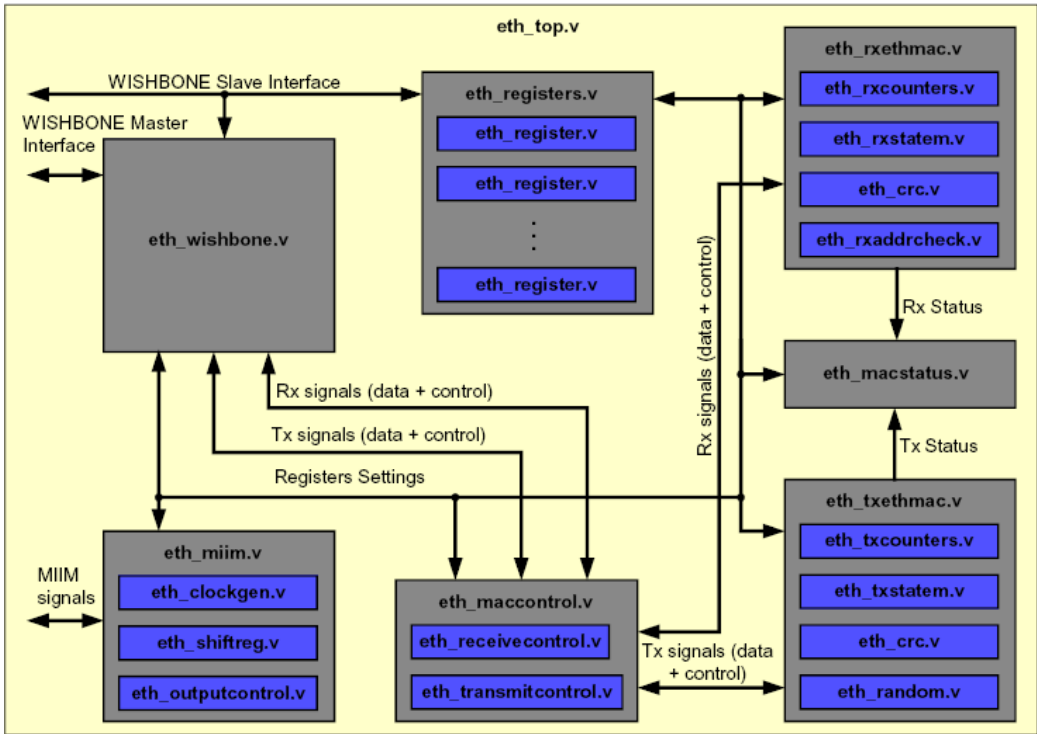


FIGURA 5-1. DIAGRAMA DE ETHERNET MAC IP CORE [28]

5.1.1 Interfaz wishbone

La interfaz wishbone consiste en dos interfaces, una maestra y otra esclava, las cuales se conectan al core del bus wishbone [28]. La interfaz maestra se usa para almacenar las tramas de los datos recibidos para la memoria y cargar los que necesitan ser enviados a la memoria para el Ethernet MAC IP Core. La interfaz wishbone es compatible con la revisión B.2 y B.3. El módulo contiene múltiples funciones:

a) Es la interfaz entre el Ethernet MAC IP Core y otros dispositivos (memoria, host). Las dos interfaces (maestro y esclavo) son usadas para el manejo de lo siguiente:

- Contiene buffer descriptores (en la RAM interna).
- Contiene receptores y transmisores FIFO.
- Contiene sincronización lógica para las señales que se extiende a través de los diferentes dominios de reloj.

b) Función relacionada con la transmisión que lee TX BD y cuando inicia la interfaz maestra wishbone, llena los FIFO TX y entonces realiza la transmisión. Al final escribe el estatus relacionado TX BD.

c) La función relacionada con la recepción que lee RX BD, ensambla los bytes entrantes a palabras y entonces escribe éstas en el FIFO RX. Son entonces escritas en la memoria a través de la interfaz maestra wishbone. Al Final escribe el estatus relacionado con RX BD.

d) Los registros Ethernet y los buffers descriptores (BD) acceden a través de la misma interfaz esclava de wishbone. Un ejemplo son los registros que están localizados en los módulos de eth_registers.

e) La selección entre los registros y accesos BD se terminan en el módulo eth_module.

f) El Ethernet MAC IP Core usa la interfaz maestra wishbone para ingresar al espacio de memoria cuando los buffers están almacenados. Ambos accesos de datos, transmisor y receptor lo hacen a través de la interfaz maestra wishbone. Se cuenta con una máquina de estados que contiene las siguientes señales:

- MasterWbTX
- MasterWbRX
- ReadTXDataFromMemory_2
- WriteRXDataToMemory
- MasterAccessFinished
- Cyc_cleared
- Tx_burst
- Rx_burst

Para mayor referencia del funcionamiento de las ocho señales consultar [28].

5.1.2 Módulo de Transmisión

El rendimiento de todas las operaciones relacionadas con la transmisión de los frames se encuentra en este módulo (generación de preámbulo, padding, CRC, entre otras). El módulo TX obtiene los datos que necesita para la transmisión desde la interfaz del módulo wishbone (WBI) en bytes. Además recibe señales que marcan el inicio y el fin de la trama de datos. Tan pronto como el módulo TX necesita los siguientes bytes de datos, pone activa la señal TXUsedData y entonces el módulo WBI provee el siguiente byte.

Este módulo consiste en cuatro submódulos:

- Eth_crc. El módulo de verificación de redundancia cíclica (CRC) que genera 32 bits para el CRC y que se agregan en el campo de datos.
- Eth_random. Genera retraso aleatorio que es necesario cuando se utiliza el algoritmo de backoff (después de una colisión).
- Eth_txcounters. Varios contadores necesarios para los paquetes transmitidos.
- Eth_txtstatem. Máquina de estados del módulo TX.

5.1.3 Módulo de Recepción

El módulo de recepción es el encargado de recibir los datos. El chip externo PHY los recibe en forma serial desde la capa física (cable), ensambla las partes y los envía al módulo de recepción (MRxD [3:0]) junto con el marco de “datos válidos” (MRxDV). El módulo de recepción entonces ensambla estas partes de datos como bytes de datos, y los envía entonces al módulo de la interfaz wishbone junto con unas pocas señales que marcan el inicio y el fin de los datos. El módulo de recepción suprime las cabeceras de preámbulo y del CRC [28].

El módulo de recepción se encuentra conformado por cuatro submódulos:

- Eth_crc.El módulo de verificación de redundancia cíclica (CRC).
- Eth_rxaddrcheck.El módulo de reconocimiento de direcciones.
- Eth_rxcounters.Varios contadores necesarios para la recepción de paquetes.
- Eth_rxstatem.La máquina de estados del módulo de recepción.

Además de estos submódulos, el módulo eth_rxethmac también cuenta con cierta lógica para:

- Generar el valor CrcHash y el marco CrcHashGood que se usan para el sistema reconocedor de direcciones.
- Asegurar los datos que se recibieron desde el chip PHY (RxData).

- Generar los frames de tipo broadcast y multicast (cuando se reciben los paquetes con direcciones de destino broadcast o multicast).

- Generar las señales RxValid, RxStartFrm, RxEndFrm que son marcos de los datos válidos.

5.1.4 Módulo de Control

El módulo de control es el encargado del manejo de flujo de datos, cuando el Ethernet MAC IP Core está en modo de operación en 100 Mbps full duplex.

El módulo de control consiste en lógica de múltiplexación y dos submódulos:

- Eth_transmitcontrol

- Eth_receivecontrol

El control de flujo se termina con el envío y la recepción de pausas de tramas de control.

El dispositivo que esté conectado a la interfaz wishbone (usualmente al procesador) y no puede procesar los paquetes que está recibiendo (por la alta tasa de envío), además de las peticiones que van llegando, crea un frame para que envíe un mensaje de pausa hacia la otra estación que se encuentra enviando los paquetes.

Tan pronto como la otra estación recibe la petición de pausa, se detiene la transmisión. Está se reanuda después del tiempo de petición de pausa o la petición de pausa se cambia por apagado (off). La transmisión del control de flujo se termina en el módulo eth_transmitcontrol.

La lógica de multiplexación es necesaria al multiplexar datos y señales de control, en una transmisión normal de datos y señales de control se usan una transmisión de tramas de control. Cuando las tramas de control se envían, el padding y la generación de CRC cambian automáticamente a encendido (on).

5.1.5 Módulo MII

El módulo MII (Media Independent Interface, Interfaz Independiente del Medio) es una interfaz al chip externo Ethernet PHY. Se usa para insertar la configuración física de los registros y leer el estatus del mismo. La interfaz consiste únicamente de dos señales: reloj (MDC) y la señal de datos bidireccional (MDIO). La señal bidireccional MDIO necesita combinarse con la de entrada Mdi, de salida Mdo, y la de habilitación MdoEn en el módulo adicional. Esta termina cuando el mismo Ethernet MAC IP Core se implementa en FPGA.

El módulo MII es el módulo principal y consiste de varios submódulos (eth_clockgen.v, eth_shiftreg.v, eth_outputcontrol.v) y de la lógica adicional. Ésta se usa para generar las siguientes señales:

- Petición de sincronización para escribir las operaciones (WriteDataOp), la lectura (ReadStatusOp) y la exploración (ScanStatusOP)
- Señal para actualizar el registro MIIRX_DATA (UpdateMIIRX_DATAReg)
- El contador binario (BitCounter) de la interfaz MII
- Señales de selección de bits que se emplean cuando los datos se cambian hacia a fuera (ByteSelect [3:0])
- Señales que se utilizan para asegurar los datos de entrada (LatchByte [1:0])

Cuando hay una necesidad para leer o escribir los datos desde el chip PHY, considerese que:

a) El registro MIIMODER necesita estar activado cuando:

1. El reloj divisor necesita iniciarse para proveer la señal del reloj en la frecuencia apropiada.
2. La generación del preámbulo se deshabilita (si PHY soporta las transmisiones sin el preámbulo). Por default el preámbulo de 32 bits es transmitido.
3. El módulo MII se reinicia por prioridad cuando éste se utilice.

b) La dirección Física (PHY) y la dirección del registro con el chip PHY seleccionado necesita estar puesto en el registro MIIADDRESS.

c) Si existe la necesidad para escribir datos en el registro seleccionado, los necesarios son escritos por el registro MIITX_DATA.

d) Escribiendo los valores apropiadamente en el registro MIICOMMAND, inicia la operación de petición.

e) Si las operaciones “estatus de lectura” o “estatus de exploración” estuvieron con peticiones donde el valor de éstas fue recibido desde el PHY, entonces pueden leerse desde el registro MIIRX_DATA.

f) El registro MIISTATUS refleja el estatus del módulo MII. Si el estatus de la falla del enlace (linkFail) se encuentra limpio únicamente después de leer el registro de estatus PHY (dirección 0x1), entonces regresa el estatus a OK.

5.1.6 Módulo de Registro

Funcionalmente los registros están descritos en la especificación del Ethernet MAC IP Core. Aunque se consideran como registros de 32 bits, únicamente se utiliza el ancho necesario. Otros bits se fijan en cero (ignoran la escritura y lectura como cero). Cada registro es instanciado con dos parámetros, los valores de ancho y reinicio. El valor de reinicio define cuando los registros se limpian y ponen el valor a cero o emplean un valor predefinido después del reinicio [28]. Este módulo contiene un solo registro. El ancho del registro está dado por dos parámetros definidos:

- WIDTH
- RESET_VALUE

5.1.7 Módulo de Estatus

El módulo de estatus es el encargado para monitorear las operaciones del MAC Ethernet. Los módulos de los monitores contienen condiciones y después de cada operación completa (recepción o envío de tramas), escribe un estatus relacionado con los buffers descriptores. No todos los estatus están escritos para los buffers descriptores. Los estados para recibir tramas se aseguran normalmente al finalizar la recepción de la etapa (cuando la señal TakeSample vale 1), tan pronto como los estados se reinicien (cuando la señal LoadRxStatus va a 1).

5.2 Módulo del Algoritmo para la Sincronización del Reloj (ASR)

Debido al requerimiento de utilizar el lenguaje Verilog, se realiza el desglose de los elementos a considerar. Se muestra en la figura 5-2 [6] el algoritmo con el que se va a trabajar en la presente tesis:

1. ***var estimates ← {}***
2. ***var adj***
3. ***function process_message(msg,time)***
4. ***/* start alg. by sending (SEND) to proc. 0 */***
5. ***if msg=(SEND)***
6. ***send(TIME,time) to all other processors***
7. ***elseif msg=(TIME,remote_time)***
8. ***estimates.add(remote_time-time+*** $\frac{\delta^- + \delta^+}{2}$ ***)***
9. ***if estimates.count=ID***
10. ***send timer(SEND) for time + max(*** $\varepsilon - \delta^- + \mu^+, \mu^+$ ***)***
11. ***if estimate.count=n-1***
12. ***adj ← (Σ estimates)/n***

FIGURA 5-2. ALGORITMO DE SINCRONIZACIÓN DEL RELOJ PARA SISTEMAS QUE UTILIZAN BROADCAST

Para comprender el funcionamiento del algoritmo e implementarlo en hardware se muestran las iteraciones que se realizan, considerando lo siguiente:

Existen cuatro dispositivos con el ASR implantado y éstos llevan a cabo en un tiempo determinado el ajuste de los relojes locales, los tiempos son fijos y varían según el dispositivo (vease página 17), cuando un dispositivo llega a ser el iniciador principal del ASR su identificador es ID=0.

1. Los cuatro dispositivos esperan la finalización del desvío (drift) del reloj y determinan cuál de ellos debe iniciar el ASR. La figura 5-4 muestra un diagrama general de los pasos de este procedimiento.

2. El dispositivo (interfaz de red o NIC) con dirección física 0 (MAC 0) es el que inicia el algoritmo de sincronización del reloj. El dispositivo genera el “mensaje de inicio” (00) y posteriormente un mensaje (01) el cual se llama “mensaje de sincronización” para que todos los máquinas inicien con él (01).

3. El dispositivo con dirección MAC 0 envía el mensaje (01) para que los demás dispositivos inicien la sincronización de sus relojes.

4. Los otros dispositivos en su interfaz de red realizan la lectura del campo de Longitud/Tipo de cada frame y verifican el valor decimal 1999.

   Si el valor es el 1999, entonces realizan la verificación de la información contenida en el campo de datos y se comprueba que existe el valor (01) para sincronizar y preparar el mensaje (10) denominado “mensaje de enterado”.

   Se guardan la dirección MAC del origen del mensaje que es el iniciador del algoritmo (ID=0). Si el valor longitud/tipo no contiene el valor 1999, deja pasar el frame para que el procesamiento lo realice el módulo wishbone.

5. Todos los máquinas, excepto MAC 0, envían mensajes de tipo broadcast para que se confirme la instrucción del mensaje de sincronización (10).

6. Una vez que se recibieron todos los mensajes (n-1 mensajes) del tipo (10), los dispositivos obtienen las direcciones MAC del transmisor y proceden a ordenar las direcciones físicas para obtener su identificador (ID). Esto es para enviar su mensaje de tiempo de manera ordenada. El algoritmo de ordenamiento que se utilizó fue Radix Sort. Se realizó una modificación a este algoritmo con respecto al Radix Sort original:

   Ordena todos los dispositivos excepto a quien inició el algoritmo, debido a que éste por defecto tiene el identificador cero (ID=0). Por ello se encuentra al inicio de la pila de direcciones MAC una vez terminado el ordenamiento.

7. Cada dispositivo contiene un identificador para que reconozca el momento en el cual debe enviar su tiempo local.

8. El dispositivo con ID=0 se auto envía un mensaje de tipo SEND (mensaje 11) para iniciar el intercambio de los tiempos locales.

9. Se auto envía el mensaje 11 y confirmar que es el identificador cero (ID=0), genera un frame con el tiempo local y envía un mensaje de tipo broadcast para que todos los dispositivos conozcan su tiempo.

10. Los demás dispositivos obtienen el valor del frame recibido, calculan la diferencia (estimado) con su respectivo tiempo local, actualizan su contador (aumentando el valor a más uno) y realizan la comparación con su identificador (ID). En caso de ser igual (ID=1), el dispositivo se envía un mensaje de tipo send (mensaje 11) y envía su tiempo local en modo brodacast.

11. Al auto-enviarse el mensaje 11 y confirmar que es el identificador uno (ID=1), genera un frame con el tiempo local y envía un mensaje de tipo broadcast para que todos los dispositivos conozcan su tiempo.

12. Los demás dispositivos obtienen el valor del frame recibido, calculan la diferencia (estimado) con su respectivo tiempo local, actualizan su contador (aumentando el valor a más uno) y realizan la comparación con su identificador (ID). En caso de ser igual (ID=2), el dispositivo se envía un mensaje de tipo send (mensaje 11) y envía su tiempo local en modo brodacast.

13. Al auto-enviarse el mensaje 11 y confirmar que es el identificador dos (ID=2), genera un frame con el tiempo local y envía un mensaje de tipo broadcast para que todos los dispositivos conozcan su tiempo.

14. Los demás dispositivos obtienen el valor del frame recibido, calculan la diferencia (estimado) con su respectivo tiempo local, actualizan su contador (aumentando el valor a más uno) y realizan la comparación con su identificador (ID). En caso de ser igual (ID=3), el dispositivo se envía un mensaje de tipo send (mensaje 11) y envía su tiempo local en modo brodacast.

15. Al auto-enviarse el mensaje 11 y confirmar que es el identificador tres (ID=3), genera un frame con el tiempo local y envía un mensaje de tipo broadcast para que todos los dispositivos conozcan su tiempo.

16. Los demás dispositivos obtienen el valor del frame recibido, calculan la diferencia (estimado) con su respectivo tiempo local, actualizan su contador (aumentando el valor a más uno) y realizan la comparación con el valor n-1. En caso de ser igual, obtengan el promedio de los estimados para que posteriormente calculen el ajuste. El ajuste es el valor correspondiente al desvío promedio con respecto a los demás procesos (n-1 procesos). Con esto finaliza la sincronización del reloj al menos 2δ de tiempo y que otro dispositivo inicie la sincronización debido a que ha pasado determinado tiempo para generar de nuevo el algoritmo.

Conocido una vez el desarrollo del algoritmo y la secuencia de los mensajes a enviar, se implanta hardware usando el dispositivo de Xilinx llamado Virtex-II PRO. El dispositivo contiene una gran cantidad de periféricos para realizar cualquier simulación de hardware conocido

(latchs, flip-flops, DSP, procesadores, memorias, interfaz de red, entre otros) y cuenta con una gran cantidad de recursos incluidos o bahías para su expansión.

En la figura 5-3 se muestra la tarjeta que se utilizó para el desarrollo de la tesis.

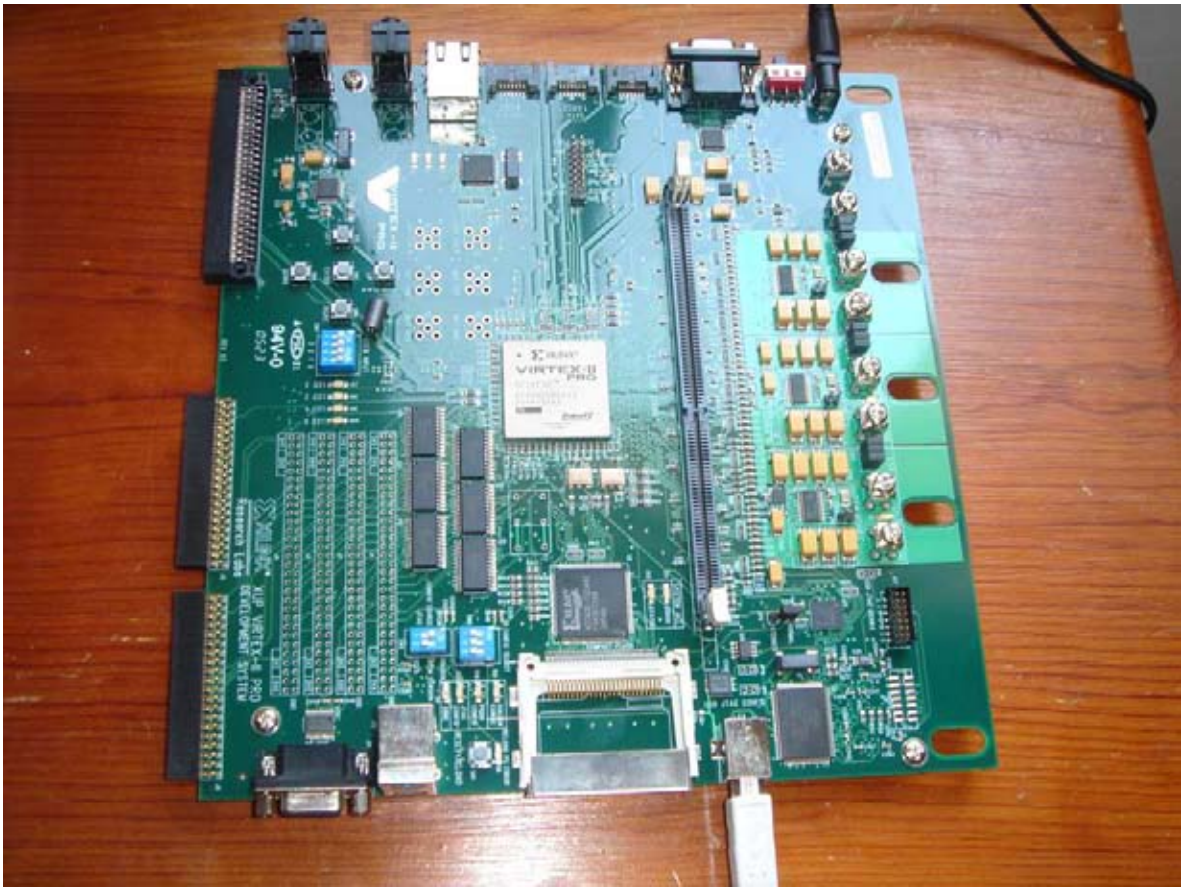
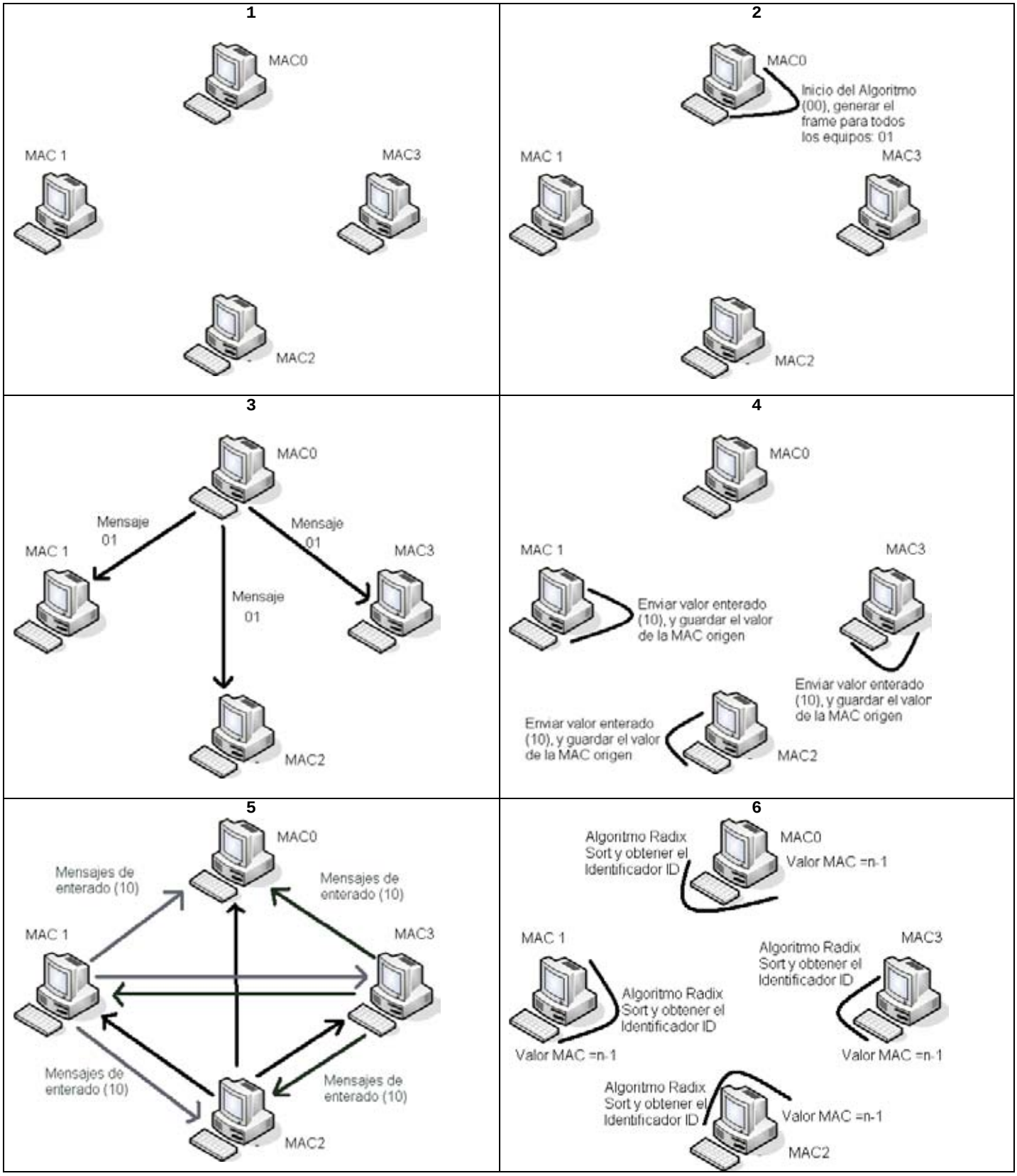


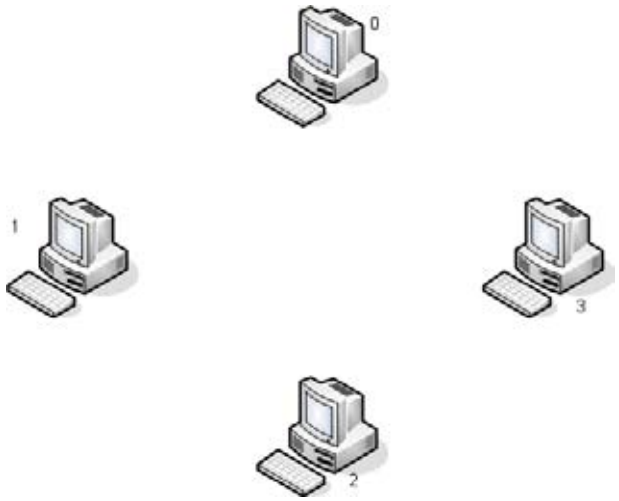
FIGURA 5-3. TARJETA VIRTEX-II PRO, DONDE SE REALIZÓ EL DESARROLLO DEL ASR

No se utilizaron todos los recursos de la tarjeta Xilinx como bus ide, memorias de expansión, memoria ram o el controlador vga. El procesador, la memoria interna y Ethernet del Virtex II son esenciales en el ASR y en el Ethernet MAC IP Core.

