Revista Iberoamericana de Automática e Informática Industrial

Revista Iberoamericana de Automática e Informática Industrial 18 (2021) 113-118

www.revista-riai.org

Método de sincronización para sistemas distribuidos con seguridad funcional

Azketa, E.*, Mendialdua, X., Ibarguren, I., Solís, A.

Ikerlan Technology Research Centre, Basque Research and Technology Alliance (BRTA). P.º J.M. Arizmendiarrieta, 2. 20500 Arrasate/Mondragón, España.

To cite this article: Azketa, E., Mendialdua, X., Ibarguren, I., Solís, A. 2021. Synchronization method for distributed systems with functional safety. Revista Iberoamericana de Automática e Informática Industrial 18, 112-118. https://doi.org/10.4995/riai.2020.14022

Resumen

La sincronización temporal es un requisito clave en varios dominios de aplicación basados en sistemas de tiempo real distribuidos. Por ello es un campo de investigación que despierta interés, especialmente en líneas como la transferencia del tiempo y la frecuencia, el diseño de relojes y osciladores, y el uso de sincronización en redes de comunicación. El presente trabajo se centra en la transferencia del tiempo entre elementos de un sistema distribuido para la sincronización de sus relojes y el software que ejecutan. En la actualidad existen diferentes protocolos para sincronizar relojes distribuidos, pero tienen una serie de inconvenientes que los hace inadecuados para determinados tipos de sistemas. En este trabajo se presenta un método para sincronizar la ejecución de software distribuido en una red de área local. Adicionalmente, se desarrolla el análisis de seguridad funcional del método y se proponen las medidas que tiene que implementar para alcanzar un nivel SIL2. El método se ha implementado y se ha validado ejecutándolo en un sistema distribuido físico realista, obteniendo unos resultados prometedores.

Palabras clave: Sincronización de reloj, Sistemas distribuidos, Seguridad funcional, Redundancia.

Synchronization method for distributed systems with functional safety

Abstract

Time synchronization is a key requirement in several application domains based on real-time distributed systems. Therefore, it is a research area of interest, especially in lines such as the transfer of time and frequency, the design of clocks and oscillators, and the use of synchronization in communication networks. This work focuses on the transfer of time between elements of a distributed system for the synchronization of their clocks and the software they execute. Currently, there are different protocols to synchronize autonomous nodes, but they have some drawbacks that make them unsuitable for certain types of distributed systems. This paper presents a method for synchronizing the execution of distributed software on a local area network. Additionally, the functional safety analysis of the method is developed and the measures it must implement to achieve a SIL2 are proposed. The method has been implemented and validated by executing it in a realistic physical distributed system, obtaining promising results.

Keywords: Clock synchronization, Distributed systems, Functional safety, Redundancy.

1. Introducción

La sincronización temporal es un requisito clave en varios dominios de aplicación basados en sistemas de tiempo real distribuidos como las redes eléctricas inteligentes, los sistemas automatizados industriales y los sistemas de transporte, entre muchos otros.

La sincronización temporal es actualmente y seguirá siendo en el futuro un aspecto clave en la implementación de sistemas ciber-físicos y empotrados de tiempo real distribuidos (Weiss, 2015). En este campo la investigación se está centrando en líneas como la transferencia del tiempo y la frecuencia, el diseño de relojes y osciladores, y la utilización de temporización sincronizada en redes de comunicaciones.

En el presente trabajo se propone un método para la transferencia del tiempo entre elementos de un sistema distribuido con seguridad funcional que permite sincronizar el software que ejecutan.

*Autor para correspondencia: eazketa@ikerlan.es

Este artículo está organizado de la siguiente manera. En el Apartado 2 se hace un análisis de los métodos de sincronización más importantes de la actualidad. En el Apartado 3 se describe el nuevo método de sincronización propuesto. En el Apartado 4 se presenta un concepto de la seguridad funcional del método de sincronización propuesto. En el Apartado 5 se expone la implementación y la validación del método. Finalmente, en el Apartado 6 se comentan las conclusiones y el trabajo futuro.