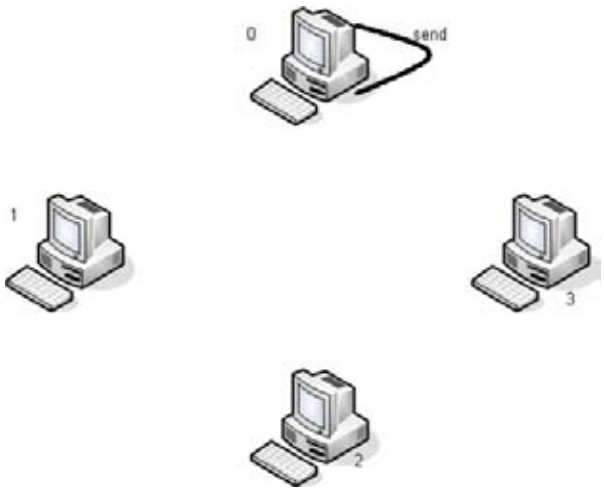
Se muestran en la figura 5-4 el diagrama de secuencia y los mensajes generados a partir de los puntos anteriores para el algoritmo de sincronización del reloj.


1
MAC0
MAC 1
MAC3
MAC2
2
MAC0
Inicio del Algoritmo (00), generar el frame para todos los equipos: 01
MAC 1
MAC3
MAC2
3
MAC0
Mensaje 01
Mensaje 01
MAC 1
MAC3
Mensaje 01
MAC2
4
MAC0
MAC 1
MAC3
Enviar valor enterado (10), y guardar el valor de la MAC origen
Enviar valor enterado (10), y guardar el valor de la MAC origen
Enviar valor enterado (10), y guardar el valor de la MAC origen
MAC2
5
MAC0
Mensajes de enterado (10)
Mensajes de enterado (10)
MAC 1
MAC3
Mensajes de enterado (10)
Mensajes de enterado (10)
MAC2
6
Algoritmo Radix Sort y obtener el Identificador ID
MAC0
Valor MAC =n-1
MAC 1
MAC3
Algoritmo Radix Sort y obtener el Identificador ID
Algoritmo Radix Sort y obtener el Identificador ID
Valor MAC =n-1
Valor MAC =n-1
Algoritmo Radix Sort y obtener el Identificador ID
Valor MAC =n-1
MAC2

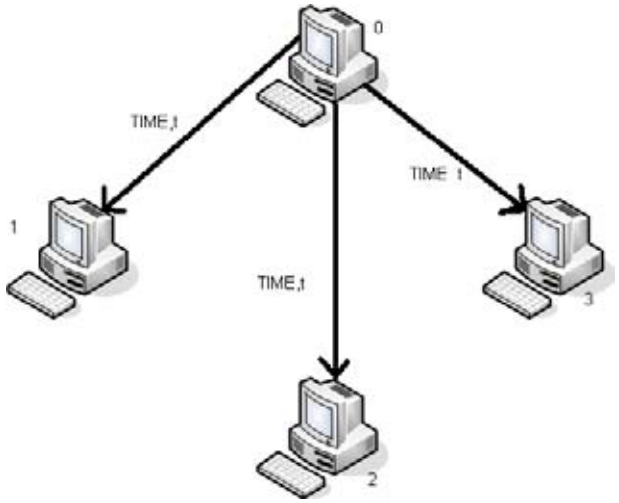
7


0
1
3
2

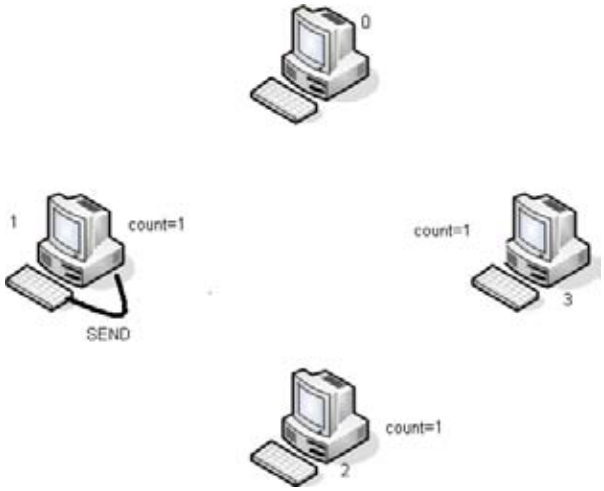
8


0
send
1
3
2

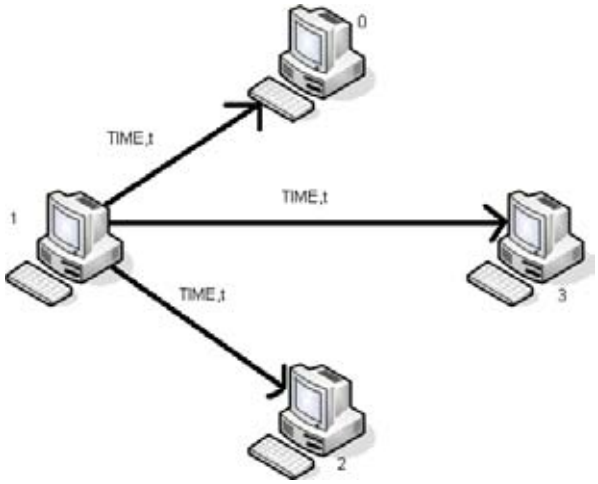
9


0
TIME,t
TIME ,t
1
TIME,t
3
2

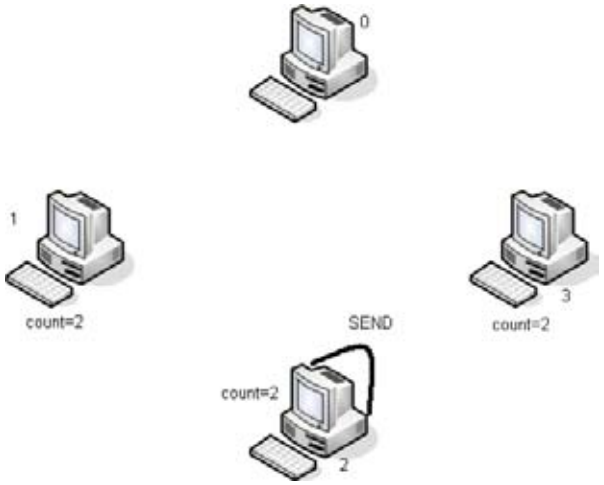
10


0
1
count=1
count=1
3
SEND
count=1
2

11


0
TIME,t
TIME,t
1
TIME,t
3
2

12


0
1
count=2
SEND
3
count=2
count=2
2

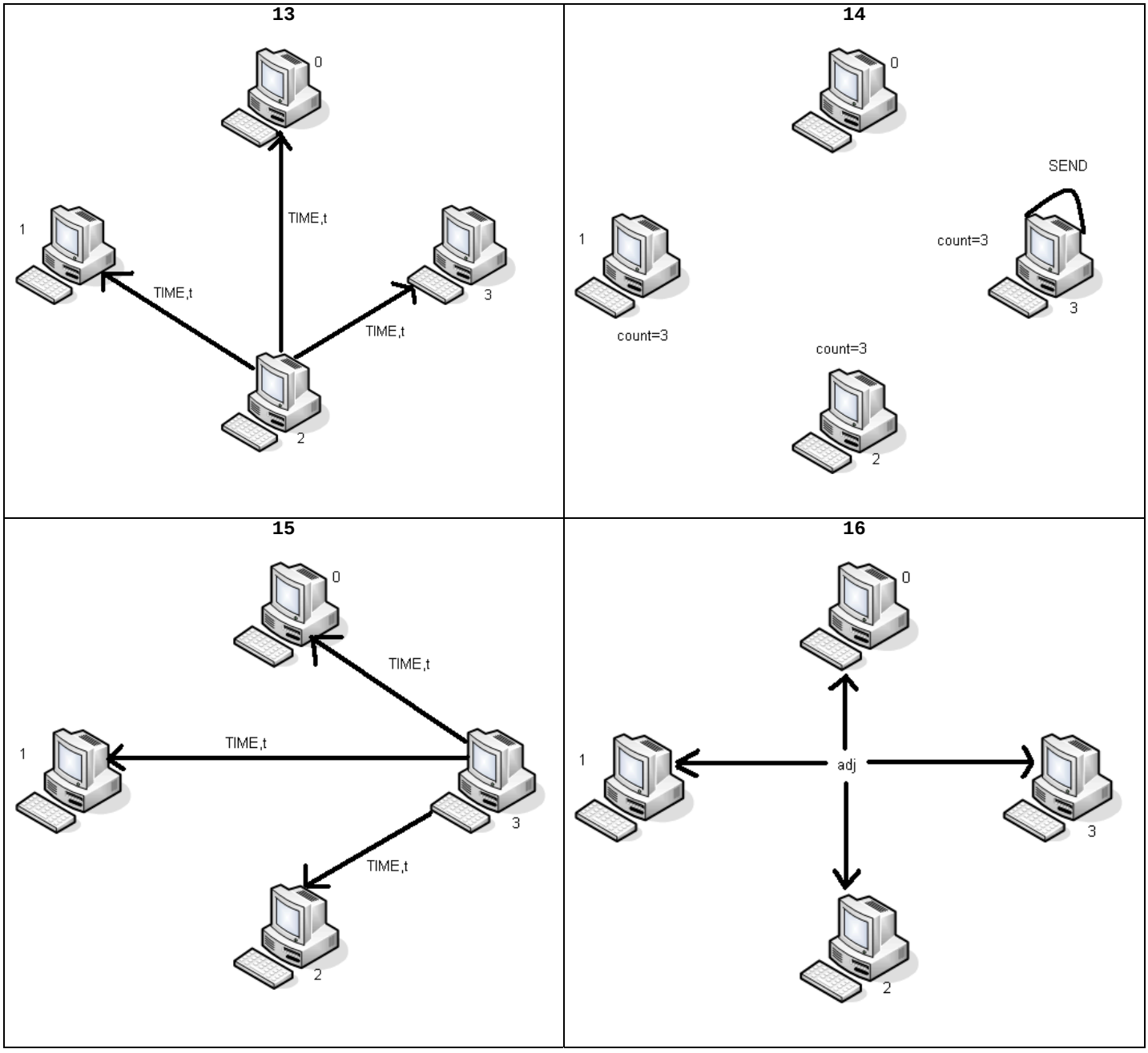


FIGURA 5-4. DIAGRAMA DE SECUENCIA DEL ASR

Finalizando la descripción del algoritmo junto con las iteraciones se muestra a continuación el diagrama de flujo del algoritmo de sincronización del reloj en la figura 5-5:

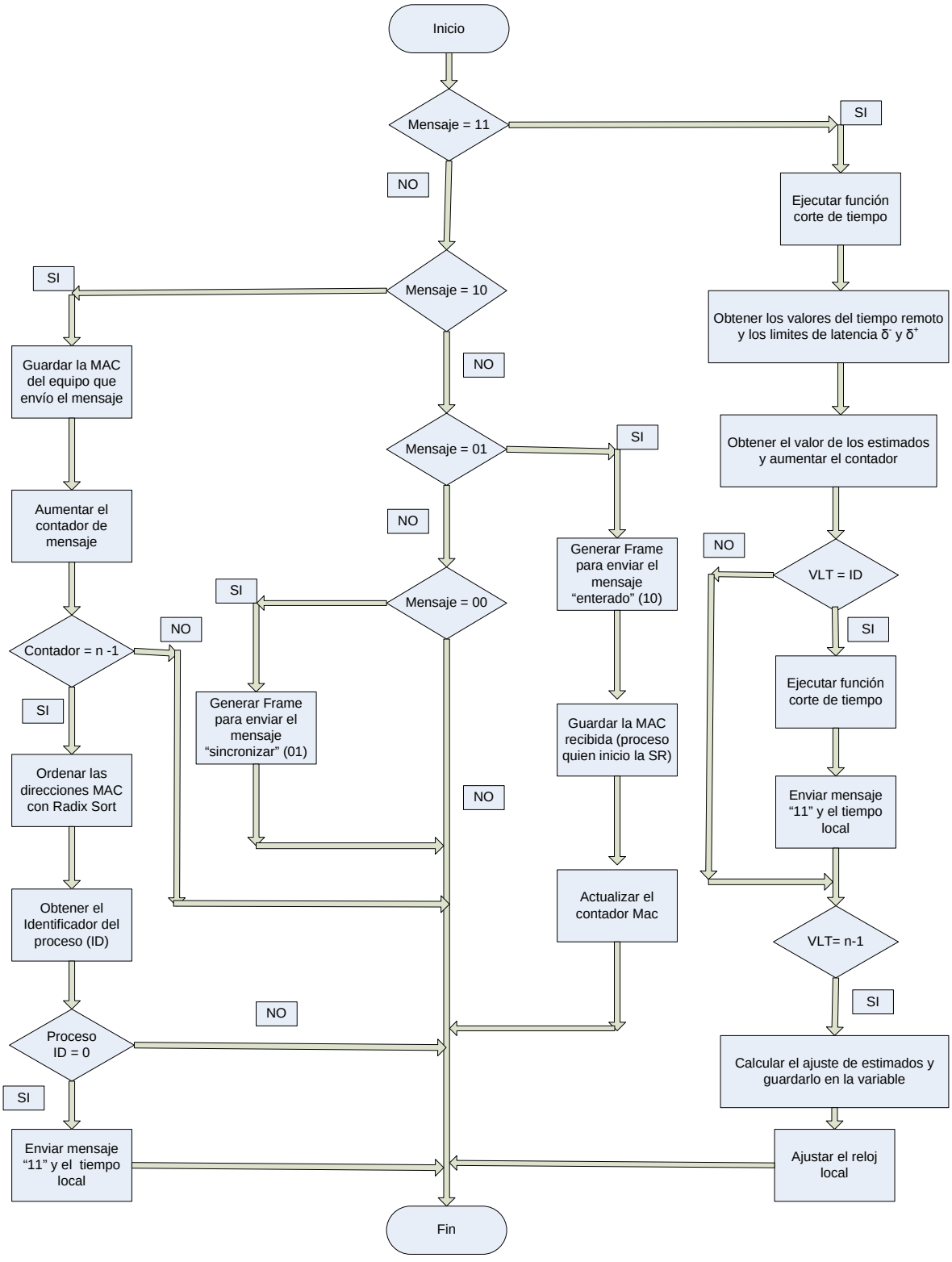


FIGURA 5-5. DIAGRAMA DE FLUJO DEL ASR PARA SISTEMAS QUE UTILIZAN BROADCAST

Se realiza la conversión del algoritmo de pseudocódigo al lenguaje verilog, el cual se muestra en la figura 5-6:

```
input [1:0]  bandera; //00=>Inicio, 01=>Sincronizar,10=>Enterado, 11=>Tiempo
input [47:0] MACIn;
input [63:0] timeRe;

register [63:0] horaLocal;   //La hora local
register [63:0] horaRe;      //La hora remota
integer VLT;                 //Contador de los estimados generados
integer ID;                  //Ese valor es el que se toma de la MAC al ordernarlo.
integer n=4;                 // no. de dispositivos conectados
integer adjTime;             //El ajuste del tiempo
integer [3:0] estimados;
register [63:0] ajuste;
register [47:0] localMAC;
integer valorMAC;
register deltaMenos=60000;
register deltaMas=1000000;
//Pila donde se guardan las MAC's de los participantes de la sincronización del reloj
register [4:0]  stackMac [47:0];

output [47:0] MACOut;
output [1:0] mensajeEx;

function [1:0] mensaje;
  input [1:0] b;
  if(mensaje == 00)
  begin
    mensajeEx=01;
    MACOut=localMAC;
  end
  else if(mensaje == 01)
  begin
          mensaje=10;
     valorMAC=1;
          mensajeEx=10;
     MACOut=localMAC;
  end
  else if(mensaje == 10)
  begin
          stackMac[valorMAC]=MACIn;
          valorMAC=valorMAC+1;
          if(valorMAC == n-1)
     begin
           assign totalMAC= ordenarMAC(stackMac);
            for(a=0;a<=4;a=a+1)
              if(localMAC == stackMAC[a])
                begin
                      ID=a;
                end
            end
            if( ID == 0)
               begin
                 c=deltaMenos;
                      always begin
                       if(c)
                         begin
                           c=c-1;
                    end
```

```
                    end
                    mensajeEx=10;
              MACOut=localMAC;
                 end
              end
        end
        else if(mensaje == 11)
        begin
          horaRe=timeRe;
          estimados[vLT]=(remote_time-localtime+((deltaMenos-deltaMas)/2);
          VLT=VLT+1;
          if(VLT == ID)
          begin
              for(int i=0;i<deltaMenos;i=i+1)
                begin
                end
              mensajeEx=11;
          end
          if(VLT == n-1)
          begin
              for(i=0;i<=4;i=i+1)
                 adjTime=adjTime+estimados[i];
              end
          ajuste=adjTime/n;
          end
        end
      endfunction
```

FIGURA 5-6. CÓDIGO DEL ASR EN LENGUAJE VERILOG

5.3 Módulo del Algoritmo de Ordenamiento (Radix Sort) para las MAC's

El módulo del algoritmo de ordenamiento para las direcciones físicas de la interfaz de red (mejor conocidas como direcciones MAC) es de gran importancia para proporcionar orden al algoritmo de sincronización del reloj debido a que sin él, todos los dispositivos enviarían al mismo tiempo "su tiempo local" y las NIC's remotas podrían perder los paquetes o simplemente ignorarlos. Por consiguiente, se necesita un algoritmo de ordenamiento para facilitarle al ASR el envío ordenado de los mensajes y utilizar las direcciones MAC para este fin.

Existen muchos algoritmos de ordenamiento:

- Heap sort,
- Quick sort,
- Merge sort,
- Bubble sort,
- Sucket sort,
- Radix sort,
- Bitonic sort,
  entre otros

Se evaluaron los algoritmos anteriores y se escogió el algoritmo de radix sort principalmente por tres razones:

1. Radix Sort funciona bien para valores conocidos, como un valor determinado con respecto al rango (de cero a nueve o de 1 a 8, por ejemplo) y en valores de magnitud de menor a mayor significado (ejemplo unidades, decenas, centenas, millares, etc.).

2. El número de las direcciones MAC son valores fijos y tiene su representación hexadecimal: AA-BB-CC-DD-EE-FF, pero al considerarse que las operaciones se realizan en la interfaz de red, esta dirección se convierte en binario de 48 bits. Por lo que se hace el recorrido sobre los 48 bits.

3. Solamente hace una comparación por iteración debido a que su rango es entre 0 y 1.

A continuación se describe el funcionamiento del algoritmo de radix sort:

a) Ordena de manera ascendente (el rango es de cero a nueve) el primer valor en cuanto a la magnitud, en este caso son las unidades.

b) Una vez realizado el ordenamiento en las unidades, realiza el mismo criterio pero en el segundo valor que son las decenas, acabando éste, en las centenas y así sucesivamente hasta que termine considerando el valor máximo de la magnitud.

Se muestra el funcionamiento del algoritmo radix sort para ordenar las direcciones MAC usando las magnitudes 0 y 1 y el rango de 48 bits, aunque solamente se observan las primeras cuatro iteraciones que realiza éste:

Las variables para el algoritmo Radix Sort son:

MAC[0]:000000000000001000101111110101010001100011110000----- 00023fd518f0
MAC[1]:000000001001000010010110101110000111001000011000----- 009096bc7218
MAC[2]:000000000001010100000000000001011100000010001010----- 00150005c08a
MAC[3]:000000001100000010011111101111001101011110101101----- 00c09fbcd7ad

Se inicia el algoritmo

========== PASADA NÚMERO: 0 ==========

- **Magnitud: 48 bits**
- **Rango: 0 y 1**

En la primera iteración en magnitud (unidades), realiza la comparación del valor si es igual a 1 (rango), si lo es, lo guarda en un arreglo temporal 1, si no (por lo que entonces es cero) lo envía al arreglo temporal 2. Acabando la primera iteración une los dos arreglos para realizar la siguiente.

*MACtemp[0]:000000000000001000101111110101010001100011110000**0**-----00023fd518f0*
*MACtemp[1]:000000001001000010010110101110000111001000011000**0**----009096bc7218*
*MACtemp[2]:000000000001010100000000000001011100000010001010**0**----00150005c08a*
*MACtemp[3]:000000001100000010011111101111001101011110101101**1**----00c09fbcd7ad*

========== PASADA NÚMERO: 1 ==========

Segunda iteración en magnitud (decenas), realiza la comparación del valor si es igual a 1 (rango), si lo es, lo guarda en un arreglo temporal 1, si no (por lo que entonces es cero) lo envía al arreglo temporal 2. Acabando la segunda iteración une los dos arreglos para realizar la siguiente.

MACtemp[0]:00000000000000100010111111010101000110001111000----00023fd518f0
MACtemp[1]:00000000100100001001011010111000011100100001100----009096bc7218
MACtemp[2]:00000000110000001001111110111100110101111010111----00c09fbcd7ad
MACtemp[3]:00000000000101010000000000000010111000000100010----00150005c08a

========== PASADA NÚMERO: 2 ==========

Tercera iteración en magnitud (centenas), realiza la comparación del valor si es igual a 1 (rango), si lo es, lo guarda en un arreglo temporal 1, si no (por lo que entonces es cero) lo envía al arreglo temporal 2. Acabando la tercera iteración une los dos arreglos para realizar la siguiente.

MACtemp[0]:00000000000000100010111111010101000110001111000----00023fd518f0
MACtemp[1]:00000000100100001001011010111000011100100001100----009096bc7218
MACtemp[2]:00000000000101010000000000000010111000000100010----00150005c08a
MACtemp[3]:00000000110000001001111110111100110101111010111----00c09fbcd7ad

========== PASADA NÚMERO: 3 ==========

Cuarta iteración en magnitud (millares), realiza la comparación del valor si es igual a 1 (rango), si lo es, lo guarda en un arreglo temporal 1, si no (por lo que entonces es cero) lo envía al arreglo temporal 2. Acabando la cuarta iteración une los dos arreglos para realizar la siguiente.

MACtemp[0]:00000000000000100010111111010101000110001111000----00023fd518f0
MACtemp[1]:00000000100100001001011010111000011100100001000----009096bc7218
MACtemp[2]:00000000000101010000000000000010111000000100010----00150005c08a
MACtemp[3]:00000000110000001001111110111100110101111010101----00c09fbcd7ad

Y así sucesivamente hasta que termina en la magnitud número 48, el cual se ordenada de la siguiente manera:

========== PASADA NÚMERO: 47 ==========

MACtemp[0]:00000000000000100010111111010101000110001111000----00023fd518f0
MACtemp[1]:00000000001010100000000000001011100000010001010----00150005c08a
MACtemp[2]:00000001001000010010110101110000111001000011000----009096bc7218
MACtemp[3]:00000001100000010011111101111001101011110101101----00c09fbcd7ad
==========

Se copia el arreglo temporal al arreglo final. El arreglo ordenado es:

MAC[0]:00000000000000100010111111010101000110001111000-----00023fd518f0
MAC[1]:00000000001010100000000000001011100000010001010-----00150005c08a
MAC[2]:00000001001000010010110101110000111001000011000-----009096bc7218
MAC[3]:00000001100000010011111101111001101011110101101-----00c09fbcd7ad

Recordando que el arreglo final del dispositivo tiene un valor de más, se guarda al inicio de la sincronización y es la dirección MAC del dispositivo que inició el ASR (ID=0).

Para ver todas las iteraciones de este ejemplo ir el apéndice A. Debido al proceso de programación, hacer test y descargarlo en la FPGA, se hizo un primer algoritmo en el lenguaje de programación JAVA para obtener resultados rápidos y posteriormente, se transformó el código de Java a código verilog. El primero se encuentra en el apéndice B y el segundo se muestra en la figura 5-7:

```
//Pila donde se guardan las MAC's de los participantes de la sincronización del reloj
register [4:0] stackMac [47:0];
//Pila temporal para el ordenamiento de los valores 0 "cero".
register [4:0] tempMacero [47:0];
//Pila temporal para el ordenamiento de los valores 1 "uno".
register [4:0] tempMacUno [47:0];
// Contadores para el manejo de pilas temporales
integer bx0, bx1;
function ordenarMAC;
  begin
   for(a=0;a<48;a=a+1)
     for(b=0;b<4;b=b+1)
           if([b]stackMac[a])
             begin
               tempMacUno[bx1]=stackMac[b];
               bx1=bx1+1;
        end  //fin "if"
           else
             begin
               tempMacero[bx0]=stackMac[b];
               bx0=bx0+1;
             end //fin else
     end   //fin for "b"
     for(c=0;c<bx0;c=c+1)
           begin
      for(cx=0;cx<48;cx=cx+1)
             begin
             stackMac[c][cx]=tempMacero[c][cx];
         end
      end  //end del copiado anidado
     end   //fin for "c"
     for(d=0;c<bx1;d=d+1)
           begin
      for(dx=0;dx<48;d=d+1)
             begin
             stackMac[d+bx0]=tempMacUno[d];
             end
           end
     end  //fin ciclo for  "d"
   end  //fin for "a"
 end   //end begin
endfunction
```

FIGURA 5-7. ALGORITMO DE ORDENAMIENTO RADIX SORT

5.4 Módulo de espera 2δ en el tiempo

El módulo probabilístico y no controlado para el algoritmo de sincronización del reloj se llama 2δ en el tiempo con respecto al retardo de los mensajes (latencia de los mensajes) cuando se envían por la red. Debido al nuevo modelo que se realizó en la tesis y que existe dificultad para medir los tiempos (mínimo y máximo) de los mensajes, se toma como base la investigación del Austriaco Daniel Albeseder del artículo denominado “Evaluation of Message Delay Correlations in Distributed Systems” [19], para medir el retardo de los mensajes mediante tres criterios y de manera gradual las mediciones con: carga en el CPU, carga en la red y carga en los dos anteriores. Los resultados de los valores

máximos en los mensajes se muestra en la figuras 5-8 como, donde el número de nodos es igual cuatro:

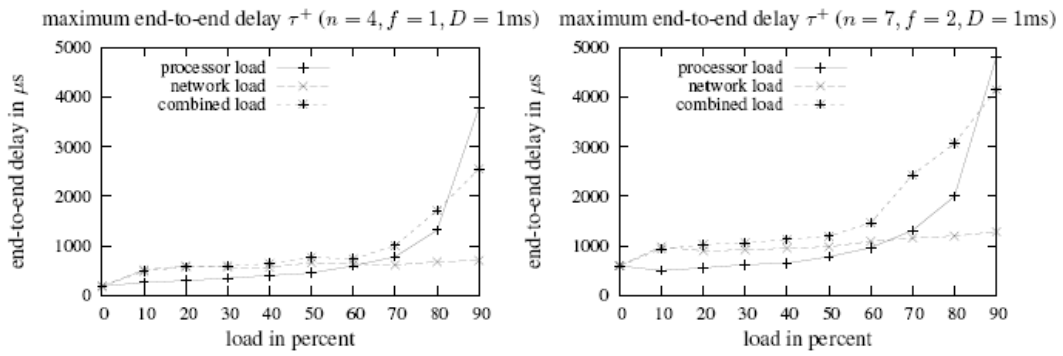


FIGURA 5-8. PORCENTAJES DE LA CARGA MÁXIMA USANDO LOS TRES CRITERIOS [19]

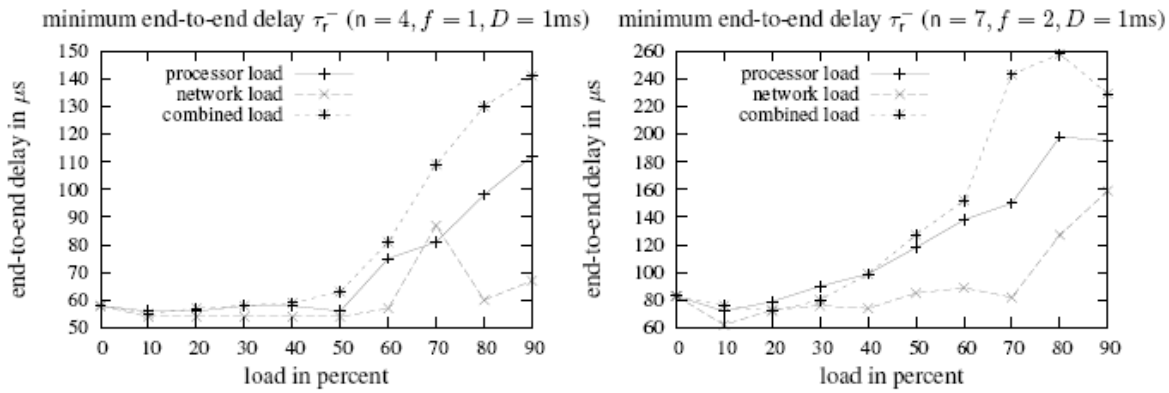


FIGURA 5-9 PORCENTAJES DE LA CARGA MÍNIMA USANDO LOS TRES CRITERIOS [19]

Considerando carga mínima y máxima, el algoritmo no es tolerante a fallas, se tomaron los valores (figura 5-8 y 5-9) con n=4 (número de nodos), los valores seleccionados para δ^- y δ^+ son:

δ^-=60µs y δ^+=1000µs (1ms)

Dichos valores son de suma importancia debido a que, la latencia del mensaje del algoritmo de ASR lo utiliza para tratar de evitar que los mensajes no estén en la cola “demasiado tiempo” o que el algoritmo se ejecute infinitamente debido a la ausencia de mensajes (mensajes perdidos o faltantes). Los valores δ^- y δ^+ están directamente en el código y se encuentran entre las líneas 8 y 10, donde se muestran en la figura 5-10:

8. estimates.add(remote_time-time+ $\frac{\delta^- + \delta^+}{2}$ ***)***

9. if estimates.count=ID

10. send timer(SEND) for time + max($\varepsilon - \delta^- + \mu^+, \mu^+$ ***)***

FIGURA 5-10. SECCIÓN DONDE SE MUESTRA LAS DELTAS (Δ^- Y Δ^+) EN EL ASR

En la línea 8 se utilizan las deltas para calcular el valor de los tiempos estimados de cada uno de los mensajes recibidos con el tiempo remoto, se obtiene la semisuma con las deltas.

La línea 10 acciona el mensaje SEND y utiliza un valor máximo ya sea $\varepsilon - \delta^- + \mu^+, \mu^+$. Debido a que no hay procesamiento en el CPU, las variables μ^- y μ^+ tienen un valor de cero.

Al obtenerse los valores de las deltas, éstos se asignaron como constantes en el módulo de ASR para realizar la misma función que en el algoritmo seleccionado. Al crear este módulo solamente faltan realizar dos para que el algoritmo se integre completamente al Ethernet Mac Core IP, los cuales son:

- Módulo de Gestión de paquetes ASR.
- Módulo de función y corte del reloj.

5.5 Módulo de Gestión de paquetes ASR

El módulo de paquetes ASR se vincula principalmente con los siguientes módulos del MAC IP Core como se muestra en la figura 5-11:

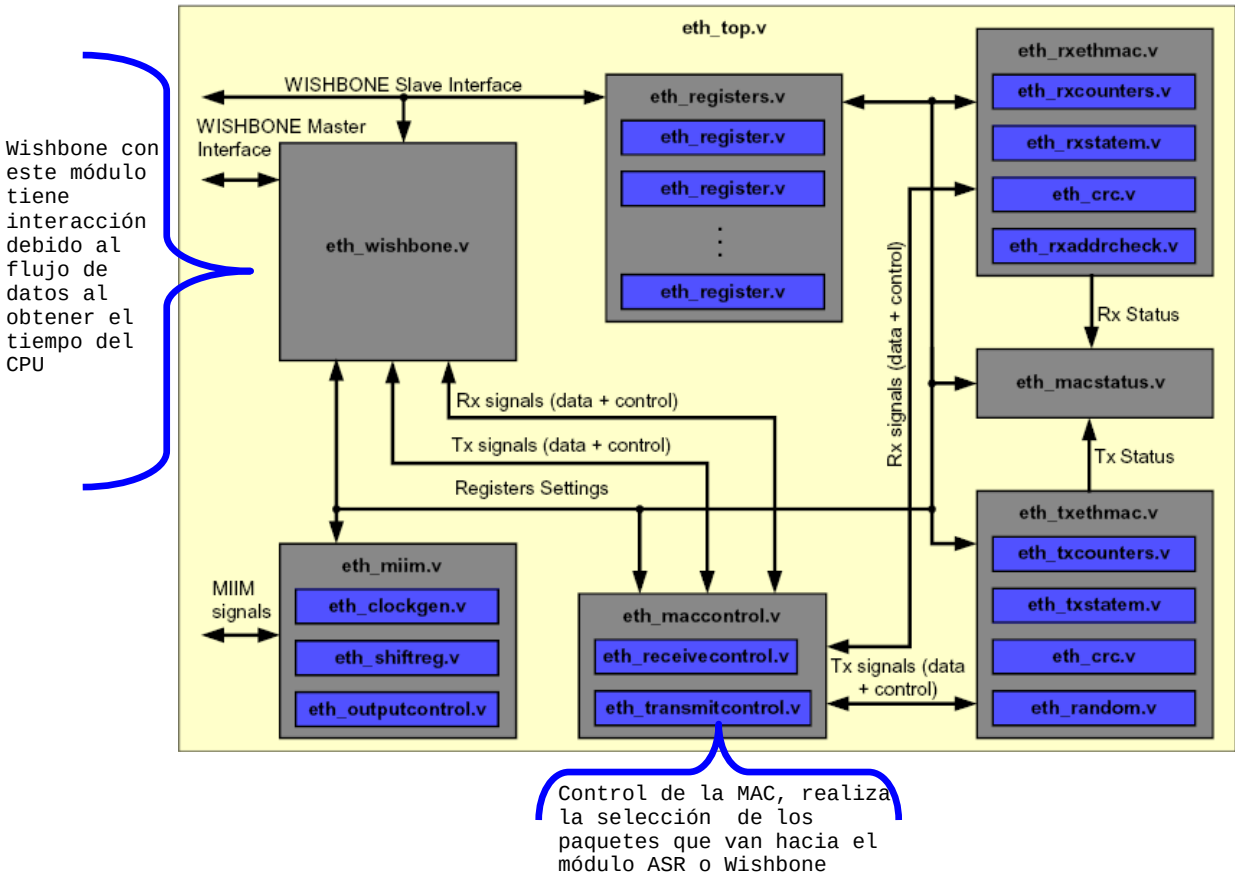


FIGURA 5-11. DIAGRAMA DE ETHERNET MAC IP CORE, SEÑALANDO LOS MÓDULOS INVOLUCRADOS CON EL ASR

El algoritmo de sincronización del reloj se ejecuta en la FPGA, se necesita crear otro módulo donde se vincule con los módulos descritos en la figura 5-11. Se debe contar con los campos que maneja los frames de Ethernet y conocer cuál va a ser el campo o parámetro para crear el frame perteneciente a una *"transmisión normal"* o de tipo ASR. En el capítulo cuatro se analizó e investigó el protocolo Ethernet y se estudiaron todos los campos pertenecientes al frame, los cuales son:

- Preámbulo
- Campo delimitador de inicio de trama (SFD)
- Campos de Dirección
- Campo de Longitud/Tipo
- Campo de Datos y PAD
- Campo de Secuencia de Verificación del Frame
- Campo de extensión

El campo que se puede manejar para utilizarlo en el ASR es el campo de Longitud/Tipo, se considera lo siguiente:

- Si el campo de datos es menor a 1500 bytes, el campo de Longitud/Tipo contiene la cantidad de bytes que vienen en el campo de datos.

- Si el campo de datos es mayor a 1500 bytes, la información del campo Longitud/Tipo contiene el tipo de protocolo de la capa superior para que lo analicen.

Existe la posibilidad de leer el valor de los campos por lo que se considera:

- Aunque el paquete sea menor a 1500 bytes, se tiene que agregar un valor mayor a éste para que no afecte los valores que sean de TX/RX de la PC.

Haciendo la consideración anterior se buscaron los valores superiores a 1500 bytes y se encontró la tabla siguiente:

ETHERTYPE	ORGANIZATION / ADDRESS	PROTOCOL
0000 - 05DC	IEEE802.3 LENGTH FIELD UNITED STATES	PROTOCOL UNAVAILABLE
0101 - 01FF	XEROX (EXPERIMENTAL) WEBSTER NY UNITED STATES	PROTOCOL UNAVAILABLE
0201	XEROX PUP ADDR TRANS WEBSTER NY UNITED STATES	PROTOCOL UNAVAILABLE
0400	NIXDORF PANDERBORN GERMANY	PROTOCOL UNAVAILABLE
0500	UNIVERSITY OF BERKELEY COMPUTER SCIENCE DEPARTMENT 570 EVANS HALL, UNIVERSITY OF CALIFORNIA BERKELEY CA 94720 UNITED STATES	PROTOCOL UNAVAILABLE
0500	SPRITE RPC UNITED STATES	PROTOCOL UNAVAILABLE
600	XEROX NS IDP WEBSTER NY UNITED STATES	PROTOCOL UNAVAILABLE
0660 - 0661	DLOG OLCHING GERMANY	PROTOCOL UNAVAILABLE
0800	XEROX UNITED STATES IPV4	INTERNET PROTOCOL VERSION HORNIG, C., "A STANDARD FOR THE TRANSMISSION OF IP DATAGRAMS OVER ETHERNET NETWORKS," RFC INTERNET SOCIETY, APR. 1984. HTTP://WWW.IETF.ORG/RFC/RFC894.TXT
80FF	NORTEL NETWORKS 8200 DIXIE RD.STE. 100BRAMPTON ONTARIO CANADA	PROTOCOL UNAVAILABLE.
8100	IEEE 802.1 11A POPLAR GROVE SALE CHESHIRE M33 3AX UNITED KINGDOM	IEEE STD 802.1Q – CUSTOMER VLAN TAG TYPE
8145 - 8147	VRIJE UNIVERSITEIT DE BOELELAAN 1081 10181 HV AMSTERDAM NETHERLANDS	ETHERTYPE USED BY THE AMOEBA DISTRIBUTED OPERATING SYSTEM PROTOCOLS
86DD	UNIVERSITY SC ISI IPV6 UNITED STATES	INTERNET PROTOCOL VERSION 6 CRAWFORD, M., "TRANSMISSION OF IPV6 PACKETS OVER ETHERNET NETWORKS," RFC-2464,INTERNET SOCIETY, DEC. 1998. HTTP://WWW.IETF.ORG/RFC/RFC2464.TXT

8847 - 8848	CISCO SYSTEMS 1414 MASSACHUSETTS AVE. BOXBOROUGH MA 01719 UNITED STATES	8847: MPLS (MULTIPROTOCOL LABEL SWITCHING) LABEL STACK -UNICAST REFERENCE: RFC 3032 URL:FTP://FTP.RFC-EDITOR.ORG/IN-NOTES/RFC3032
885B	PHILIPS MEDIZIN SYSTEME BOBLINGEN GMBH HEWLETT-PACKARD-STRASSE 2 BOEBLINGEN 71034 GERMANY	THIS ETHERTYPE IS USED FOR REAL-TIME COMMUNICATION BETWEEN MEDICAL DEVICES
886B	BAY NETWORKSPO BOX 58185 SANTA CLARA CA 95052-8185 UNITED STATES	NORTEL NETWORKS PROPRIETARY PROTOCOL
888E	IEEE 802.1 11A POPLAR GROVE SALE CHESHIRE M33 3AX UNITED KINGDOM	IEEE STD 802.1X - PORT-BASED NETWORK ACCESS CONTROL
88ED	MESHCOM TECHNOLOGIES, INC MERITULLINKATU 1 C HELSINKI 00170 FINLAND	MESHCOM MESH PROTOCOL (MMP). WWW.MESHCOM.COM
88EE	METRO ETHERNET FORUM 19900 MACARTHUR BLVD. SUITE 810 IRVINE CA 92612 UNITED STATES	ETHERNET LOCAL MANAGEMENT INTERFACE (E-LMI). HTTP://WWW.METROETHERNETFORUM.ORG/TECHSPEC.HTM(THE DOCUMENT NUMBER IS "MEF 16.")

Tabla 5.1 Diferentes protocolos contenidos en el campo Longitud/tipo del campo CSMA/CD

Examinando todos los campos de la tabla 5.1 existen muchos valores por encima del valor 1500 y se pueden utilizar para la realización de la sincronización del reloj, sin necesidad de utilizar las capas superiores de TCP/IP. El valor a utilizar es 1999 en decimal (07cf en hexadecimal). Cuando el dispositivo FPGA reconoce el campo 1999, no envía los datos a las capas superiores pues el procesamiento lo realiza el FPGA. Si el otro valor es diferente a 1999, el protocolo Ethernet realiza los pasos habituales.

Para mayor información de los campos a esta tabla, se hace una referencia en el apéndice C.

Los diagramas de flujo para la transmisión y recepción de los datos se muestran en las figuras 5-12 y 5-13:

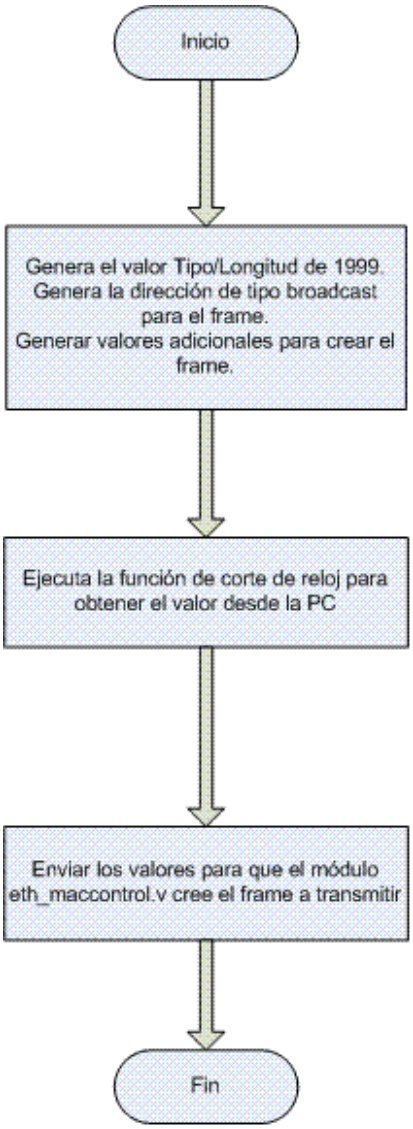


FIGURA 5-12. DIAGRAMA DE FLUJO DEL TRANSMISOR DE FRAMES DE TIPO ASR

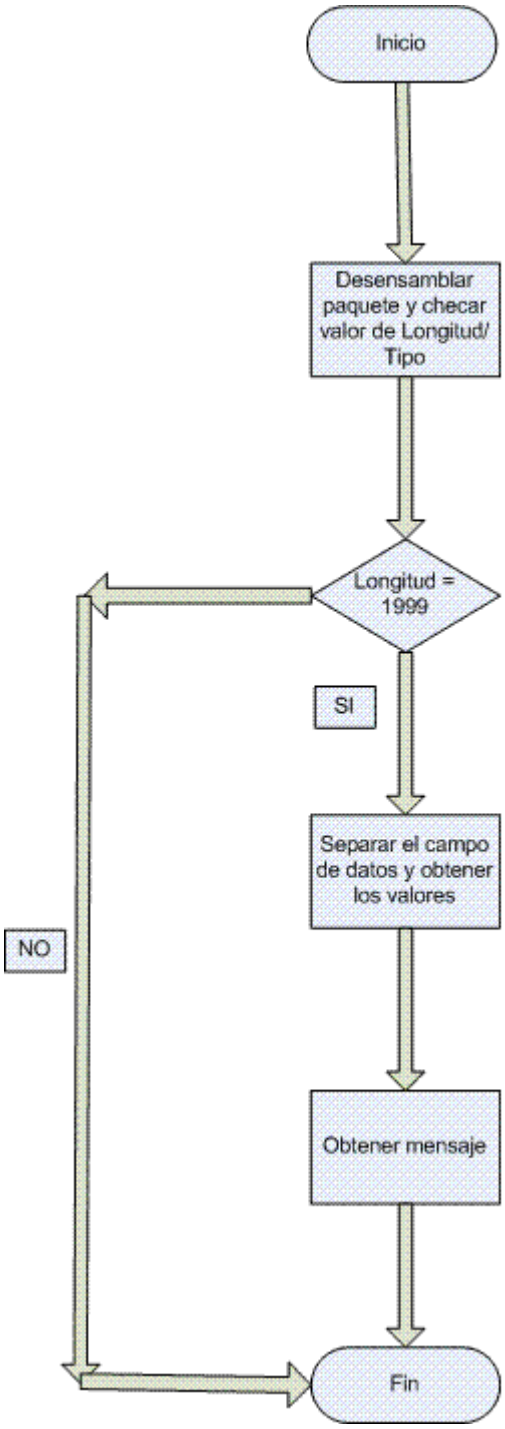


FIGURA 5-13. DIAGRAMA DE FLUJO DEL RECEPTOR DEL ASR

Finalmente el código desarrollado en verilog para estos dos módulos es el siguiente:

Para el módulo de transmisión (figura 5-14):

```
input  [7:0]  TxDataIn;              // Bytes de datos del paquete transmitido
input  [47:0] MAC;                   // Dirección MAC (se obtiene de los registros)
input  [1:0]  mensajeEx;             // Entrada del valor a enviar.
output  [69:0] TxDataOut;            // Bytes de datos del paquete transmitido (a enviar)
output [15:0] LengthType;            // Longitud_Tipo para el Frame CSMA
output broadcast;                    // El valor de envío del broadcast
parameter [15:0] LengthTypeTemp;     // Longitud_Tipo de dato para generarlo 0x07cf
reg  broadcastx;                     // 1 para que el mensaje le llegue a todos.
time reloj;                          // Valor para que guardar el valor del reloj.
reg [1:0] mensaje;                   // El valor del mensaje a enviar
reg [3:0] separador;                 // Separador de campo entre el mensaje
                                     // y el valor del reloj
assign LengthTypeTemp[15:12] = 0000;
assign LengthTypeTemp[11:8]  = 0111;
assign LengthTypeTemp[7:4]   = 1100;
assign LengthTypeTemp[3:0]   = 1111;
assign LengthType= LengthTypeTemp;
assign broadcast = 1;
assign separador = 1111;
assign mensaje = mensajeEx;
assign TxDataOut=  mensaje+separador+reloj;
```

FIGURA 5-14. CÓDIGO DEL MÓDULO DE TRANSMISIÓN EN EL ASR

Para el módulo de recepción (figura 5-15):

```
input  [69:0]  RxDatosIn;            // Bytes de datos del paquete transmitido
input  [15:0] LonType;               // Bytes la longitud/Tipo del Frame
input  [47:0] MACO;                  // Dirección MAC (se obtiene de los registros)
parameter [15:0] LengthTypeTemp;     // Longitud_Tipo de dato de 0x07cf
reg [69:0] Datos;
reg [47:0] MACx;                     //Dirección MAC
reg [15:0] longtype;                 //Registro del tipo de dato
integer    contador;                 //Contador para agregar los datos
output [1:0]  mensajeEx;             //El valor del mensaje Remoto
output [63:0] tiempoRe;              //El tiempo Remoto
output [47:0] MAC;
assign LengthTypeTemp[15:12] = 0000;
assign LengthTypeTemp[11:8]  = 0111;
assign LengthTypeTemp[7:4]   = 1100;
assign LengthTypeTemp[3:0]   = 1111;
assign longtype=LonType;
assign MACx=MAC0;
 if(longtype == LengTypeTemp)
   begin
   if(Datos[69:68] == 11)
     for(i=0;i<70;i=i+1)
        Datos[contador+i]=RxDatosIn[i];
     end
     assign tiempoRe=Datos[63:0];
   end
```

```
 else
  begin
       assign MAC=MACx;
  end
 assign mensajeEx=Datos[69:68];
end
```

FIGURA 5-15. CÓDIGO DEL MÓDULO DE RECEPCIÓN PARA ASR

Con estos códigos el dispositivo FPGA ahora puede realizar la distinción entre los frames del tipo sincronización del reloj y los otros frames, y así el procesamiento y cálculo del algoritmo no lo realiza el CPU sino el dispositivo FPGA y en un futuro lo realizará la NIC de las computadoras.