2. Análisis de métodos de sincronización

En (Lévesque, 2016) se presenta una revisión exhaustiva sobre sincronización de relojes en redes de conmutación de paquetes. En ese trabajo se exponen los fundamentos básicos de la sincronización de relojes y se hace un análisis detallado de las principales tecnologías relacionadas. Las más relevantes con relación al tipo de sistema abordado en el presente artículo se describen en los siguientes párrafos.

El Network Time Protocol (NTP) (Mills, 2010) es un protocolo para sincronizar relojes en redes de computadores usando un conjunto de servidores y clientes repartidos. Utiliza algoritmos especiales para estimar el tiempo de retraso promedio entre el servidor NTP y cada cliente individual en la red. NTP está desarrollado específicamente para operar en redes con tiempos de transmisión de mensajes variables, como por ejemplo Internet. Típicamente se puede llegar a una precisión temporal de decenas de milisegundos en Internet y por debajo de un milisegundo en redes de área local en condiciones ideales. Existe una versión de NTP denominada Simple Network Time Protocol (SNTP) que funciona únicamente en el lado del cliente y no provee los mecanismos de filtrado, estadísticas y seguridad de NTP.

El Precision Time Protocol (PTP) (Eidson, 2006) es un protocolo de comunicaciones con sincronización temporal de red para sincronizar los relojes de dispositivos conectados a una red Ethernet. PTP es un mecanismo de sincronización temporal relativa en el que se selecciona un participante para que actúe como reloj maestro y envíe mensajes de sincronización temporal a todos los dispositivos esclavos. A diferencia de NTP, PTP está integrado en la capa física y realiza un etiquetado temporal basado en hardware, lo que le permite alcanzar precisiones por debajo de un microsegundo. El principal inconveniente de PTP es que requiere hardware específico que soporte el protocolo.

Los sistemas globales de navegación por satélite (GNSS) (Hofmann-Wellenhof, 2007) como el GPS estadounidense, el Galileo europeo y el GLONASS ruso, son sistemas de radionavegación basados en una constelación de satélites geoestacionarios sincronizados a relojes atómicos. Los satélites emiten periódicamente determinada información espacial y temporal que permite a un aparato receptor calcular su propia posición global y el tiempo coordinado universal (UTC). La precisión temporal teórica del GPS es de decenas de nanosegundos, pero en la práctica la precisión que se obtiene es de un centenar de nanosegundos debido a la pérdida de precisión que sufren la mayoría de los receptores en la interpretación de la señal. El principal inconveniente de los GNSS es que requieren hardware específico que soporte el protocolo.

Las tecnologías descritas pueden ser inadecuadas para sincronizar los relojes de los nodos conectados a una red de área local por múltiples razones. En el caso de NTP, no siempre es posible o deseable instalar un servidor dentro de la red de área local y con los servidores públicos disponibles en Internet puede no obtenerse el grado de sincronización requerido. En cuanto a PTP, no siempre hay disponible hardware que lo soporte para los nodos del sistema y si lo hay puede no ser posible o deseable adquirirlo o instalarlo. Por su parte, los GNSS presentan los mismos inconvenientes que el PTP con el añadido de que la cobertura de la señal en la ubicación del sistema puede no ser la adecuada.

Estas tecnologías actúan sobre el reloj horario del nodo, que determina el tiempo real, pero la ejecución del software está gobernada por el reloj del planificador, que es un contador monotónico creciente iniciado en el arranque del nodo. Para sincronizar la ejecución del software de los diferentes nodos se tiene que modificar el reloj del planificador. Esta operación ha de realizarse de manera correcta y precisa para evitar ejecuciones del software no deseadas, especialmente en sistemas con seguridad funcional.