5.6 Módulo de función de reloj

El módulo de función de reloj es una aplicación que realiza el intercambio de información entre el dispositivo FPGA y el CPU, consiste en dos módulos: LEER y AJUSTAR.

En el módulo leer, el dispositivo FPGA solicita la hora al CPU con base en su reloj interno. Estas lecturas se realizan cuando hay comparaciones con los otros relojes y al calcular los estimados.

En el módulo ajustar, el dispositivo FPGA realiza una escritura del valor de ajuste en el reloj del CPU.

El tamaño del valor a intercambiar es de 64 bits para el ajuste del reloj. El valor se utiliza para los cálculos de los estimados y lograrel ajuste de los relojes, tanto locales como remotos. Es el valor principal en el campo de datos en los frames de tipo Ethernet. Para enviar la función del valor es simple: solamente se realizan comparaciones y ajustes con los dispositivos FPGA remotos.

El ajuste se realiza vía aplicación y esto depende del sistema operativo a utilizar. Lo anterior se realiza en investigaciones posteriores y el intercambio se hace a través de interrupciones (IRQ) en los diferentes rangos.

Conceptualmente, el desarrollo de la aplicación es simple debido a que las funciones de lectura y comparación las realizan los sistemas operativos.

Los diagramas de flujo de las operaciones LEER y AJUSTAR se muestran a continuación en las figuras 5-16 y 5-17 para comprender el intercambio de información FPGA-CPU y viceversa.

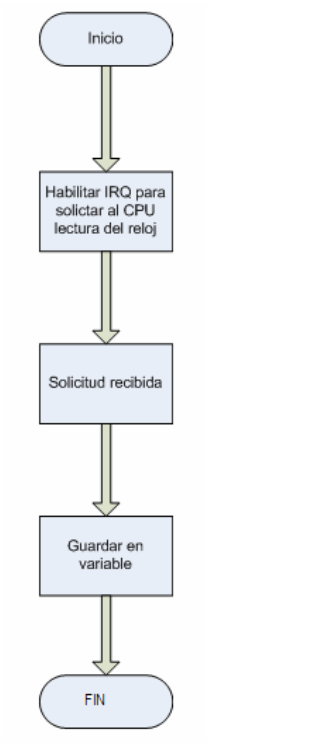


FIGURA 5-16. DIAGRAMA DE FLUJO DE LA LECTURA DEL TIEMPO.

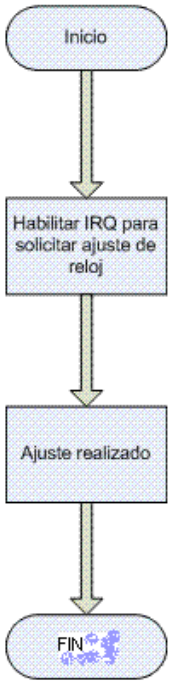


FIGURA 5-17. DIAGRAMA DE FLUJO PARA EL AJUSTE DEL TIEMPO.

La función aunque es simple, se vuelve esencial para el intercambio del tiempo debido a que la lectura y ajuste del reloj a administrar son externo al FPGA, aunque el desarrollo es corto y a nivel de aplicación es la única manera de realizar la transparencia del ajuste del tiempo.

Al desarrollar el módulo y las pruebas a nivel programación con verilog y la gestión de la función del reloj desde el FPGA, los resultados no fueron favorables, ni siquiera se llegó al punto del control de la lectura de los datos desde la PC ya que es un dispositivo externo y no se cuenta con la conexión ni la jerarquía para interactuar directamente con el reloj o el CPU.

Se deseaba que el módulo fuera independiente de la plataforma y sistema operativo para no crear un módulo por cada sistema operativo y diferente plataforma.

El módulo queda fuera del alcance de la aplicación del dispositivo FPGA debido a que es un módulo externo y no se puede integrar directamente en el dispositivo actualm.

5.7 Integración de la implementación

Realizando los módulos para el funcionamiento del ASR, se integró con el módulo Ethernet MAC IP Core como módulo adicional y queda de la siguiente manera:

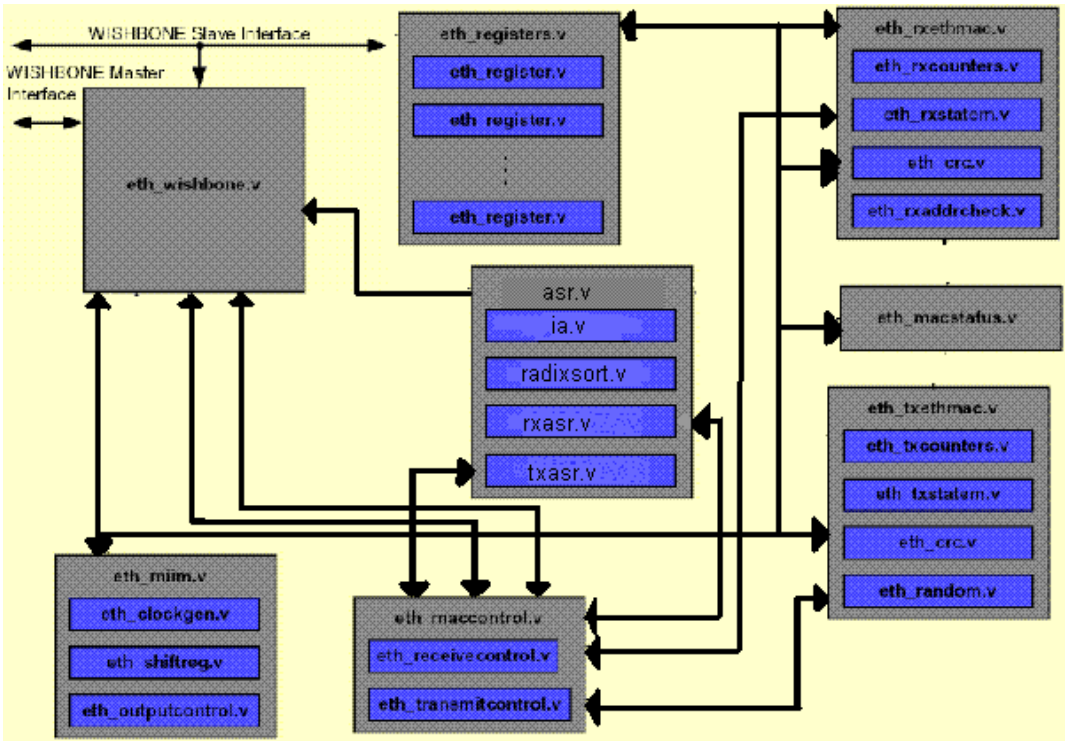


FIGURA 5-18. DIAGRAMA FINAL DE BLOQUES DE ETHERNET MAC IP CORE Y ASR

Analizando el nuevo diagrama, éste realiza lo siguiente:

El archivo ia.v realiza el inicio de la sincronización del reloj y se encuentra ligado a los módulos externos eth_maccontrol.v y eth_wishbone.v

Las operaciones dependen de dos parámetros fundamentales “2δ” de tiempo para esperar los paquetes y el valor “1999” del campo Longitud/Tipo de la cabecera Ethernet.

Los módulos rxasr.v y txasr.v se encargan del flujo de la información y el archivo radixsort.v del ordenamiento de las direcciones físicas.

La etapa de integración se realizó viendo los módulos principales para obtener entradas y salidas, las cuales se realizan en el archivo asr.v

Las pruebas con otros dispositivos FPGA quedan fuera del alcance de esta tesis debido a que no hay dispositivos con características similares para realizar las pruebas de rendimiento e intercambio de paquetes. Con los dispositivos se puede ver el rendimiento del algoritmo y obtener mejores “δ” de tiempo.

Al contar con mejores variables se puede perfeccionar el algoritmo y comprobar que con carga en el CPU o carga en la red, funciona mejor el en esta etapa de desarrollo a nivel hardware. Agregar otro algoritmo para “rutas” óptimas como la “Teoría de la Sincronización” ayuda primordialmente a mejorar en tiempos más cercanos el valor de origen, conociendo la latencia de mensajes.

Se hizo la integración satisfactoria y se comprobó que los algoritmos funcionan de manera unitaria y en algunos casos con la iteracción de otros módulos, pero no se pueden comprobar si el dispositivo receptor recibe los paquetes adecuadamente debido a que al menos se necesitan dos dispositivos de intercambio de mensajes que implementen el algoritmo.

Cabe señalar que al cargar los módulos de asr.v y todo el Ethernet MAC IP Core en el dispositivo Virtex II aún se cuenta con espacio suficiente para realizar módulos de optimización de rutas, esto es importante para un mejor desempeño del algoritmo de sincronización del reloj. Existe mucho campo de investigación para los algoritmos de cómputo distribuido a nivel de hardware en tiempo real.

CONCLUSIONES

La principal contribución de la tesis es desarrollar y realizar pruebas unitarias y con esto comprobar que existe una manera de resolver el intercambio de mensajes para el ASR vía hardware y mejorar el problema de la sincronización del reloj. Se muestra que el CPU no tiene que realizar todo el procesamiento sino que se involucran directamente los periféricos, en este caso la interfaz de red.

Se sugieren nuevos puntos de vista al problema de sincronización del reloj y se presentan un nuevo enfoque y técnicas conjuntas de algoritmos para repartir el funcionamiento global de la sincronización del reloj. Los resultados indican que no es la última solución que existe, ya que al estudiar la capa dos del modelo TCP/IP se analiza que en la sub-capa LLC (Logical Link Control) puede existir sincronización, lo suficiente para que el algoritmo funcione aún mejor y en varios modos: unicast, multicast o broadcast.

Por el lado teórico, se reafirma la importancia de la sincronización del reloj en tiempo real y en la tesis se demuestra la importancia de conocer las variables involucradas en la sincronización del reloj. Cuando se realiza la independencia de las mismas, el algoritmo se puede controlar y determinar sus límites inferior y superior. Sin embargo es indispensable cumplir los requerimientos de Lundelius y Lynch [7] para que el algoritmo de sincronización funcione, esto debe ser en tiempo real por las aplicaciones que actualmente dependen de ello en [1] y [2].

Se ha demostrado con el Modelo Teta [21] y los cálculos hechos por Daniel Albeseder [19] y la modificación de los existentes cálculos para tiempo real por Moser Heinrich y Schmid Ulrich [6] que se necesita controlar los mensajes de tiempo y de sincronización para que funcione adecuadamente el algoritmo.

Con la nueva solución se puede investigar más a fondo si es posible, realizar la sincronización del reloj sin utilizar el CPU y que todo sea por medio de dispositivos lógicos programables (FPGA). Utilizando el reloj interno esto genera una arquitectura independiente de la plataforma (sistema operativo y arquitecturas de procesadores), dependiendo solamente del comportamiento conocido de la red de datos.

Si se logra esta nueva arquitectura de red únicamente con dispositivos FPGA y si se incorpora la optimización de rutas propuesta por Boaz [5], se puede controlar casi totalmente una variable independiente a la sincronización del reloj, la cual es la latencia del mensaje. Además, los nuevos modelos que existirán no serán compatibles con esta arquitectura, por lo cual se necesitará un modelo que pueda leer los mensajes entrantes y salientes para la medición del tiempo y que cumpla con los tiempos establecidos o incluso que estén por debajo de los mismos.

Mejorar el algoritmo es posible y mayormente adaptable si se cambia la programación por parte del dispositivo FPGA a VHDL, debido a que tiene más componentes en hardware y mayor compatibilidad con los mismos.

Otro enfoque nuevo y posible es migrar esta solución a tarjetas de la compañía Intel, ya que éstas son más compatibles con interfaces de red. El algoritmo no es tolerante a fallas, por lo cual se debe profundizar para que sea robusto, y resuelva más de un tipo de fallas (ejemplo bizantino).

Lo anterior comprueba una vez más que al integrar el problema de sincronización del reloj en un hardware diferente al CPU se reducen tiempos y es posible plantear más soluciones a problemas en tiempo real, esto da pauta a un nuevo campo de investigación con periféricos externos, debido a que se trata de un enfoque diferente de interpretación y solución. Las primeras soluciones a este problema con Lamport [9] en 1976, de crear hardware o integrar algoritmos dentro del hardware era muy caro, difícil y poco práctico por el tipo de tecnología con el que se contaba en aquel tiempo. Se ha visto, con las soluciones de Liskov y Linch (decadas 80’s y 90’s), que actualmente la tecnología ha llegado a un grado de reducción de tamaño en cuanto a los dispositivos y de aumento de la capacidad para almacenar información que sería posible contar en estos momentos con la disponibilidad para crear los dispositivos en ambientes menos especializados y abandonar las tareas que solamente las grandes empresas podían realizar.

Usando dispositivos FPGA se cuenta con varias ventajas:

- Creación de código optimizado al programar a bajo nivel en hardware.
- No se necesitan crear los dispositivos hasta que esté completamente realizado el diseño.
- Se pueden realizar pruebas desde los dispositivos y ver el desempeño del tiempo desde el mismo lugar.

La principal desventaja al realizar programación en los dispositivos es el costo por el hardware FPGA, al crear una interfaz de red. Las empresas de comunicaciones invierten en esta tecnología para simular dichos componentes en FPGA y reducir el tiempo y el costo considerablemente. Se inicia un nuevo campo de investigación con periféricos externos usando frames de capas inferiores y nuevos formatos en los mensajes. Con el fin de optimizar el problema de la sincronización del reloj, su solución es simple pero pero no fácil, debido a que necesita retroalimentación presencial y en el ambito de redes es difícil de sincronizar eventos de manera presencial.

Glosario

Atomicidad. Atomicity, En los sistemas de base de datos es una de las propiedades de transacción ACID (Atomicity, Consistency, Isolation, Durability). Una transacción atómica es una serie de operaciones en las bases de datos la cual indica todo lo que ocurre, si falla o no. Esto previene que la base de datos se actualice parcialmente.

Backoff. Mejor conocido como Binary Exponential Backoff o Truncated Binary Exponential Backoff es un algoritmo que usa retroalimentación para multiplicar y realizar el decremento a una tarifa del mismo proceso y así obtener una tarifa aceptable al momento de transmitir. Después de transmitir la señal de interferencia (JAM) se espera una cantidad de tiempo aleatorio tras lo que se intenta transmitir de nuevo, este algoritmo es utilizado en el protocolo CSMA/CD.

Núm Intento(s)	Rango Intervalo	Tiempo (ms)	Retardo Medio (ms)	Retardo Acumulado (ms)
0	0	0	0	0
1	0-1	0-51.2	25.6	25.6
2	0-3	0-153.6	76.8	102.4
3	0-7	0-358.4	179.2	281.6
4	0-15	0-768.0	384.0	665.6
5	0-31	0-1587.2	793.6	1459.2
6	0-63	0-3225.6	1612.8	3072.0
7	0-127	0-6502.4	3251.2	5323.2
8	0-255	0-13056	6528	12851.2
9	0-511	0-26163.2	13081	25932.8
10	0-1023	0-52377.6	26188.8	52121.6
11	0-1023	0-52377.6	26188.8	78310.4
12	0-1023	0-52377.6	26188.8	104499.2
13	0-1023	0-52377.6	26188.8	130688.0
14	0-1023	0-52377.6	26188.8	156876.8
15	0-1023	0-52377.6	26188.8	183065.6
16	Se descarta			

Commit. En el contexto de Ciencias de la computación y la gestión de datos, commit(acción de cometer) se refiere a la idea de hacer que un conjunto de cambios "tentativos, o no permanentes" se conviertan en permanentes. Un uso popular es al final de una transacción de base de datos. En sincronización de relojes y en específico en los artículos de Liskov, hablan de commit en los algoritmos usados para las técnicas de sincronización con los sistemas de archivos Harp.

Correctness. En la teoría de ciencias de la computación, la correctitud (*correctness)* de un algoritmo es afirmar cuando éste se dice que es un algoritmo correcto con respecto a la especificación

CRC. La comprobación de redundancia cíclica (CRC) es un código detector de errores en el envio de paquetes de información a través de un medio, esta función toma como entrada un flujo de datos de cualquier longitud y produce como salida un valor de un cierto tamaño fijo. El término CRC se usa en ocasiones para denotar tanto la función como la función de la producción. Un CRC puede emplearse como una suma de verificación para

detectar la alteración de los datos durante la transmisión o el almacenamiento. La Comprobación Redundante Cíclica es muy popular porque es fácil de poner en práctica en forma binaria, en hardware, el CRC es fácil de analizar matemáticamente, y es particularmente bueno en la detección de los errores causados por el ruido en los canales de transmisión.

Día solar. El intervalo entre dos tránsitos consecutivos del sol.

DIX. Trama de Ethernet versión 2, también conocido como DIX Ethernet (nombrado después por el mayor número de participantes para la creación de la trama en este protocolo: Digital Equipment Corporation, Intel, Xerox) interpreta un campo de 2 bytes siguiendo de las direcciones origen y destino como un EtherType (tipo de Ethernet), identificando así una capa superior al protocolo.

Factor Q. Denominado factor de calidad o factor de mérito, es un parámetro usado en electrónica para comparar la calidad de un sistema resonante. Los sistemas resonantes responden a una frecuencia determinada, llamada frecuencia natural, frecuencia propia o frecuencia de resonancia, mucho más que al resto de frecuencias. El rango de frecuencias a las que el sistema responde significativamente es el ancho de banda, y la frecuencia central es la frecuencia de resonancia eléctrica.

FPGA. Un FPGA (*field programmable gate array*) es un circuito integrado que contiene componentes lógicos programables e interconexiones programables entre ellos. Los componentes lógicos programables pueden ser programados para duplicar la funcionalidad de compuertas lógicas básicas o funciones combinacionales más complejas tales como decodificadores o funciones matemáticas simples. En muchos FPGA, estos componentes lógicos programables (o bloques lógicos, según el lenguaje comúnmente usado) también incluyen elementos de memoria, los cuales pueden ser simples flip-flops o bloques de memoria más complejos.

HDL. Es el acrónimo de Hardware Description Language (Lenguaje de Descripción de Hardware). Son lenguajes de programación que tienen como objetivo programar un circuito electrónico.

El flujo de diseño suele ser típico:

Definir la tarea o tareas que tiene que hacer el circuito.

Escribir el programa usando un lenguaje HDL. También existen programas de captura de esquemas que pueden hacer esto, pero no son útiles para diseños complicados.

Comprobación de la sintaxis y simulación del programa.

Programación del dispositivo y comprobación del funcionamiento. Un rasgo común a estos lenguajes suele ser la independencia del hardware y la modularidad o jerarquía, es decir, una vez hecho un diseño, éste puede utilizarse dentro de otro diseño más complicado y con otro dispositivo compatible.

ICMP. El Protocolo de Mensajes de Control de Internet o ICMP (por sus siglas en inglés, Internet Control Message Protocol) es el subprotocolo de control y notificación de errores para el Protocolo de Internet (IP). Como tal, se usa para enviar mensajes de error, indicando por ejemplo que un servicio determinado no está disponible o que un router o host no puede ser localizado

ICMP difiere del propósito de TCP y UDP ya que generalmente no se utiliza directamente por las aplicaciones del usuario. La única excepción es la herramienta *ping* y *traceroute*.

IGMP. El protocolo de red IGMP se utiliza para intercambiar información acerca del estado de pertenencia entre enrutadores IP que admiten la multidifusión y miembros de grupos de multidifusión. Los hosts miembros individuales informan acerca de la pertenencia de hosts al grupo de multidifusión y los enrutadores de multidifusión sondean periódicamente el estado de la pertenencia.

Kerberos. Es un protocolo de autenticación de redes de computadoras que permite a dos computadoras en una red insegura demostrar su identidad mutuamente de manera segura. Sus diseñadores se concentraron primeramente en un modelo cliente/servidor, y brinda autenticación mutua:

Tanto cliente como servidor verifican la identidad uno del otro.

Los mensajes de autenticación están protegidos para evitar ataques de Replay.

Kerberos se basa en criptografía de clave simétrica y requiere a un tercero en confianza. Además, existen extensiones del protocolo para poder utilizar criptografía de clave asimétrica.

Liveness. Viveza o vivacidad, es una condición que debe llevar a cabo algunas veces, posiblemente un número infinito de veces.

MII. Es una interfaz estándar que se usa para conectar un bloque MAC Fast-Ethernet hacia la capa física (PHY). El MII debe estar conectado a un dispositivo “transceiver” externo vía un conector o simplemente conectar dos chips en la misma tarjeta impresa. Se dice que es independiente del medio debido a que cualquier tipo de dispositivos físicos pueden ser usados sin rediseñar o reemplazar la dirección MAC. El equivalente del MII para otras velocidades es el AUI (para 10 Mbps Ethernet), GII (para Gigabit Ethernet) y XAUI (para 10 Gigabit Ethernet).

NIST. El Instituto Nacional de Estándares y Tecnología, es una agencia de la Administración de Tecnología del Departamento de Comercio de los Estados Unidos. La misión de este instituto es promover la innovación y la competencia industrial en Estados Unidos mediante avances en metrología, estándares y tecnología de forma que mejoren la estabilidad económica y la calidad de vida. Los Laboratorios del NIST en Boulder son mejor conocidos como NIST-F1, un reloj que comparte la distinción de ser el reloj atómico más preciso del mundo junto a uno similar en París, Francia. El NIST-F1 sirve como fuente para la hora oficial de Estados Unidos; de su precisión basada en la frecuencia de resonancia natural del cesio, se basa el NIST para transmitir señales para estación de radio de onda larga llamadas WWVB en Fort Collins, Colorado, y de onda corta

llamadas WWV y WWVH localizadas en Fort Collins y Kekaha, Hawaii respectivamente.

PDU (Protocol Data Units). Se utiliza para el intercambio entre unidades pares, dentro una capa de protocolos de comunicación. Existen dos clases de PDUs:

> PDU de datos, que contiene los datos del usuario final (en el caso de la capa de aplicación) o la PDU del nivel inmediatamente superior.
>
> PDU de control, que sirven para administrar el comportamiento completo del protocolo en sus funciones de establecimiento y ruptura de la conexión, control de flujo, control de errores, etc. No contienen información alguna proveniente del nivel N+1.

Piezoelectricidad. Del griego *piezein*, estrujar o apretar, es un fenómeno presentado por determinados cristales que al ser sometidos a tensiones mecánicas adquieren una polarización eléctrica en su masa, apareciendo una diferencia de potencial y cargas eléctricas en su superficie. Este fenómeno también se presenta a la inversa, esto es, se deforman bajo la acción de fuerzas internas al ser sometidos a un campo eléctrico. El efecto piezoeléctrico es normalmente reversible: al dejar de someter los cristales a un voltaje exterior o campo eléctrico, recuperan su forma.

RTL. El Nivel de Transferencia de Registros RTL (Register Transfer Level) en diseño de circuitos integrados es la descripción para la operación de un circuito digital síncrono. El diseño del RTL, en el comportamiento del circuito está definido en los términos o el flujo de las señales (o transferencia de datos) entre los registros de hardware, y las operaciones lógicas rindiendo estas señales. La abstracción del RTL es usada en los lenguajes descriptores de hardware (HDL’s) como Verilog, y VHDL para crear representaciones de alto nivel de un circuito, desde el cual las representaciones de bajo nivel y el cableado como última instancia del cual puede ser derivado.

SCMP. Synchronized Clock Mesasge Protocol, este protocolo provee un apretón de manos (handshake) entre los que envían y reciben la información usando la sincronización de reloj. La idea de esto es recordar la recepción de todas las comunicaciones recientes. Si un mensaje de un emisor particular es reciente, el receptor tendrá que comparar con la información almacenada y decide si el mensaje está duplicado o no.

Skew. El término skew, se refiere generalmente a una cierta diferencia de un valor previsto u óptimo, es común en telecomunicaciones y matemáticas.

Tránsito del sol. El evento en que el sol alcanza su punto aparentemente más alto en el cielo.

Timestamp. Se refiere al codigo del tiempo o al “sello de tiempo firmado digitalmente” el cual verifica las estampillas para la existencia de documentos firmados o con contenido.

UTC. **Tiempo Universal Coordinado**, también conocido como *tiempo civil*, es la zona horaria de referencia respecto a la cual se calculan todas las

otras zonas del mundo. Es el sucesor del GMT (*Greenwich Mean Time*: tiempo promedio del Observatorio de Greenwich, en Londres) aunque todavía coloquialmente algunas veces se le denomina así. La nueva denominación fue acuñada para eliminar la inclusión de una localización específica en un estándar internacional, así como para basar la medida del tiempo en los estándares atómicos, más que en los celestes.

VHDL. Acrónimo que representa la combinación de VHSIC y HDL, donde VHSIC es el acrónimo de Very High Speed Integrated Circuit y HDL es a su vez el acrónimo de Hardware Description Language. Es un lenguaje usado por ingenieros definido por el IEEE (Institute of Electrical and Electronics Engineers) (ANSI/IEEE 1076-1993) que se usa para diseñar circuitos digitales. Otros métodos para diseñar circuitos son la captura de esquemas (con herramientas CAD) y los diagramas de bloques, pero éstos no son prácticos en diseños complejos. Otros lenguajes para el mismo propósito son Verilog y ABEL.

VLSI. Acrónimo de (Very Large Scale Integration) integración en escala muy grande. Los primeros chips semiconductores contenían sólo un transistor cada uno. A medida que la tecnología de fabricación fue avanzando, se agregaron más y más transistores, y en consecuencia más y más funciones fueron integradas en un mismo chip. El microprocesador es un dispositivo VLSI.

Anexos

Apéndice A. Ordenamiento de Mac’s usando el Algoritmo Radix Sort

```
Las variables para el algoritmo Radix Sort son:
MAC[0]:000000000000001000101111110101010001100011110000-----
MAC[1]:000000001001000010010110101110000111001000011000-----
MAC[2]:000000000001010100000000000001011100000010001010-----
MAC[3]:000000001100000010011111101111001101011110101101-----
Iniciamos el algoritmo
==================  PASADA NUMERO: 0  ==================
MACtemp[0]:000000000000001000101111110101010001100011110000|0|-----
MACtemp[1]:000000001001000010010110101110000111001000011000|0|-----
MACtemp[2]:000000000001010100000000000001011100000010001010|0|-----
MACtemp[3]:000000001100000010011111101111001101011110101101|1|-----
================================================
==================  PASADA NUMERO: 1  ==================
MACtemp[0]:000000000000001000101111110101010001100011110000|0|0-----
MACtemp[1]:000000001001000010010110101110000111001000011000|0|0-----
MACtemp[2]:000000001100000010011111101111001101011110101101|0|1-----
MACtemp[3]:000000000001010100000000000001011100000010001010|1|0-----
================================================
==================  PASADA NUMERO: 2  ==================
MACtemp[0]:000000000000001000101111110101010001100011110000|0|00-----
MACtemp[1]:000000001001000010010110101110000111001000011000|0|00-----
MACtemp[2]:000000000001010100000000000001011100000010001010|0|10-----
MACtemp[3]:000000001100000010011111101111001101011110101101|1|01-----
================================================
==================  PASADA NUMERO: 3  ==================
MACtemp[0]:000000000000001000101111110101010001100011110000|0|000-----
MACtemp[1]:000000001001000010010110101110000111001000011000|1|000-----
MACtemp[2]:000000000001010100000000000001011100000010001010|1|010-----
MACtemp[3]:000000001100000010011111101111001101011110101101|1|101-----
================================================
==================  PASADA NUMERO: 4  ==================
MACtemp[0]:000000000001010100000000000001011100000010001010|0|1010-----
MACtemp[1]:000000001100000010011111101111001101011110101101|0|1101-----
MACtemp[2]:000000000000001000101111110101010001100011110000|1|0000-----
MACtemp[3]:000000001001000010010110101110000111001000011000|1|1000-----
================================================
==================  PASADA NUMERO: 5  ==================
MACtemp[0]:000000000001010100000000000001011100000010001010|0|01010-----
MACtemp[1]:000000001001000010010110101110000111001000011000|0|11000-----
MACtemp[2]:000000001100000010011111101111001101011110101101|1|01101-----
MACtemp[3]:000000000000001000101111110101010001100011110000|1|10000-----
================================================
==================  PASADA NUMERO: 6  ==================
MACtemp[0]:000000000001010100000000000001011100000010001010|0|001010-----
MACtemp[1]:000000001001000010010110101110000111001000011000|0|011000-----
MACtemp[2]:000000001100000010011111101111001101011110101101|0|101101-----
MACtemp[3]:000000000000001000101111110101010001100011110000|1|110000-----
================================================
```

```
==================  PASADA NUMERO: 7  ==================
MACtemp[0]:00000000100100001001011010111000011100l0|0011000-----
MACtemp[1]:00000000000101010000000000000010111000000|1|0001010-----
MACtemp[2]:00000000110000001001111110111100110101l1|0101101-----
MACtemp[3]:00000000000000100010111111010100100011000|1|1110000-----
================================================
==================  PASADA NUMERO: 8  ==================
MACtemp[0]:00000000100100001001011010111000011100l1|0|00011000-----
MACtemp[1]:00000000000101010000000000000010111100000|0|10001010-----
MACtemp[2]:00000000000000100010111111010101000110l0|0|11110000-----
MACtemp[3]:00000000110000001001111110111100110101l1|1|10101101-----
================================================
==================  PASADA NUMERO: 9  ==================
MACtemp[0]:00000000000101010000000000000101110000|0|010001010-----
MACtemp[1]:00000000000000100010111111010101000110|0|011110000-----
MACtemp[2]:00000000100100001001011010111000011100|1|000011000-----
MACtemp[3]:00000000110000001001111110111100110101|1|110101101-----
================================================
==================  PASADA NUMERO: 10  ==================
MACtemp[0]:00000000000101010000000000000010111000|0|0010001010-----
MACtemp[1]:00000000000000100010111111010101000110|0|0011110000-----
MACtemp[2]:00000000100100001001011010111000011110|0|1000011000-----
MACtemp[3]:00000000110000001001111110111100110101|1|1110101101-----
================================================
==================  PASADA NUMERO: 11  ==================
MACtemp[0]:00000000000101010000000000000101110000|0|00010001010-----
MACtemp[1]:00000000100100001001011010111000011110|0|01000011000-----
MACtemp[2]:00000000110000001001111110111100110101|0|11110101101-----
MACtemp[3]:00000000000000100010111111010101000110|1|00011110000-----
================================================
==================  PASADA NUMERO: 12  ==================
MACtemp[0]:00000000000101010000000000000101110|0|000010001010-----
MACtemp[1]:00000000100100001001011010111000011|1|001000011000-----
MACtemp[2]:00000000110000001001111110111100110|1|011110101101-----
MACtemp[3]:00000000000000100010111111010101000|1|100011110000-----
================================================
==================  PASADA NUMERO: 13  ==================
MACtemp[0]:00000000000101010000000000000010111|0|0000010001010-----
MACtemp[1]:00000000110000001001111110111100110|1|1011110101101-----
MACtemp[2]:00000000000000100010111111010101000|0|1100011110000-----
MACtemp[3]:00000000100100001001011010111000011|1|1001000011000-----
================================================
==================  PASADA NUMERO: 14  ==================
MACtemp[0]:00000000000000100010111111010101000|0|01100011110000-----
MACtemp[1]:00000000000101010000000000000010111|1|00000010001010-----
MACtemp[2]:00000000110000001001111110111100110|1|01011110101101-----
MACtemp[3]:00000000100100001001011010111000011|0|11001000011000-----
================================================
==================  PASADA NUMERO: 15  ==================
MACtemp[0]:00000000000000100010111111010101000|0|001100011110000-----
MACtemp[1]:00000000100100001001011010111000011|0|111001000011000-----
MACtemp[2]:00000000000101010000000000000010111|1|100000010001010-----
MACtemp[3]:00000000110000001001111110111100110|1|101011110101101-----
================================================
```

```
==================  PASADA NUMERO: 16  ==================
MACtemp[0]:00000000100100001001011010111100|0|0111001000011000-----
MACtemp[1]:00000000110000001001111111011110|0|1101011110101101-----
MACtemp[2]:00000000000000010001011111101010|1|0001100011110000-----
MACtemp[3]:00000000000101010000000000000010|1|1100000010001010-----
================================================
==================  PASADA NUMERO: 17  ==================
MACtemp[0]:00000000100100001001011010111 0|0|0011100100011000-----
MACtemp[1]:00000000110000001001111111011111|0|0110101111010110 -----
MACtemp[2]:00000000000000010001011111101 01|0|1000110001111000 0-----
MACtemp[3]:00000000000101010000000000000001|0|1110000001000101 0-----
================================================
==================  PASADA NUMERO: 18  ==================
MACtemp[0]:00000000100100001001011010111|0|000111001000011000-----
MACtemp[1]:00000000110000001001111110111|1|001101011110101101-----
MACtemp[2]:00000000000000010001011111101|0|010001100011110000-----
MACtemp[3]:00000000000101010000000000000|1|011100000100010 10-----
================================================
==================  PASADA NUMERO: 19  ==================
MACtemp[0]:00000000000000010001011111101|0|101000110001111000 0-----
MACtemp[1]:00000000000101010000000000000|0|101110000010001010-----
MACtemp[2]:00000000100100001001011010111|1|000011100100001100 0-----
MACtemp[3]:00000000110000001001111110111|1|100110101111010110 1-----
================================================
==================  PASADA NUMERO: 20  ==================
MACtemp[0]:00000000000101010000000000|0|010111000000100010 10-----
MACtemp[1]:00000000000000010001011111110|1|010100011000111100 00-----
MACtemp[2]:00000000100100001001011010 1|1|100001110010000110 00-----
MACtemp[3]:00000000110000001001111110 1|1|110011010111101011 01-----
================================================
==================  PASADA NUMERO: 21  ==================
MACtemp[0]:00000000000101010000000000|0|001011100000100010 10-----
MACtemp[1]:00000000000000010001011111 1|0|101010001100011100 00-----
MACtemp[2]:00000000100100001001011010|1|110000111001000011 000-----
MACtemp[3]:00000000110000001001111110|1|111001101011110101 101-----
================================================
==================  PASADA NUMERO: 22  ==================
MACtemp[0]:00000000000101010000000000|0|000101110000001000 10 10-----
MACtemp[1]:00000000100100001001011 01|0|111000011100100001 1000-----
MACtemp[2]:00000000110000001001111111|0|111100110101111010 1101-----
MACtemp[3]:00000000000000010001011111|1|010101000110001111 0000-----
================================================
==================  PASADA NUMERO: 23  ==================
MACtemp[0]:00000000000101010000000 0|0|000010111000000100 10 10-----
MACtemp[1]:00000000100100001001011 0|1|011100001110010000 11000-----
MACtemp[2]:00000000110000001001111 1|1|011110011010111101 01101-----
MACtemp[3]:00000000000000010001011 11|1|101010100011000111 10000-----
================================================
==================  PASADA NUMERO: 24  ==================
MACtemp[0]:00000000000101010000000|0|000001011100000010001010-----
MACtemp[1]:00000000100100001001011|0|101110000111001000011000-----
MACtemp[2]:00000000110000001001111|1|101111001101011110101101-----
MACtemp[3]:00000000000000100010111|1|110101010001100011110000-----
================================================
```

```
==================  PASADA NUMERO: 25  ==================
MACtemp[0]:00000000000101010000000|0|00000010111000000010001010-----
MACtemp[1]:00000000100100001001010|1|01011100001110010000011000-----
MACtemp[2]:00000000110000001001110|1|11011110011010111110101101-----
MACtemp[3]:00000000000000100010110|1|11101010100011100011110000-----
================================================
==================  PASADA NUMERO: 26  ==================
MACtemp[0]:00000000000101010000000|0|00000010111000000010001010-----
MACtemp[1]:00000000100100001001010|1|10101110000111001000011000-----
MACtemp[2]:00000000110000001001110|1|11101111001101011110101101-----
MACtemp[3]:00000000000000100010110|1|11101010100011100011110000-----
================================================
==================  PASADA NUMERO: 27  ==================
MACtemp[0]:00000000000101010000000|0|00000000101110000000010001010-----
MACtemp[1]:00000000100100001001010|0|11010111000011100100011000-----
MACtemp[2]:00000000110000001001110|1|11110111100110101111010110-----
MACtemp[3]:00000000000000100010110|1|11111010101000110001111000-----
================================================
==================  PASADA NUMERO: 28  ==================
MACtemp[0]:00000000000101010000000|0|00000000101110000000010001010-----
MACtemp[1]:00000000000000100010110|0|11111101010100011000111100-----
MACtemp[2]:00000000100100001001010|1|01101011100001110010000011000-----
MACtemp[3]:00000000110000001001110|1|11110111100110101111010110-----
================================================
==================  PASADA NUMERO: 29  ==================
MACtemp[0]:00000000000101010000000|0|00000000001011100000010001010-----
MACtemp[1]:00000000100100001001010|0|10110101110000111001000011000-----
MACtemp[2]:00000000110000001001110|0|11111101111001101011110101101-----
MACtemp[3]:00000000000000100010110|1|01111110101010001100011110000-----
================================================
==================  PASADA NUMERO: 30  ==================
MACtemp[0]:00000000000101010000000|0|00000000000101110000010001010-----
MACtemp[1]:00000000100100001001010|0|01011010111000011100100011000-----
MACtemp[2]:00000000110000001001110|0|01111110111100110101111010110-----
MACtemp[3]:00000000000000100010110|0|10111111010101000110001111000-----
================================================
==================  PASADA NUMERO: 31  ==================
MACtemp[0]:00000000000101010000000|0|00000000000010111000000010001010-----
MACtemp[1]:00000000100100001001010|0|01011111101010100011000111100-----
MACtemp[2]:00000000110000001001110|1|00101101011100001110010000011000-----
MACtemp[3]:00000000100100001001010|1|00111111011110011010111010110-----
================================================
==================  PASADA NUMERO: 32  ==================
MACtemp[0]:00000000000101010000000|0|00101111110101010001100011110000-----
MACtemp[1]:00000000100100001001010|0|10010110101110000111001000011000-----
MACtemp[2]:00000000110000001001110|0|10011111101111001101011110101101-----
MACtemp[3]:00000000000101010000000|1|00000000000010111000000010001010-----
================================================
==================  PASADA NUMERO: 33  ==================
MACtemp[0]:00000000100100001001010|0|01001011010111000011100100011000-----
MACtemp[1]:00000000110000001001110|0|01001111110111100110101111010110-----
MACtemp[2]:00000000000101010000000|0|10000000000000101110000010001010-----
MACtemp[3]:00000000000000101010000|1|00010111111010101000110001111000-----
================================================
```

```
==================  PASADA NUMERO: 34  ==================
MACtemp[0]:0000000010010|0|00100101101011100001110010000110000-----
MACtemp[1]:0000000011000|0|00100111111011110011010111110101101-----
MACtemp[2]:0000000000000|0|10001011111101010100011000111100000-----
MACtemp[3]:0000000000010|1|01000000000000010111000000100010100-----
================================================
==================  PASADA NUMERO: 35  ==================
MACtemp[0]:000000001001|0|000100101101011100001110010000110000-----
MACtemp[1]:000000001100|0|000100111111011110011010111110101101-----
MACtemp[2]:000000000000|0|010001011111101010100011000111100000-----
MACtemp[3]:000000000001|0|101000000000000010111000000100010100-----
================================================
==================  PASADA NUMERO: 36  ==================
MACtemp[0]:00000000110|0|000010011111101111001101011110101101-----
MACtemp[1]:00000000000|0|001000101111110101010001100011110000-----
MACtemp[2]:00000000100|1|000010010110101110000111001000011000-----
MACtemp[3]:00000000000|1|010100000000000010111000000100010100-----
================================================
==================  PASADA NUMERO: 37  ==================
MACtemp[0]:0000000011|0|000001001111110111100110101111010110 1-----
MACtemp[1]:0000000000|0|000100010111111010101000110001111000 0-----
MACtemp[2]:0000000010|0|100001001011010111000011100100001100 0-----
MACtemp[3]:0000000000|0|101010000000000000101110000001000101 0-----
================================================
==================  PASADA NUMERO: 38  ==================
MACtemp[0]:000000000|0|000010001011111101010100011000111100 00-----
MACtemp[1]:000000001|0|010000100101101011100001110010000110 00-----
MACtemp[2]:000000000|0|010101000000000000010111000000100010 10-----
MACtemp[3]:000000001|1|000000100111111011110011010111101011 01-----
================================================
==================  PASADA NUMERO: 39  ==================
MACtemp[0]:00000000|0|000001000101111110101010001100011110000-----
MACtemp[1]:00000000|0|001010100000000000010111000000100010100-----
MACtemp[2]:00000000|1|001000010010110101110000111001000011000-----
MACtemp[3]:00000000|1|100000010011111101111001101011110101101-----
================================================
==================  PASADA NUMERO: 40  ==================
MACtemp[0]:0000000|0|000000100010111111010101000110001111000 0-----
MACtemp[1]:0000000|0|000101010000000000000101110000001000101 0-----
MACtemp[2]:0000000|0|100100001001011010111000011100100001100 0-----
MACtemp[3]:0000000|0|110000001001111110111100110101111010110 1-----
================================================
==================  PASADA NUMERO: 41  ==================
MACtemp[0]:000000|0|000000010001011111101010100011000111100 00-----
MACtemp[1]:000000|0|000010101000000000000010111000000100010 10-----
MACtemp[2]:000000|0|010010000100101101011100001110010000110 00-----
MACtemp[3]:000000|0|011000000100111111011110011010111101011 01-----
================================================
==================  PASADA NUMERO: 42  ==================
MACtemp[0]:00000|0|000000001000101111110101010001100011110000-----
MACtemp[1]:00000|0|000001010100000000000010111000000100010100-----
MACtemp[2]:00000|0|001001000010010110101110000111001000011000-----
MACtemp[3]:00000|0|001100000010011111101111001101011110101101-----
================================================
```

```
==================  PASADA NUMERO: 43  ==================
MACtemp[0]:0000|0|0000000001000101111110101010001100011110000-----
MACtemp[1]:0000|0|0000001010100000000000000101110000000010001010-----
MACtemp[2]:0000|0|0001001000010010110101110000111001000011000-----
MACtemp[3]:0000|0|0001100000010011111101111001101011110101101-----
================================================
==================  PASADA NUMERO: 44  ==================
MACtemp[0]:000|0|0000000000100010111111010101000110001111000-----
MACtemp[1]:000|0|0000000101010000000000000010111000000010001010-----
MACtemp[2]:000|0|0000100100001001011010111000011100100001100-----
MACtemp[3]:000|0|0000110000001001111110111100110101111010110-----
================================================
==================  PASADA NUMERO: 45  ==================
MACtemp[0]:00|0|0000000000010001011111101010100011000111100000-----
MACtemp[1]:00|0|0000000010101000000000000010111000000010001010-----
MACtemp[2]:00|0|0000010010000100101101011100001110010000110000-----
MACtemp[3]:00|0|0000011000000100111111011110011010111101011000-----
================================================
==================  PASADA NUMERO: 46  ==================
MACtemp[0]:0|0|0000000000001000101111110101010001100011110000-----
MACtemp[1]:0|0|0000000001010100000000000001011100000001000101 0-----
MACtemp[2]:0|0|0000001001000010010110101110000111001000011000-----
MACtemp[3]:0|0|0000011000000100111111011110011010111101011000-----
================================================
==================  PASADA NUMERO: 47  ==================
MACtemp[0]:|0|0000000000000100010111111010101000110001111000 0-----
MACtemp[1]:|0|0000000000101010000000000000101110000000100010 10-----
MACtemp[2]:|0|0000000100100001001011010111000011100100001100 0-----
MACtemp[3]:|0|0000000110000001001111110111100110101111010110 1-----
================================================


Copiando arreglo stackMac a totalMac
El arreglo ordenado es:
MAC[0]:0000000000000010001011111101010100011000111100000-----
MAC[1]:0000000000010101000000000000010111000000010001010-----
MAC[2]:0000000010010000100101101011100001110010000110000-----
MAC[3]:0000000011000000100111111011110011010111101011000-----
```