3. Propuesta de método de sincronización

3.1. Arquitectura

El sistema distribuido está compuesto de nodos de ejecución conectados por una red de comunicaciones de área local. Idealmente es una red basada en prioridades donde las comunicaciones relacionadas con la sincronización tienen la prioridad más alta. En su defecto, es una red conmutada donde la segmentación evita que las comunicaciones relacionadas con la sincronización sufran retrasos innecesarios. Las tramas de red ofrecen una carga útil de al menos los bytes necesarios para el mensaje de sincronización de mayor longitud. La tarjeta de red o su controlador registran y ponen a disposición del software el tiempo del reloj horario en el que se reciben las tramas. Esta funcionalidad puede encontrarse en diversas tarjetas comerciales.

La arquitectura se basa en el paradigma cliente-servidor. Un nodo del sistema distribuido se configura como servidor y el resto como clientes. Los clientes conocen la dirección del servidor, pero el servidor no conoce a los clientes.

La Figura 1 muestra los conceptos que se detallan a continuación mediante un diagrama que refleja la ejecución temporal del software en el servidor y en el cliente que trata de sincronizarse con el servidor. En todos los nodos el software se ejecuta periódicamente con el mismo tiempo de ciclo T_{cycle}. Al comienzo de este ciclo se reserva un tiempo T_{sync} para el proceso de sincronización, en su variante servidor P_S o cliente P_C, dependiendo del nodo. El tiempo restante de T_{cycle} se reparte como proceda entre los procesos del software de aplicación P_A, que en los diferentes nodos pueden ser iguales o distintos como en el ejemplo de la Figura 1. T_{cycle} y T_{sync} son tiempos fijos de la planificación del sistema y sus valores dependen de la aplicación y de la implementación del software. El único requisito es que dentro de T_{sync} tiene que dar tiempo a los P_C a enviar una petición y procesar una respuesta, y al P_S a responder al menos a una petición de cada uno de los P_C. Para hacer efectivos T_{cycle} y T_{sync} se pueden emplear mecanismos

como, por ejemplo, la planificación basada en particiones o la activación de los procesos mediante temporizadores.

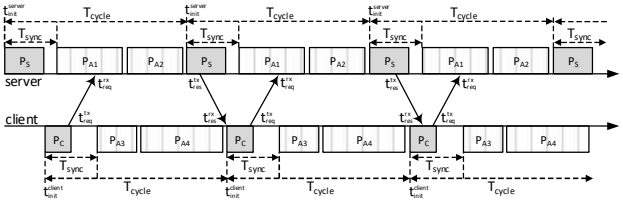


Figura 1: Conceptos del método de sincronización.

3.2. Protocolo

Al comienzo de cada ciclo los nodos servidor y cliente ejecutan sus P_S y P_C, respectivamente, y a continuación sus P_A. En cada ciclo, los P_C establecen una sesión de sincronización con el P_S enviándole un mensaje de petición con un identificador de sesión, cuya necesidad se justifica en el análisis de seguridad funcional presentado en el Apartado 4. El cliente registra el tiempo del reloj horario t_{req}^{tx} en el que envía el mensaje de petición.

El P_S registra el tiempo del reloj horario t_{req}^{rx} en el que recibe el mensaje de petición. Después, envía al P_C un mensaje de respuesta con el mismo identificador de sesión recibido en el mensaje de petición, el tiempo del reloj horario t_{init}^{server} en el que se ha iniciado el ciclo actual en el nodo servidor, el tiempo del reloj horario t_{req}^{rx} en el que ha recibido el mensaje de petición, el tiempo del reloj horario t_{res}^{tx} en el que va a enviar el mensaje de respuesta y el CRC de todos ellos. La necesidad de incluir el identificador de sesión y el CRC en el mensaje se justifica en el análisis de seguridad funcional del Apartado 4.

El P_C registra el tiempo del reloj horario t_{res}^{rx} en el que recibe el mensaje de respuesta. Después, si el CRC y el identificador de sesión son correctos, ejecuta el algoritmo de sincronización.