Apéndice B. Programa RadixSort.java

```
public class RadixSortMAC{
/*
0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,1,1,1,1,1,1,0,1,0,1,0,1,0,0,0,1,1,0,0,0,1
,1,1,1,0,0,0,0; 00023fd518f0
0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,1,0,0,1,0,1,1,0,1,0,1,1,1,0,0,0,0,1,1,1,0,0,1,0,0
,0,0,1,1,0,0,0; 009096bc7218
0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,0,0,0,0,0,0,1
,0,0,0,1,0,1,0; 00150005c08a
0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,1,1,1,1,1,1,0,1,1,1,1,0,0,1,1,0,1,0,1,1,1,1
,0,1,0,1,1,0,1; 00c09fbcd7ad
*/
int stackMac[][]
={{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,1,1,1,1,1,1,0,1,0,1,0,1,0,0,0,1,1,0,0,
0,1,1,1,1,0,0,0,0},{0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,1,0,0,1,0,1,1,0,1,0,1,1,1,0,0
,0,0,1,1,1,0,0,1,0,0,0,0,1,1,0,0,0},{0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,0,0,0,0,0,
0,0,0,0,0,0,0,1,0,1,1,1,0,0,0,0,0,0,1,0,0,0,1,0,1,0},{0,0,0,0,0,0,0,0,1,1,0,0,0,0
,0,0,1,0,0,1,1,1,1,1,1,0,1,1,1,1,0,0,1,1,0,1,0,1,1,1,1,0,1,0,1,1,0,1} };
int totalMac[][] = new int[4][48];
int bx0,bx1;

  public void inicia(){
      System.out.println("Las variables para el algoritmo Radix Sort son:");
      imprime(stackMac);
      System.out.println("Iniciamos el algoritmo");
      ordenarMAC(stackMac);
      System.out.println("El arreglo ordenado es:");
      imprime(totalMac);
  }
  public void imprime(int[][] matriz){
      for(int i=0;i<4;i++){
        System.out.print("MAC["+i+"]:");
        for(int j=0;j<48;j++)
          System.out.print(matriz[i][j]);
          System.out.println("----- ");
      }
  }

  public void imprime(int[][] matriz,int indicador){
     for(int i=0;i<4;i++){
        System.out.print("MACtemp["+i+"]:");
        for(int j=0;j<48;j++){
         if(j == (47-indicador))
               System.out.print("|"+matriz[i][j]+"|");
         else
               System.out.print(matriz[i][j]);
       }
        System.out.println("----- ");
     }
  }
  public int[][] copiar(int[][] matrix0,int[][] matrix1, int indice0, int
indice1,int indicador){
       int matriz[][] = new int [4][48];
        for(int c=0;c<indice0;c++){
            for(int cx=0;cx<48;cx++)
              matriz[c][cx]=matrix0[c][cx];
       }
        for(int d=0;d<indice1;d++){
            for(int dx=0;dx<48;dx++)
               matriz[indice0+d][dx]=matrix1[d][dx];
        }
```

```
      System.out.println("Impresion de matriz");
      imprime(matriz,indicador);
      System.out.println("================================================");
      return matriz;
  }

  public void ordenarMAC(int[][] stackMac){
    for(int a=0;a<48;a++){
       System.out.println("==================  PASADA NUMERO :"+a+"
==================");
      int tempMacero[][] = new int[4][48];
      int tempMacUno[][] = new int[4][48];
      for(int b=0;b<4;b++){
         if(stackMac[b][47-a] == 1){
           tempMacUno[bx1]=stackMac[b];
           bx1=bx1+1;
        }
        else{
           tempMacero[bx0]=stackMac[b];
           bx0=bx0+1;
        }
      }
      stackMac=copiar(tempMacero,tempMacUno,bx0,bx1,a);
      bx0=0;
      bx1=0;
    }
    System.out.println("Copiando arreglo stackMac a totalMac");
    totalMac=stackMac;
  }

  public static void main(String args[]){
      RadixSortMAC rsmac = new RadixSortMAC();
      rsmac.inicia();
  }
}
```

Apéndice C. Valores del campo Longitud/Tipo del protocolo Ethernet

ETHERTYPE	ORGANIZATION / ADDRESS	PROTOCOL
0000 - 05DC	IEEE802.3 Length Field, UNITED STATES	Protocol unavailable
0101 - 01FF	Xerox (Experimental) Webster NY UNITED STATES	Protocol unavailable
0201	Xerox PUP Addr Trans Webster NY UNITED STATES	Protocol unavailable
0400	Nixdorf Panderborn GERMANY	Protocol unavailable
0500	University of Berkeley Computer Science Department 570 Evans Hall, University of California Berkeley CA 94720 UNITED STATES	Protocol unavailable
0500	Sprite RPC UNITED STATES	Protocol unavailable
600	Xerox NS IDP Webster NY UNITED STATES	Protocol unavailable
0660 - 0661	DLOG Olching GERMANY	Protocol unavailable
1600	Valid Systems UNITED STATES	Protocol unavailable.
2000 - 207F	LRW Systems Stamford CT UNITED STATES	Protocol unavailable.
3181	VG Data Systems St George's Court Altrincham Chesire WA145UG UNITED KINGDOM	Protocol unavailable.
3C10 - 3C1F	3Com 5400 Bayfront Plaza Santa Clara CA 95052 UNITED STATES	Protocol unavailable.
3C20	3Com 3 Com Corporation 80 Central Street Boxborough MA 01719 UNITED STATES	Protocol unavailable.
4242	PCS Basic Block Protocol UNITED STATES	Protocol unavailable.
4400 - 4409	Computer Generation, Inc. 3855 Presidential Pkwy Atlanta GA 30340 UNITED STATES	Protocol unavailable.
454C	SR Research Ltd. SR Research Ltd. 1472 Thorndyke Crescent Mississauga Ontario L4X 1R3 CANADA	Protocol unavailable.
5000 - 5009	Intel UNITED STATES	Protocol unavailable.
5208	BBN Simnet UNITED STATES	Protocol unavailable.
6000 - 6009	DEC 1925 Andover St., Tewksbury MA 01876 UNITED STATES	Protocol unavailable.
6010 - 6014	3Com Corporation Santa Clara CA UNITED STATES	Protocol unavailable.
6558	Trans Ether Bridging UNITED STATES	Protocol unavailable.
0800	Xerox UNITED STATES IPv4	Internet Protocol Version Hornig, C., “A Standard for the Transmission of IP Datagrams over Ethernet Networks," RFC Internet Society, Apr. 1984. http://www.ietf.org/rfc/rfc894.txt

80FF	Nortel Networks 8200 Dixie Rd.Ste. 100Brampton Ontario CANADA	Protocol unavailable.
8100	IEEE 802.1 11a Poplar Grove Sale Cheshire M33 3AX UNITED KINGDOM	IEEE Std 802.1Q – Customer VLAN Tag Type
8145 - 8147	Vrije Universiteit De Boelelaan 1081 10181 HV Amsterdam NETHERLANDS	EtherType used by the Amoeba Distributed Operating System protocols
86DD	University SC ISI IPv6 UNITED STATES	Internet Protocol Version 6 Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks," RFC-2464,Internet Society, Dec. 1998. http://www.ietf.org/rfc/rfc2464.txt
8847 - 8848	Cisco Systems 1414 Massachusetts Ave. Boxborough MA 01719 UNITED STATES	8847: MPLS (multiprotocol label switching) label stack -unicast reference: RFC 3032 URL:ftp://ftp.rfc-editor.org/in-notes/rfc3032
885b	Philips Medizin Systeme Boblingen GmbH Hewlett-Packard-Strasse 2 Boeblingen 71034 GERMANY	This EtherType is used for real-time communication between medical devices
886b	Bay NetworksPO Box 58185 Santa Clara CA 95052-8185 UNITED STATES	Nortel Networks proprietary protocol
888e	IEEE 802.1 11a Poplar Grove Sale Cheshire M33 3AX UNITED KINGDOM	IEEE Std 802.1X - Port-based network access control
88ed	Meshcom Technologies, Inc Meritullinkatu 1 C Helsinki 00170 FINLAND	Meshcom Mesh Protocol (MMP). www.meshcom.com
88ee	Metro Ethernet Forum 19900 MacArthur Blvd. Suite 810 Irvine CA 92612 UNITED STATES	Ethernet Local Management Interface (E-LMI). http://www.metroethernetforum.org/TechSpec.htm (The document number is "MEF 16.")
8901	Nokia P.O.BOX 370 Espoo FI-00045 Nok, FINLAND	Flow Layer Internal Protocol (FLIP) for inter-unit messaging.
8902	IEEE 802 LAN/MAN Standards Committee 11a Poplar Grove Sale Cheshire M33 3AX UNITED KINGDOM	IEEE 802.1ag Connectivity Fault Management (CFM) protocol. The latest draft can be found here: http://www.ieee802.org/1/files/private/ag-drafts/d7 Username: p8021 Password: go_wildcats
8903	Cisco Systems Inc. 170 W Tasman Drive, San Jose CA 95134, UNITED STATES	DCE
8904	Cisco Systems Inc. 170 W Tasman Drive, San Jose CA 95134, UNITED STATES	BCN (Backward Congestion Notification) data frame tag
8905	Cisco Systems Inc. 170 W Tasman Drive, San Jose CA 95134, UNITED STATES	BCN (Backward Congestion Notification) data frame tag
8906	Cisco Systems Inc. 170 W Tasman Drive, San Jose CA 95134, UNITED STATES	FCoE - Fibre Channel over Ethernet
8907	Zhejiang University Institute of Advanced Process Control,Zhejiang University Building of Industrial Control/Box 1483,Zhejiang University Hangzhou Zhejiang 310027, CHINA	DRP(Distributed Redundancy Protocol) is a high availability Ethernet protocol developed by Zhejiang University. This protocol guarantee short recovery time in case of either link failure or switch failure. And in this protocol, all switches are distributed and equal without master and slave. This protocol is useful for ring/tree/star topology.
8908	Waves Audio LTD Azrilei Center 3, Triangle tower, 32nd floor, Tel Aviv Central 67023, ISRAEL	Following the EtherType are the following fields, 2 bytes each in little-endian order: Workgroup ID, Destination LUN, Source LUN, opcode. The combination of the first three fields along with the Ethernet address is used for classifying packets into flows. The Opcode field determines the structure of

		the information contained in the remaining part of the packet. Detailed information regarding the packet structure for diferent opcode values is not being made public at this time.
8909	Cisco Systems 170 West Tasman Drive, San Jose CA 95134, UNITED STATES	Decline to disclose. Cisco Proprietary.
890a	WIT, 138 Avenue Leon Beranger St-Laurent-Du-Var France 06706 FRANCE	Protocol WIO. For Building Automation
890b	Panduit Corp 17301 Ridgeland Ave, Tinley Park IL 60477, UNITED STATES	Panduit Proprietary Protocol.
8A96 - 8A97	Invisible Software Foster City CA, UNITED STATES	Protocol unavailable.
8F00 - 8FFF	Xerox, UNITED STATES	Protocol unavailable.
9000	Xerox, Webster NY UNITED STATES	Protocol unavailable.
9001 - 9003	3Com	Protocol unavailable.
9c40	Pilz GmbH & Co. KG Felix-Wankel-Str. 2 Ostfildern Baden-Württemberg 73760 GERMANY	SafetyNET p is a deterministic real-time Ethernet for the industrial environment. Established technology from the SafetyBUS p safe bus system was also considered and refined. So SafetyNET p is an Ethernet-based network for industry, which can be used for real-time and safe communication functions. http://www.pilz.com
A580	Siemens Gammasonics UNITED STATES	Protocol unavailable.
C020 - C02F	University of Berkeley UNITED STATES	Protocol unavailable.
C220 - C22F	University of Berkeley UNITED STATES	Protocol unavailable.
FC0F	Senetas Security Pty Ltd Level 1/11 Queens Road Melbourne VICTORIA 3004 AUSTRALIA	Proprietary protocol.
FEA0 - FEAF	NTT Electronics Technology Corp.1-14-5 Kichijoji-Honmachi Musashino-shi Tokyo 180, JAPAN	Protocol unavailable.
FF00 - FF0F	ISC-Bunker Ramo Spokane WA UNITED STATES	Protocol unavailable.
FFFF	RFC1701, UNITED STATES	Protocol unavailable.

Apéndice D. El Cuarzo

El número de latidos de un cristal de cuarzo depende del tipo de circuito: en un reloj de pulsera suele vibrar a 32.768 Hz, es decir, vibra 32.768 veces en un segundo. Dado que este valor es múltiplo de 2, puede ser introducido en un divisor electrónico, y tras consecutivas subdivisiones convertirlo en un único latido por segundo, que una vez aplicado a un servomotor hará avanzar con gran precisión la manecilla del segundero, o mostrar los dígitos (en el caso de un reloj digital).

Otro ejemplo es un sintonizador de radio de FM, su cristal de cuarzo puede oscilar a varios millones de ciclos por segundo; esa señal introducida en un circuito resonante variable, permitirá situarnos sobre la frecuencia de la emisora que nos interesa recibir. Los cristales de cuarzo utilizados en los circuitos pueden llegar a ser del tamaño de una cabeza de cerillo. Pero, este factor tiene una limitación, ya que la cantidad de superficie cristalina es un parámetro directamente relacionado con su frecuencia fundamental de vibración. Además, por cuestiones técnicas, en ocasiones no conviene que los cristales vibren a frecuencias demasiado elevadas.

Sin embargo, estas limitaciones físicas pueden ser rescatadas eléctricamente, gracias a que un cristal de cuarzo no vibra solo en un sentido, pues lo hace longitudinalmente, transversalmente, y en modo flexor (curvándose o arqueándose), esto significa que además de resonar en una frecuencia fundamental determinada, también lo hace en múltiples frecuencias armónicas, llamadas también sobretonos. Por ejemplo, si quisiéramos seleccionar un cristal para conducir un sintonizador de FM que trabaja en una frecuencia fija de 100 Mhz, podríamos utilizar uno que vibrara a 100 Mhz (frecuencia fundamental), a 50 Mhz (primer sobretono), a 25 Mhz (segundo sobretono), o a 12.5 Mhz (tercer sobretono), etc. Hay que considerar, que cuanto más alejado esté el sobretono de la frecuencia fundamental, más débil será la señal entregada. En electrónica se denomina pérdida de factor Q.

La construcción de un cristal de cuarzo resulta curiosa: primero se corta una lámina (normalmente de forma circular) de sección aproximada a la frecuencia de resonancia fundamental que se desea obtener. Como es técnicamente imposible conseguir la frecuencia exacta durante el corte, es necesario esmerilarla con máquinas muy precisas. De los cortes que se pueden hacer, el corte AT es el más popular y es fabricado hasta frecuencias relativamente altas, mostrando una excelente estabilidad de frecuencia frente a las variaciones de la temperatura en la figura X. Posteriormente, se pulveriza y hornea en ambas caras unas finas películas metálicas de solución de plata, o también mediante evaporación de oro, plata o aluminio, con objeto de disponer de dos superficies de contacto (los electrodos). Mediante una soldadura de bajo punto de fusión, se conectan a las superficies metálicas dos alambres conductores.

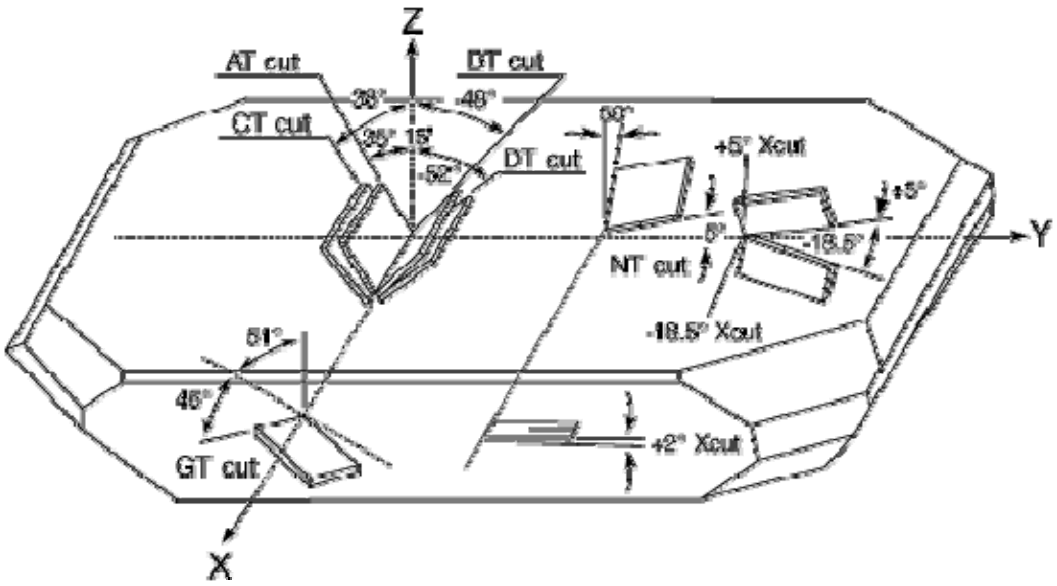


FIGURA D-1. EL CRISTAL DE CUARZO Y SUS ÁNGULOS DE CORTE

El punto de contacto para soldar los alambres debe ser elegido cuidadosamente, y tiene que ser exclusivamente en un punto llamado nodal, es decir, donde el cristal no produzca vibración, pues en caso contrario el alambre amortiguará la resonancia y dejará de vibrar, de la misma forma que una campanilla se ahogaría si la tocamos con la mano cuando está sonando. Finalmente, el cristal es encapsulado en una caja herméticamente cerrada, de vidrio, metálica u otro material adecuado. Antes de su comercialización, al cristal se le hace vibrar durante unas horas para que envejezca y se estabilice.

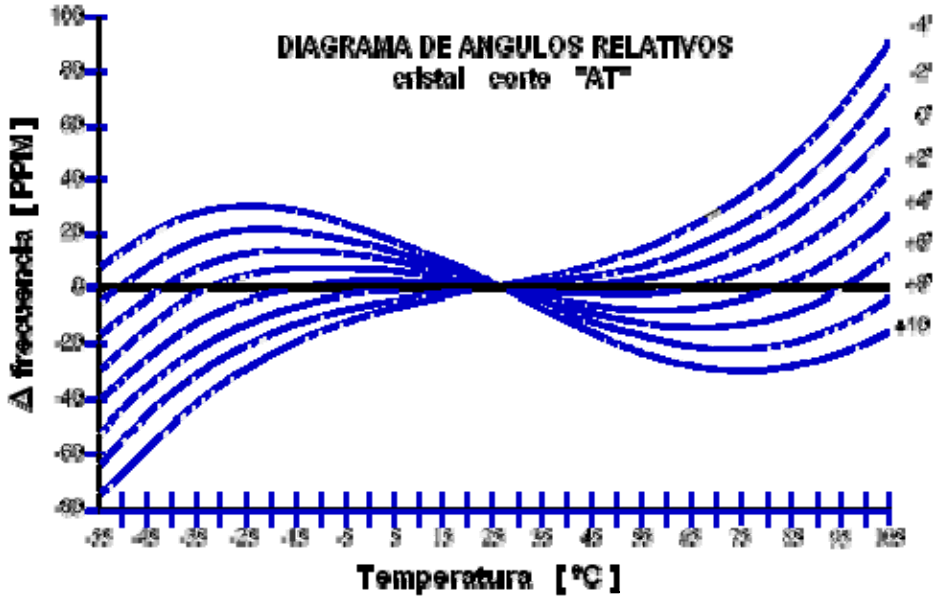


FIGURA D-2. RELACIÓN TEMPERATURA-FRECUENCIA EN LOS CRISTALES DE CUARZO CON LOS CORTES AT

El circuito eléctrico equivalente que se muestra en la figura 1-5 es un esquema del cristal de cuarzo trabajando a una determinada frecuencia de resonancia. El capacitor Co o capacidad en paralelo, representa en total la capacidad entre los electrodos del cristal más la capacidad de la carcaza y sus terminales. R1, C1 y L1 conforman la rama principal del cristal y se conocen como componentes o parámetros donde:

L1 representa la masa vibrante del cristal.
C1 representa la elasticidad del cuarzo.
R1 representa las pérdidas que ocurren dentro del cristal.

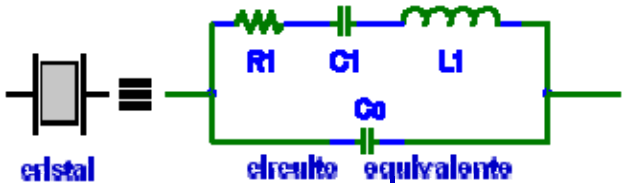


FIGURA D-3. EL CIRCUITO EQUIVALENTE QUE REPRESENTA UN CRISTAL DE CUARZO

Un cristal tiene dos frecuencias de fase cero, como se muestra en la figura 1-6. La más baja es la frecuencia de Resonancia Serie indicada como FS. En este punto el cristal se comporta como una resistencia en el circuito, la impedancia está en un mínimo y la corriente que circula es la máxima, a medida que se incrementa la frecuencia, el cristal pasa por la Frecuencia de Resonancia en Paralelo y llega a la frecuencia de anti-resonancia FA en la cual la impedancia es la máxima, y las reactancias de L1 y la de Co se cancelan. En este punto, la corriente que circula por el cristal es mínima.

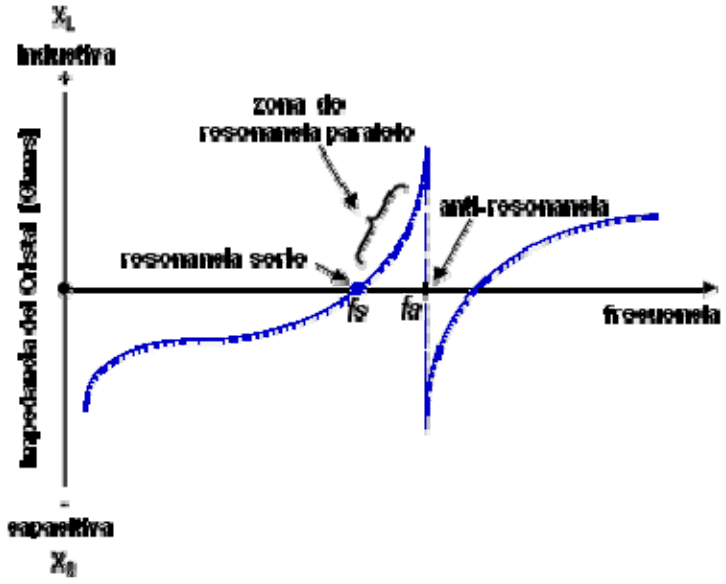


FIGURA D-4. COMPORTAMIENTO INDUCTIVO-CAPACITIVO DEL CRISTAL DE CUARZO

Apéndice E. Aplicaciones que utilizan la sincronización del reloj

A continuación se explican dos ejemplos que utilizan el cómputo distribuido con la sincronización del reloj y un ejemplo de cómputo local y distribuido para observar sus ventajas y desventajas.

E.1 Entrega de mensajes a lo más uno

Se refiere a la forma de reforzar la entrega de “a los más un mensaje” hacia un servidor, incluso en presencia de fallas. El método tradicional es:

Cada mensaje debe tener un número de mensaje, y que cada servidor guarde los números de los mensajes que ha visto, de modo que pueda establecer diferencia entre los mensajes nuevos y las retransmisiones. El problema con el algoritmo es que si un servidor falla y vuelve a arrancar, pierde su tabla con los números de mensajes. Otro requerimiento es saber cuánto tiempo deben conservarse los números de los mensajes [1].

Con el uso del tiempo, el algoritmo se puede modificar de la siguiente manera. En este caso, cada mensaje tiene un identificador de conexión (elegido por el emisor) y una marca de tiempo (timestamp). Para cada conexión, el servidor registra en una tabla la marca de tiempo más reciente que haya visto. Si el número de cualquier mensaje recibido en cierta conexión es menor que la marca de tiempo guardada en esa conexión, el mensaje se rechaza como un duplicado [2].

Para eliminar las marcas de tiempo anteriores, cada servidor mantiene de manera continua una variable global:

G = Tiempo Actual - Tiempo Máximo de Vida - Desviación Máxima del Reloj

En donde "Tiempo Máximo de Vida" es el tiempo máximo de vida de un mensaje y "Desviación Máxima del Reloj" es la peor distancia de alejamiento entre UTC y el reloj.

Cualquier marca de tiempo anterior a *G* se puede eliminar con seguridad de la tabla, pues todos los mensajes con esa edad ya han muerto.

Si un mensaje recibido tiene un identificador de conexión desconocido, se acepta si su marca de tiempo es más reciente que *G* y rechazado si su marca de tiempo es anterior a *G*, puesto que todo lo anterior seguramente es un duplicado.

- *G* es el resumen los números de mensaje de todos los mensajes anteriores.
- Cada ΔT, es el tiempo actual se escribe en disco.
- Cuando falla un servidor y vuelve a arrancar, vuelve a cargar *G* del tiempo guardado en el disco y lo incrementa según el periodo de actualización, ΔT.

- Cualquier mensaje recibido con una marca anterior a *G* se rechaza como duplicado.
- Como consecuencia, se rechaza cada mensaje que podría ser aceptado antes de la falla.
- Algunos de los nuevos mensajes podrán rechazarse de manera incorrecta, pero bajo todas las condiciones, el algoritmo conserva la semántica a lo más uno.

E.2 Consistencia del caché con base en el reloj

Se refiere a la consistencia del caché en un sistema distribuido de archivos. Por razones de desempeño, es deseable que los clientes puedan ocultar archivos de manera local. Sin embargo, el ocultarlos introduce potencialmente inconsistencia, esto es: si dos clientes modifican el mismo archivo al mismo tiempo. La solución usual consiste en:

Distinguir las caracteristicas del archivo (lectura o escritura) para ocultarlas. La desventaja de este esquema es que si un cliente tiene un archivo oculto para su lectura, antes de que otro cliente pueda obtener una copia para su escritura, el servidor tiene que solicitar al cliente de lectura que invalide su copia, incluso si la copia se realizó unas cuantas horas antes. Este costo adicional se puede eliminar mediante los relojes sincronizados [19].

La idea básica es que, cuando un cliente desea un archivo, se la da en renta.

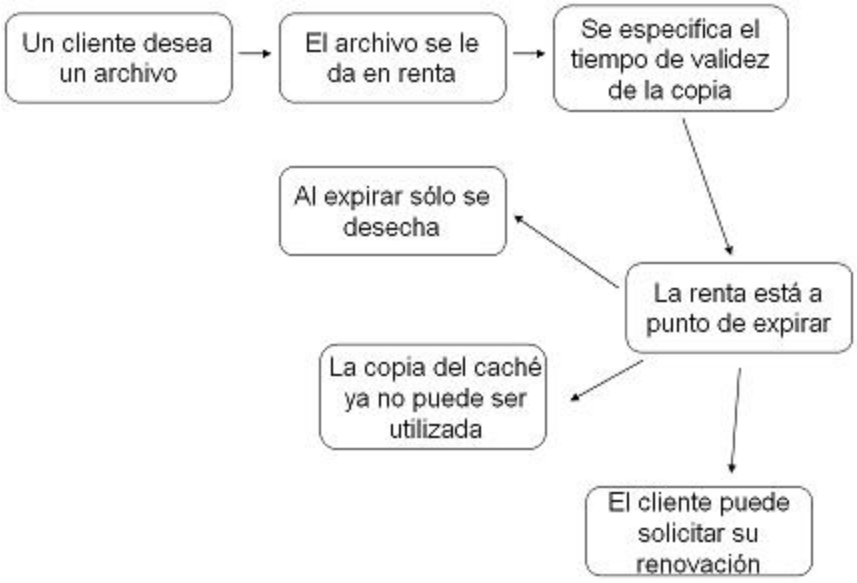


FIGURA E-1. REPRESENTACIÓN DE LA CONSISTENCIA DEL CACHE USANDO EL CONCEPTO DE RENTA DEL ARCHIVO

Si una renta expira y el archivo (aún en el caché) se necesita un poco más de tiempo, el cliente puede preguntar al servidor si la copia que tiene (identificada por una marca de tiempo) sigue siendo la activa. En tal caso, se genera una nueva renta, pero sin retransmitir el archivo.

Si uno o más clientes tienen un archivo en caché para su lectura y entonces otro cliente desea escribir en el archivo, el servidor debe solicitar a los lectores que terminen sus rentas de manera prematura. Si uno o más de ellos ha sufrido una falla, el servidor sólo espera hasta que expire la renta del cliente muerto.

En el algoritmo tradicional, donde el permiso para guardar el caché debe regresarse de manera explícita del cliente al servidor, ocurre un problema si el servidor pide al cliente o clientes que regresen al archivo (es decir, que lo eliminen de su caché) y no hay respuesta. El servidor no puede determinar si el cliente está muerto o sólo está lento. Con el algoritmo basado en un cronómetro (basado en la sincronización del reloj), el servidor puede esperar y dejar que expire la renta [19].

E.3 Programa Make

Otra aplicación del tiempo en la computación es el funcionamiento del programa make en unix. Cuando un programador termina de modificar todos los archivos fuentes, inicia make, el cual examina la hora en que todos los archivos fuentes y objetos fueron modificados por última vez. Si el archivo fuente salida.c tiene la hora 21:51 y el correspondiente archivo objeto salida.o tiene la hora 21:50, make sabe que salida.c tiene modificaciones desde la creación de salida.o por lo que entonces hay que volver a compilar salida.c. Por otro lado si salida.c tiene la hora de 21:44 y salida.o tiene la hora 21:45, entonces no es necesario volver a compilar. Así el programa make revisa todos los archivos fuentes para determinar aquellos que deban volver a compilarse y llamar al compilador para que realice esta tarea [19].

¿Que pasa si este mismo ejemplo sucede en un sistema distribuido donde no existe un acuerdo global en el tiempo?

Supongamos que salida.o tiene la hora 21:44 y que poco después se modifica salida.c, pero se le asigna la hora 21:43 debido a que el reloj de esta máquina es un poco lento, como se muestra en la siguiente figura. Make no llamará al compilador. El programa en binario ejecutable resultante contendrá una mezcla de archivos objetos de las fuentes anteriores y nuevas. El programa no funciona y el programador no encuentra el error por la parte incorrecta del código. Tiene varias opciones el programador para resolver el problema, si es que el programador sabe el origen del mismo, pero le llevará tiempo encontrarlo.

Por esto, es la solución más rápida pero no la mejor consiste en “borrar” todos los archivos “.o” para que así el programa make pueda compilar todos los archivos “.o” con toda su funcionalidad. Esta solución funciona para este tipo de problema pero el origen del mismo es la uniformidad de tiempo en las máquinas involucradas. Cada computadora tiene su propio reloj y se asigna un tiempo anterior a un evento ocurrido después de otro esto se muestra en la figura E-2

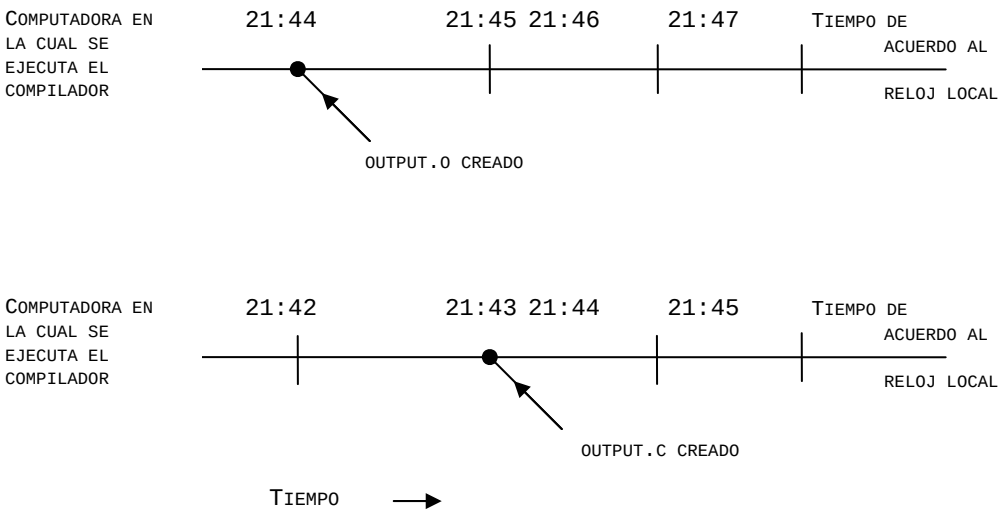


FIGURA E-2. ASIGNACIÓN DE TIEMPOS A EVENTOS EN LA COMPUTADORA

Apéndice F. Generación de una Gráfica de Sincronización

Considere un sistema con 2 procesadores *u* y *v*. Suponga que *u* tiene un reloj sin desvíos [1,1] y *v* tiene un reloj con desvíos [0.5, 1.5].

El procesador *u* envía el mensaje m_1 a *v* en el tiempo local -1, tal que m_1 es garantizado para ser entregado en no menos de 2 unidades de tiempo y no más de 3 unidades de tiempo.

m_1 es recibido en *v* en el tiempo local 1.

El procesador *v* envía el mensaje m_2 a *u* en el tiempo local 3, tal que m_2 es garantizado a ser entregado en no menos de 5 unidades y no hay limite de transmisión en el límite superior. m_2 es recibido en *u* en el tiempo local 8. El escenario de está ejecución y la gráfica tipo V se encuentra en la figura F-1.

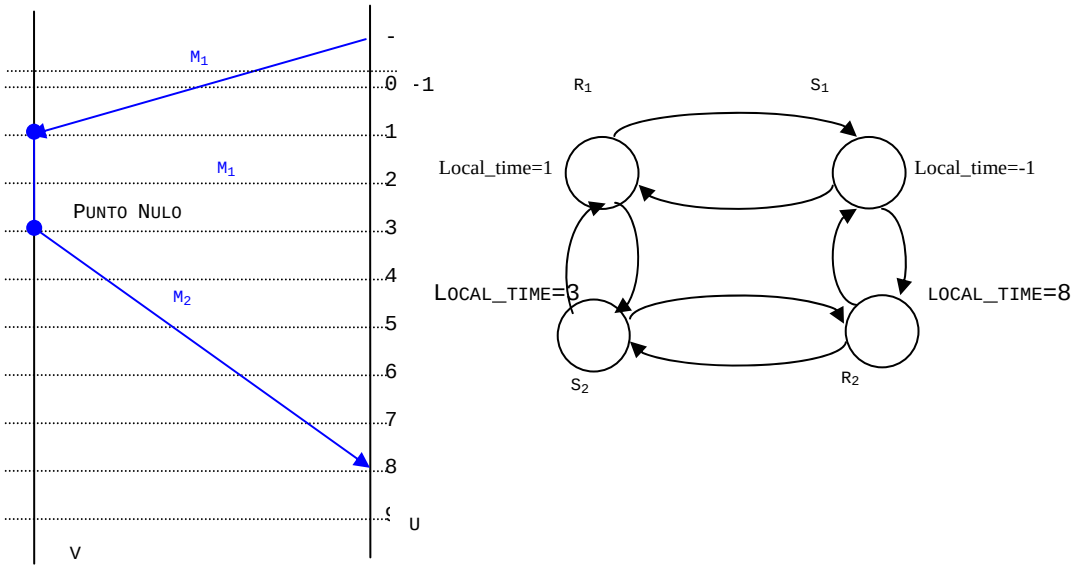


FIGURA F-1. ESCENARIO Y GRÁFICA TIPO V PARA LA EJECUCIÓN DE LOS PROCESADORES U,V

A continuación se calcula el retardo virtual, así como el mapeo de límites.

$$virt\_del(p,q) = local\_time(p) - local\_time(q)$$

$virt\_del(s_1,r_1) = -2$	$virt\_del(r_1,s_1) = 2$
$virt\_del(s_2,r_2) = -5$	$virt\_del(r_2,s_1) = 9$
$virt\_del(s_1,r_2) = -9$	$virt\_del(r_1,s_2) = -2$
$virt\_del(s_2,r_1) = 2$	$virt\_del(r_2,s_2) = 5$

TABLA F-1. FORMULA Y RESULTADO DE LOS RETRASOS VIRTUALES EN LOS PUNTOS R_1, S_1, R_2 Y S_2

A partir de los retrasos virtuales se generará los mapeos de límites para normalizar los eventos, se calculan todos los valores de B y se obtiene la siguiente tabla.

	B(P,Q)			
	-L(M)	H(M)	VIRT_DEL(P,Q)/ $\underline{\rho}$	VIRT_DEL(P,Q)/ $\overline{\rho}$
$B(S_1,R_1)$	-2	---	---	---
$B(R_1,S_1)$	---	3	---	---
$B(S_2,R_2)$	-5	---	---	---
$B(R_2,S_2)$	---	∞	---	---
$B(S_1,R_2)$	---	---	-9	---
$B(R_2,S_1)$	---	---	---	9
$B(S_2,R_1)$	---	---	4	---
$B(R_1,S_2)$	---	---	---	-4/3

TABLA F-2. RESULTADO DE LOS MAPEOS DE LÍMITE

Una vez obtenido los retrasos virtuales y los mapeos de límites, se obtienen las gráficas de tipo *P* y las compensaciones (offset).

Primero se va a generar las compensaciones absolutas de los puntos de la ejecución dada, a partir de la gráfica tipo P se obtiene:

$\delta(s_1)= 1$ $\quad\quad$ $\delta(r_1)= 1$
$\delta(s_2)= 0.5$ $\quad\quad$ $\delta(r_2)= 1$

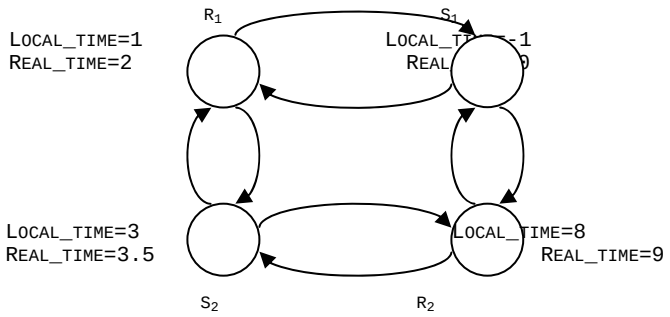


FIGURA F-2. GRÁFICA DE TIPO P

Posteriormente se realizará el cálculo de las compensaciones relativas para cada pareja de puntos

$\delta(s_1,r_1)= 0.5$ $\quad\quad$ $\delta(s_1,r_2)= 0$
$\delta(s_1,s_2)= 0.5$ $\quad\quad$ $\delta(r_1,s_2)= 0.5$
$\delta(r_1,r_2)= 0$ $\quad\quad$ $\delta(s_2,r_2)= -0.5$

En este momento se cuentan con todos los elementos necesarios para generar la gráfica de sincronización.

A continuación se tiene la tabla de los pesos de los arcos y en la figura 2-8 se tendrá la gráfica de sincronización.

	B(P,Q)	VIRT_DEL(P,Q)	W(P,Q)
(S_1,R_1)	-2	-2	0
(R_1,S_1)	3	2	1
(S_2,R_2)	-5	-5	0
(R_2,S_2)	∞	5	∞
(S_1,R_2)	-9	-9	0
(R_2,S_1)	9	9	0
(S_2,R_1)	4	2	2
(R_1,S_2)	-4/3	-2	2/3

TABLA F-3. OBTENCIÓN DE LOS PESOS DE LA EJECUCIÓN

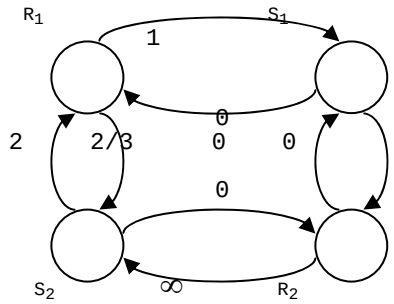


FIGURA F-3. GRÁFICA DE SINCRONIZACIÓN

La prueba del lema y del teorema se omite, pero se pueden encontrarse en [5].

Por ultimo se encontrarán las distancias mínimas de acuerdo a los pesos mínimos. Se asegurará que no hay ciclos de pesos negativo (todo esto es debido a varios teoremas y lemas definidos en [5]) y con ello se obtiene $\delta(p,q) \varepsilon [-d(p,q), d(p,q)]$.

$D(S_1,R_1)= 0$ $D(R_1,S_1)= 2/3$
$D(S_1,S_2)= 2/3$ $D(S_2,S_1)= 0$
$D(S_1,R_2)= 0$ $D(R_2,S_1)= 0$
$D(S_2,R_1)= 0$ $D(R_1,S_2)= 2/3$
$D(S_2,R_2)= 0$ $D(R_2,S_2)= 2/3$
$D(R_2,R_1)= 0$ $D(R_1,R_2)= 2/3$

Con esto, se puede obtener la distancia más corta entre un evento y otro y así conocer los arcos que hay que considerar.

Apéndice G. Capa de Enlace de Datos

La capa de enlace de datos puede diseñarse para ofrecer varios servicios. Los servicios reales ofrecidos pueden variar de sistema en sistema. Los tres servicios proporcionados son:

1. Servicio sin acuse sin conexión.
2. Servicio con acuse sin conexión.
3. Servicio con acuse y orientado a conexión.

El servicio sin acuse y sin conexión consiste en hacer que la computadora de origen envíe marcos (frames) independientes a la computadora destino sin pedir que estas los reconozca o no envíe acuse de recibo. Al no haber conexiones establecidas, si se pierde un marco debido al ruido, distorsión, diafonía u otro elemento físico en el medio de transmisión, no se intenta recuperarlo en esta capa, sino solicita retransmisión que permite mandar frames:

- A un único destino (punto a punto o transferencia unicast).
- A múltiples destinos de la misma red (multicast).
- A todas las estaciones de la red (broadcast).

El uso de multicast y broadcast puede reducir el tráfico de la red cuando la misma información tiene que ser enviada a todas las estaciones de la red.

En el servicio con acuse y sin conexión, consiste en que el marco es enviado y reconocido individualmente. De esta manera, el transmisor sabe si el marco ha llegado correctamente o no. Si no ha llegado en un tiempo especificado, puede enviarse nuevamente. Este servicio es útil en canales inestables, como en los sistemas inalámbricos. Además soporta conexión punto a punto (point to point).

El servicio con acuse y orientado a conexión es el más elaborado que puede proporcionar la capa de enlace de datos a la capa de red. Con este servicio, las computadoras de origen y destino establecen una conexión antes de transferir los datos. Cada marco es enviado a través de la conexión está y numerado, la capa de enlace de datos garantiza que cada marco enviado llegará a su destino, es más, se garantiza que cada marco será recibido exactamente una vez y que todos los marcos sean recibidos en el orden adecuado. En contraste, con el servicio sin conexión es concebible que un acuse de recibo perdido cause el envío de un marco varias veces, y por tanto recibido varias veces. El servicio orientado a la conexión, por su parte, proporciona a los procesos de la capa de red el equivalente de un flujo de bits confiable.

Al usarse un servicio orientado a conexión, las transferencias tienen tres fases distintas. En la primera fase, la conexión se establece haciendo que ambos lados inicialicen las variables y contadores necesarios para seguir la pista de los marcos que han sido recibidos y los que no. En la segunda fase se transmiten uno o más marcos. En la última fase (tercera fase), la conexión se cierra, liberando todas las

variables, los buffers y otros recursos utilizados para mantener la conexión.

G.1 Control de Acceso al Medio

La subcapa de control de acceso al medio (MAC) de la capa de enlace de datos administra el protocolo de acceso al medio físico de red. Esta capa (o subcapa) proporciona transferencia no orientada a conexión de frames. Dado que las transmisiones en redes LAN están relativamente libres de errores, los protocolos MAC no suelen incluir procedimiento de control de errores. La entidad MAC acepta un bloque de datos de la subcapa LLC o directamente de la capa de red y construye un PDU que incluye las direcciones MAC de origen y destino y una secuencia de comprobación de trama, FCS (Frame Check Sequence), que no es más que una comprobación *CRC*. Las direcciones MAC especifican las conexiones lógicas de las estaciones de trabajo a la red LAN. La principal tarea de las entidades MAC es ejecutar las conexiones lógicas de las estaciones de trabajo a la red LAN.

En las especificaciones de la IEEE se definen las direcciones MAC, que permiten que varios dispositivos se identifiquen sin repetición, entre unos y otros en la capa de enlace de datos.Existen varias técnicas de control de acceso al medio de las cuales se mencionan algunas:

- CSMA/CD (mejor conocido Ethernet),
- Token Ring,
- FDDI,
- 100VG AnyLAN,
- ATM,

La técnica más difundida en las redes de tipo LAN es la de Acceso Múltiple Sensible a la Portadora con Detección de Colisiones (CSMA/CD, Carrier Sense Multiple Access with Collision Detection).

La versión original de está técnica en banda base fue desarrollada por Xerox para redes LAN Ethernet. La idea original fue de Robert Metcalfe, donde el concepto se muestra en la siguiente figura G-1:

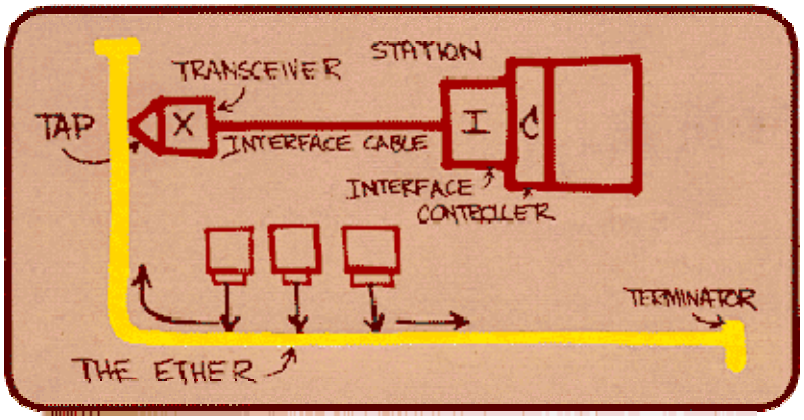


FIGURA G-1. DIAGRAMA QUE REPRESENTA AL PROTOCOLO ETHERNET [25]

A continuación se explica las tramas ó marcos que opera el CSMA/CD, por lo cual no se aborda los temas de la operación, historia, evolución y técnica del control de acceso al medio. Todo con el propósito de entender como crear paquetes CSMA/CD junto con paquetes LLC para enviar a sus respectivos destinos.

Las especificaciones de CSMA/CD determinan la estructura de la trama cuando una estación la envía. Dicha trama varía dependiendo de las especificaciones de IEEE o de DIX. En cuanto a la longitud y número de campos son idénticos en ambos estándares, la única diferencia en las tramas está en el contenido de los campos y la interpretación subsecuente de estos contenidos por las estaciones que envían o reciben las tramas. A continuación en la figura G-2 se muestran las tramas de los estándares de DIX e IEEE 802.3

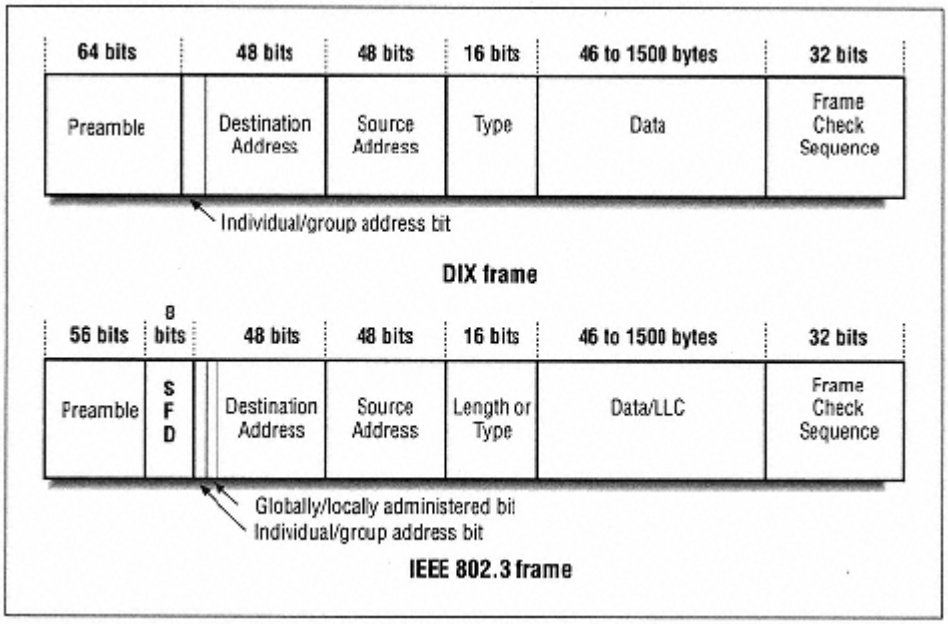


FIGURA G-2. TRAMAS DE LOS ESTÁNDARES DIX E IEEE 802.3 [23]

Los campos de las tramas son los siguientes:

Preámbulo. El campo consta de 7 bytes que son usados para permitir al circuito PLS alcanzar el estado constante de la sincronización con el tiempo de la trama recibida. El patrón del preámbulo es:

10101010 10101010 10101010 10101010 10101010 10101010 10101010

Campo delimitador de inicio de trama (SFD). El campo SFD es la secuencia 10101011. Sigue inmediatamente al patrón de preámbulo e indica el inicio de la trama.

Campos de Dirección. Cada trama MAC contendrá dos campos de dirección: la dirección origen y la dirección de destino, en ese orden. El campo de la dirección de origen específica la(s) dirección(es) destino a la cual la trama es enviada. La dirección origen identifica la estación para el cual

la trama fue iniciada. La representación de cada dirección se puede ver en la siguiente figura G-3

FIGURA G-3. FORMATO DEL CAMPO DE DIRECCIÓN [23]

Cada campo de dirección contiene 48 bits de longitud. Mientras el IEEE 802 especifica de 16 o 48 bits de direcciones, no está puesta en práctica y el IEEE 802.3 utiliza direcciones de 16 bits. El uso de 16 bits de dirección está específicamente excluido por este estándar.

El primer bit (LSB) se usa en el campo de dirección destino como un bit asignado, para identificar a cualquier dirección destino como una dirección de grupo o una individual. Si el valor del bit es cero (0), indica que el campo de dirección contiene una dirección individual. Si el valor del bit es 1, indica una dirección grupo de 1 o más estaciones, incluso a todas las estaciones conectadas a la LAN. En la dirección origen el primer bit esta reservado y puesto en cero [23].

El segundo bit es usado para distinguir las direcciones local o global. Para las direcciones administradas globalmente (o U de Universal), el valor del bit es 0. Si una dirección es local, entonces su valor es 1. Note que para las direcciones de tipo broadcast se encuentran también en 1.

Cada bit de cada campo de dirección esta transmitiendo el primer bit menos significativo.

La subcapa de dirección MAC tiene dos tipos:

a) Dirección Individual. La dirección está asociada con una estación particular en la red.

b) Dirección de Grupo. La dirección multidestino, asocia con uno o más estaciones que están en la red. Estos son los dos géneros de la dirección multicast:

   i) Dirección de Grupo de Multicast. Una dirección asociada por convención de alto nivel con un grupo de estaciones lógicamente relacionadas.

   ii) Direcciones Broadcast. Las direcciones multicast predefinida que siempre denotan en poner todas las estaciones de una LAN.

Si se asigna 1’s en el campo de la dirección destino esta predefiniendo para ser una dirección de tipo broadcast. Este grupo esta predefinido para cada medio de comunicación que consiste de todas las estaciones

activas conectadas al medio. Todas las estaciones podrán reconocer la dirección broadcast. No es necesario que una estación sea capaz de generar la dirección broadcast.

En el campo de la dirección destino se especifica la(s) estación(es) para las cuales la trama es enviada. Puede ser una dirección individual, multicast o broadcast. En el campo de la dirección origen se específica la estación que envió la trama. Este campo no es interpretado por la subcapa MAC.

Campo de Longitud/Tipo. Este campo consta de 2 bytes tomando dos significados, dependiendo del valor del número. Para la evaluación numérica. El primer byte es el más significativo de este campo.

Si el valor de este campo es menor o igual al valor de la máxima trama válida entonces el campo de tipo/longitud indica el número bytes de los clientes MAC contenidos en el campo subsecuente a la trama.

Si el valor es mayor o igual a 1536 en decimal, (0600 hexadecimal) entonces el campo de longitud/tipo indica el protocolo del cliente MAC. La interpretación de los campos de longitud y tipo son mutuamente excluyentes.

Sin importar la interpretación del campo longitud/tipo, si la longitud del campo de datos es menor al mínimo requerido para las operaciones propias del protocolo, entonces un campo PAD (una secuencia de bytes) sera agregado al final del campo de datos, antes del campo FCS. El procedimiento para determinar el tamaño del campo PAD se especifica más adelante.

Campo de Datos y PAD. El campo de datos contiene una secuencia de *n* bytes. Los datos completos transparentes están provistos en el sentido que cualquier secuencia arbitraria de los valores de bytes puede aparecer en el campo de datos hasta un número máximo específico por la implantación del estándar que está usando. Un tamaño de trama mínimo es requerido para que la operación del protocolo CMSA/CD sea correcto y está especificado por la implantación particular del estándar. Si es necesario, el campo de datos se extiende para agregar bits extras (esto es un pad) en unidades de bytes después del campo de datos. El tamaño del PAD, es determinado por el tamaño del campo de datos proporcionado por el cliente MAC y el tamaño mínimo de la trama y el tamaño de los parámetros de dirección de la implantación en particular. El máximo tamaño del campo de los datos esta determinado por el máximo tamaño de la trama y el tamaño de los parámetros de la implantación en particular.

La longitud del campo PAD requerido para que el cliente de datos MAC que esta de *n* bytes es el max[0,minFrameSize-(8 x n +2 x addressSize + 48)] bits. El tamaño máximo posible del campo de datos es max UntaggedFrameSize – (2 x addressSize+48)/ 8 bytes.

Campo de Secuencia de Verificación de la Trama (FCS, Frame Check Sequence). Un chequeo de redundancia cíclica (CRC) es usado por los algoritmos transmisor y receptor para generar un valor CRC para el campo FCS. El campo FCS contiene 4 bytes (32 bits) del valor CRC (Chequeo de Redundancia Cíclica). El valor es calculado como una función que contiene la dirección origen, dirección destino, longitud, datos LLC y PAD (que

esto es, todos los campos excepto el preámbulo, SFD, FCS, y extensión). La codificación esta definida por la siguiente función polinomial:

$$G(x)=x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$$

Matemáticamente, el valor CRC corresponde a la obtención de la trama que esta definido por el siguiente procedimiento:

a) Los primeros 32 bits de la trama se complementan.
b) Los n bits de la trama están consideradas para ser los coeficientes de un polinomio M(x) del grado n-1. (El primer bit del campo de la dirección destino corresponde al $x^{(n-1)}$ término y al último bit del campo de datos correspondientes al termino x^0).
c) M(x) es multiplicado por x^{32} y dividido por G(x), produciendo un residuo R(x) de grado $\leq$ 31.
d) Los coeficientes de R(x) están considerado para ser una secuencia de 32 bits.
e) La secuencia de bit está complementado y el resultado es el CRC.

Los 32 bits del valor de CRC están colocados en el campo FCS así que el x^{31} término es el bit más a la izquierda del primer byte, y el término x^0 es el bit más a la derecha del último byte. Los bits del CRC son así transmitidos en ese orden: x^{31}, x^{30},…., x^1, x^0.

Campo de extensión. El campo de extensión es el campo posterior a FCS y se compone de una secuencia de bits de extensión, que son fácilmente distinguidos de los bits de datos. La longitud del campo está en el rango de cero (slotTime – minFrameSize) bits inclusive. El contenido de los campos de extensión no está incluido en el cálculo de FCS.

El campo de extensión debe tener una longitud mayor que cero bajo ciertas condiciones. La longitud del campo de extensión deberá ser cero bajo todas las demás condiciones.

Cada bit de la trama MAC, con excepción del FCS, es transmitido primero por el orden bajo o menos significativo.

Una trama MAC inválida estará definida con al menos una de las siguientes condiciones:

a) La longitud de la trama es inconsistente con un valor de longitud específico en el campo longitud/tipo. Si el campo de longitud/tipo contiene un valor definido en las condiciones de longitud/tipo, entonces la longitud de la trama se asume para que sea consistente con este campo y no deberá considerarse una trama invalida en esta base.

b) No sea un número entero en la longitud de bits.

c) Los bits de la trama entrante (excluyendo los campos mismo de FCS) no genera un valor CRC idéntico a uno recibido.

Los contenidos de las tramas MAC inválidas no pasarán a las subcapas de control LLC o MAC. La ocurrencia de una trama MAC inválida deberá comunicarse a la administración de la red.

G.2 Control de Enlace Lógico

La subcapa de control de enlace lógico (LLC) de la capa de enlace de datos administra las comunicaciones entre dispositivos sobre un solo enlace de la red. LLC está definido en las especificaciones de IEEE 802.2. LLC soporta servicio orientado a conexión y servicio orientado a no conexión, ambos, usados por los protocolos de las capas más altas.

El formato PDU (Protocol Data Unit) de LLC es como se muestra en la figura G-4:

Dirección DSAP	Dirección SSAP	Control	Información
8 bits	8 bits	8 o 16 bits	M*8 bits

FIGURA G-4. FORMATO PDU DE LLC

Campos de Direcciones

Cada PDU contiene dos campos de direcciones: DSAP (Destination Services Access Point) y SSAP (Source Services Access Point) en ese orden. El campo de la dirección DSAP identifica uno o más servicios del punto de acceso, el cual la información del campo LLC está previsto. El campo de la dirección SSAP identifica el servicio especifico del punto de acceso para el cual la información del campo LLC fue iniciado [23].

Dichas representaciones de cada campo de dirección se muestra a continuación en las figuras G-5 y G-6:

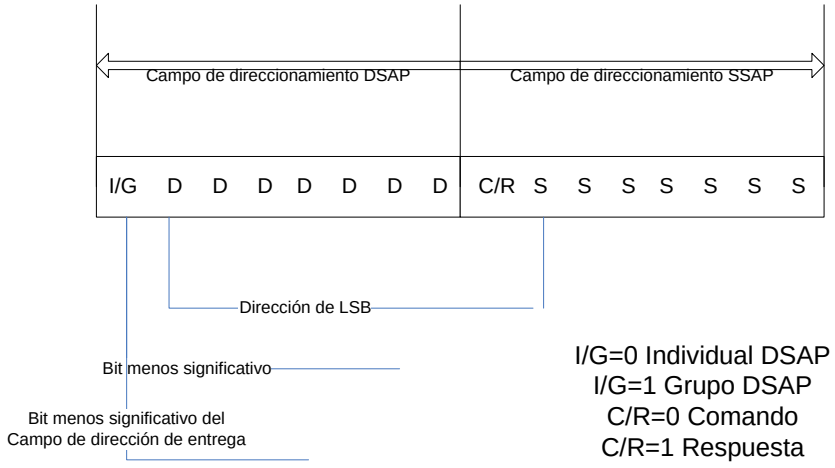


FIGURA G-5. FORMATO DE LOS CAMPOS DE LAS DIRECCIONES DSAP Y SSAP

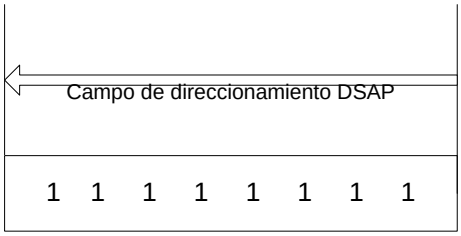


FIGURA G-6. FORMATO GLOBAL DEL CAMPO DE LA DIRECCIÓN DSAP

Cada campo de dirección contiene 1 byte.

Cada campo de dirección contiene 7 bits de la dirección actual y un bit que es usa en el campo de la dirección DSAP como cualquier dirección individual o de grupo de direcciones (llamado el bit para el tipo de dirección) y en el campo de dirección SSAP identifica que el PDU de LLC como valor de tipo comandoe o de respuesta (llamado el bit de identificación comando/respuesta).

El bit asignado para el tipo de elección es localizado en la posición del bit menos significativo del campo de dirección DSAP. Si el valor del bit es “0”, indica que la dirección es una dirección individual DSAP. Si el valr del bit es “1”, indica que la dirección de grupo DSAP y que no identifica a ninguno, uno o más o incluso a todos los servicios de puntos de acceso que están en servicio por la entidad LLC.

El bit de identificación comando/respuesta se localiza en el bit menos significativo del campo de dirección SSAP. Si el valor del bit es “0”, indica que el PDU es un comando. Si el valor del bit es “1” entonces indica que el PDU es una respuesta [23].

Campo de Control

El campo de control consiste de uno o dos bytes que se usan para designar funciones de comandos y respuestas, y que contiene un número de secuencia cuando se requiere.

Los tres formatos definidos para el campo de control se encuentran en la siguiente figura G-7, se usa el rendimiento de las funciones de transferencia de información enumerada, transferencia de supervisión enumerada, control sin enumerar y transferencia de información sin enumerar. Las funciones de transferencia de supervisión y de información enumerada únicamente se aplican al tipo 2 de operación. Las funciones de transferencia de control sin enumerar e información sin enumerar se aplican a los tipos 1, 2 y 3 de operación dependiendo de la función seleccionada.

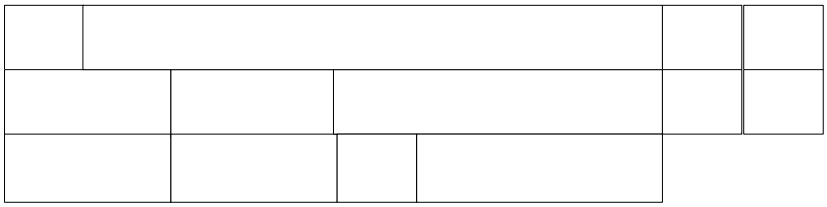


FIGURA G-7. FORMATO DEL PDU Y LOS CAMPOS DE CONTROL DE LLC

PDU de Comando/Respuesta d

transferencia de información

PDU de Comando/Respuesta d

Supervisión

PDU de Comando/Respuesta c

Enumeración

Formato de transferencia de información

El PDU del formato de información es usado para el rendimiento de la transferencia de la información, en cuanto a las operaciones del tipo 2. Excepto cuando otro caso se especifique, será únicamente el PDU LLC que deba contener un campo de información. Las funciones de N(S), N(R) y P/F son independientes, cada formato de información tendrán un número de secuencia N(S), un número de secuencia N(R) que confirma o no al formato de información, se está recibiendo formatos de información LLC y un bit P/F que está puesto a “1” o “0”.Este formato se encuentra en la figura G-8.

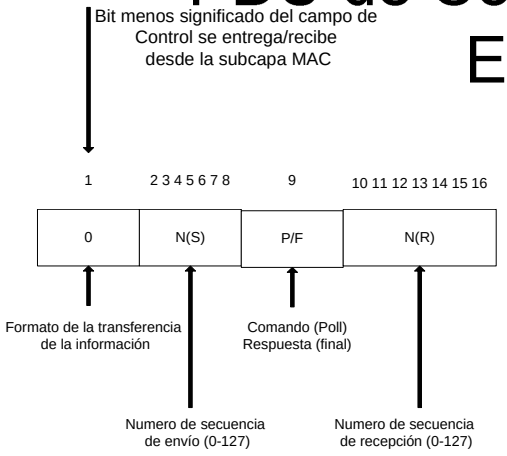


FIGURA G-8. CAMPO DE BITS DEL FORMATO TRANSFERENCIA DE INFORMACIÓN

Formato de Supervisión

El PDU del formato de supervisión se usa para rendir la supervisión de los enlaces de datos, en las funciones de control de las operaciones de tipo 2, como para el formato PDU de la información de confirmación, PDU’s del formato de información de petición de retransmisión, y PDU’s del formato de información de las peticiones de suspensión de transmisión temporales. Las funciones de N(R) u P/F son independientes, cada PDU del formato de supervisión es un número de secuencia N(R) que es o no una

confirmación del formato información adicional al que está recibiendo el LLC y un bit P/F que su vañlor es “1” o “0”. La figura G-9 muestra el PDU.

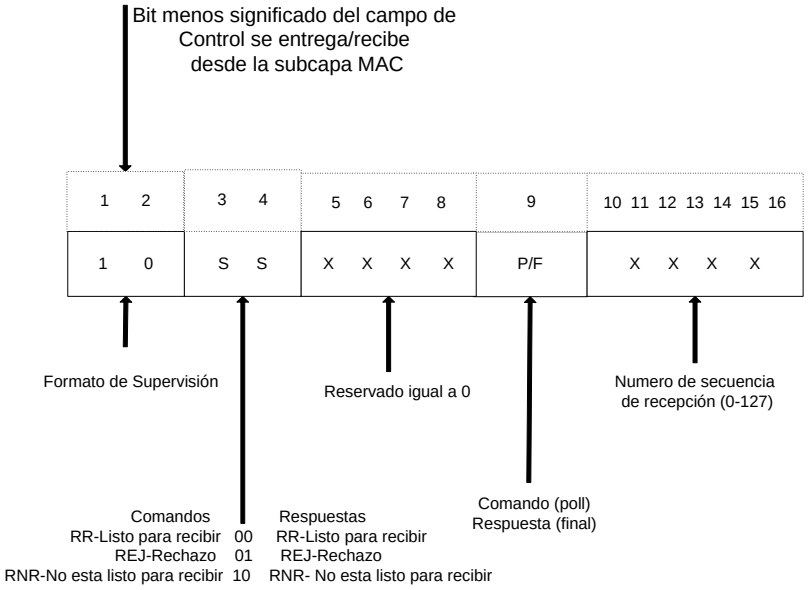


FIGURA G-9. PDU DEL FORMATO DE SUPERVISIÓN

Formato sin enumerar

El formato sin enumerar es usado en las operaciones del tipo 1, 2 y 3 que proveen funciones de control adicional al enlace de datos y proveen transferencia de información sin enumerar. El formato sin enumerar no contiene números de secuencia, pero incluye un bit P/F y su valor es “1” o “0”, su campo se muestra en la figura G-10.

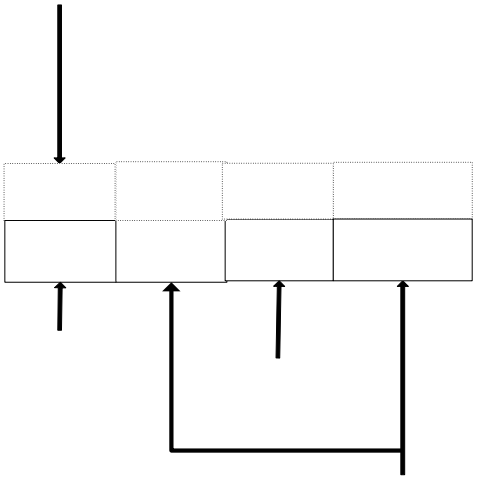


FIGURA G-10. PDU FORMATO SIN ENUMERAR

De los tres formatos, solo el formato sin enumerar se usa normalmente. El formato de un paquete PDU se identifica por los dos bits más bajos del primer byte del campo de control.

Campo de Información

El Campo de información consiste de cualquier número (incluyendo el cero) entero de bytes.

Un PDU inválido está definido como un PDU que cumpla al menos una de las siguientes condiciones:

- No es un número entero de bytes en la longitud.
- Si la longitud es menor de 3 bytes (1 byte del campo de control) o 4 bytes (2 bytes en el campo de control).

Estos PDU son ignorados.

Bibliografía

[1] Liskov Bárbara. Practical uses of synchronized clocks in distributed systems. Annual ACM Symposium on Principles of Distributed Computing, Proceedings of the tenth annual ACM symposium on Principles of distributed computing, Montreal Québec, Canadá, páginas 1-9, 1991.

[2] Liskov, B., Shira, L., and Wroclawski, J. Efficient At-Most-Once Messages Based on Synchronized Clocks. ACM Transactions on Computers Systems. Volume 9, páginas 125-142, 1991.

[3] Steiner, J.G., Neuman, C., Schiller, J.I. Kerberos: An Autenticación Service for Open Network Systems. Project Athena. MIT, Cambridge, MA, Marzo, 1988. ftp://athena-dist.mit.edu/pub/kerberos/doc/usenix.PS

[4] Liskov, B., Ghemawat, S., Gruber, R., Johnson, P., Shira, L., and Williams, M. Replication in the Harp File System. Proceedings of 13th ACM Symposium on Operating Systems Principles Pacific Grove, California, Estados Unidos, páginas: 226 – 238, 1991.

[5] Patt-Shamir, Boaz and Rajsbaum Sergio. A theory of clock synchronization. Annual ACM Symposium on Theory of Computing, Montreal, Proceedings of the twenty-sixth annual ACM symposium on Theory of computing. Québec, Canadá, páginas: 810-819,1994.

[6] Moser Heinrich and Schmid Ulrich. Optimal clock synchronization revisited: Upper and lower bounds in real-time systems, Proceedings of the International Conference on Principles of Distributed Systems 2006.

[7] Lundelius Jennifer and Lynch Nancy. A New Fault-Tolerant Algorithm for Clock Synchronization, Annual ACM Symposium on Principles of Distributed Computing, Proceedings of the third annual ACM symposium on Principles of distributed computing, Vancouver, British Columbia, Canadá, páginas: 75-88, 1984

[8] Mills David L. Computer Network Time Synchronization: The Network Time Protocol. Editorial Taylor & Francis Group, ISBN 0-8493-5805-1 ,2006.

[9] Lamport L., Time, clocks and the ordering of events in a distributed system. Communcations of the ACM, Volume 21(7), páginas: 558-565, 1978.

[10] L. Sha, T. Abdelzaher, et. Al Real time scheduling theory: a historical perspectiva. Real-Time Systems Journal, 28(2/3): páginas 101-155, 2004

[11] J. H. Anderson, Y-J. Kim, et al. Shared memory mutual exclusion: Major research trends since 1986. Distributed Computing, Volume 16, páginas 75-110,2003.

[12] J. H. Anderson, J-H. Yang, et al. Time/contention tradeoffs for multiprocessor synchronization. Information and Control, Volume 124 No. 1, páginas 68-84, 1996.

[13] R. Alur and D. L. Hill. A theory of timed automata. Theorical Computer Science, Volume 126, 22, páginas 183-235, 1994

[14] D. K. Kaynar, N. Lynch, et. Al. Timed i/o automata: A mathematical framework for modeling and analyzing real-time systems. Proceedings 24th IEEE internacional Real-Time Systems Symposium (RTSS'03), página 166, 2003

[15] D. Dolev, J. Halpen and R. Strong. On the possibility and impossibility of achieving clock synchronization, Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing, páginas 504-11, 1984.

[16] N. Lynch. Simulation techniques for proving properties of real-time systems. In Proceedings of the 2nd Annual ACM Symposium on Principles of Distributed Computing, páginas 44-54, 1983.

[17] Tanenbaum S Andrew. Sistemas Operativos Distribuidos. Editorial Prentice Hall, ISBN 9688806277 ,1996.

[18] Patt-Shamir Boaz. A theory of clock synchronization, PhD thesis. Massachussets Institute of Technology, 1994

[19] Albeseder Daniel.Evaluation of message delay correlation in distributed systems. In Proceedings of the Third Workshop on Intelligent Solutions for Embedded Systems, Hamburgo, Alemania, página 139-150, 2005

[20] Cormen H. Thomas, Leiserson E. Charles, Rivest L. Ronald and Stein Clifford. Introduction to Algorithms, The MIT Press, Segunda Edición ISBN: 0-262-03293-7, 2001.

[21] J.-F. Hermant and J. Widder. Implementig reliable distributed real-time systems with the ⊖-model. In proceedings of the 9th Internacional Conference on principles of Distributed Systems, Springer Verlag, Pisa, Italia, páginas 334-350, 2005.

[22] Tanenbaum S. Andrew. Computer Network., Prentice Hall, Tercera Edición, ISBN: 0-13-066102-3, 1997.

[23] LAN/MAN Estandards Committe of the IEEE Computer Society, http://standards.ieee.org/getieee802/download/802.2-1998.pdf, 1998.

[24] IEEE 802.3, sitio oficial del grupo de desarrollo de los estandares Ethernet basados en LAN. http://www.ieee802.org/3/

[25] Metcalfe M. Robert, diagrama para la presentación de Ethernet, http://www.intel.com/standards/case/case_ethernet.htm o http://www.ieee802.org/3/ , año 1976

[26] Sitio de códigos IP Core para proyectos en FPGA, http://www.opencores.org

[27] Comité IEEE 1364-2001, Estandar del lenguaje de Programación Verilog http://ieeexplore.ieee.org/xpl/standardstoc.jsp?isnumber=20656&isYear=2001

[28] Ethernet, http://www.opencores.org/projects.cgi/web/ethmac/overview, sitio web donde se especifica el MAC Ethernet para FPGA.

[29] Attiya Hagit, Jennifer Welch. Distributed computing : fundamentals, simulations and advanced topics. Wiley 2004 2da. Edición. ISBN: 0471453242.

[30] Artículo que describe como fue evolucionando el reloj desde la ENIAC, hasta antes del problema de Y2K y se realizaron las soluciones a problemas particulares.

http://www.computer.org/portal/site/computer/menuitem.eb7d70008ce52e4b0ef1bd108bcd45f3/index.jsp?&pName=computer_level1&path=computer/homepage/July07&file=ourtime.xml&xsl=article.xsl&