3.3. Algoritmo

El P_C calcula la desviación θ del reloj horario de su nodo con respecto al del nodo servidor mediante la Ecuación 1, de manera análoga a NTP.

$$\theta = \frac{(t_{req}^{rx} - t_{req}^{tx}) + (t_{res}^{tx} - t_{res}^{rx})}{2} \quad (1)$$

Un tiempo obtenido con el reloj horario del nodo cliente puede trasladarse a la referencia temporal de un tiempo obtenido con el reloj horario del nodo servidor simplemente sumándole la desviación. Así, NTP sincroniza el reloj horario del cliente sumándole esta desviación. Sin embargo, el objetivo del algoritmo propuesto no es sincronizar los relojes horarios sino la ejecución del software. Para ello, el P_C normaliza el tiempo del reloj horario en el que se ha iniciado su ciclo actual t_{init}^{client} sumándole a éste la desviación θ, como puede verse en la Ecuación 2, donde *N* significa normalizado.

$$t_{init,N}^{client} = t_{init}^{client} + \theta \quad (2)$$

Con los tiempos de inicio de ciclo del nodo cliente y del nodo servidor situados en la misma referencia temporal, el P_C calcula el error de sincronización ε haciendo la resta de ambos como se muestra en la Ecuación 3.

$$\varepsilon = t_{init}^{server} - t_{init,N}^{client} \quad (3)$$

Basándose en ε, el P_C modifica el tiempo del reloj del planificador para alargar o acortar el tiempo de ejecución reservado para el propio P_C. Como consecuencia, también se alarga o se acorta la duración del ciclo, de modo que la diferencia entre t_{init}^{server} y $t_{init,N}^{client}$, o sea ε, disminuye en cada ciclo. Cuando el valor absoluto de ε es inferior a un umbral, se considera que el software del nodo cliente está sincronizado con el del servidor. El peor caso de sincronización se produce cuando $|\varepsilon| = T_{cycle} / 2$.

Un ε positivo es el resultado de que $t_{init}^{server} > t_{init,N}^{client}$, lo que significa que el ciclo del nodo cliente ha comenzado antes que el del nodo servidor. En consecuencia, el cliente tiene que alargar la duración del ciclo con el objetivo de disminuir la diferencia en el siguiente ciclo. Para alargar la duración del ciclo hay que retrasar el reloj del planificador, pero nunca hasta un tiempo anterior a t_{init}^{client} para evitar que una nueva instancia del P_C se ejecute en el ciclo actual cuando el reloj del planificador alcance otra vez el valor t_{init}^{client}. En la Figura 2 puede observarse que el tiempo reservado para el P_C es más largo en los ciclos segundo y tercero, retrasando el comienzo del P_{A3}. Así los P_A del cliente y del servidor acaban ejecutándose en sincronía a partir del tercer ciclo.

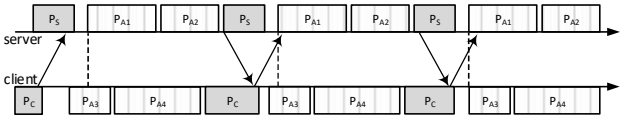


Figura 2: Ciclos del servidor y del cliente cuando ε es positivo.

Un ε negativo es el resultado de que $t_{init}^{server} < t_{init,N}^{client}$, lo que significa que el ciclo del nodo cliente ha comenzado más tarde que el del nodo servidor. En consecuencia, el cliente tiene que acortar la duración del ciclo con el objetivo de disminuir la diferencia en el siguiente ciclo. Para acortar la duración del ciclo hay que adelantar el reloj del planificador, pero nunca hasta un tiempo posterior a $t_{init}^{client}+T_{sync}$ para evitar quitarle tiempo de ejecución al primer P_A. En la Figura 3 puede observarse que el tiempo reservado para el P_C es más corto en los ciclos segundo y tercero, adelantando el comienzo del P_{A3}. Así los P_A del cliente y del servidor acaban ejecutándose en sincronía a partir del tercer ciclo.

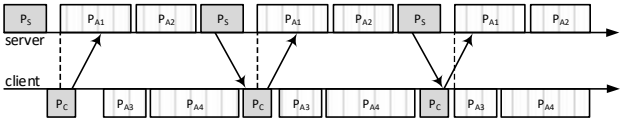


Figura 3: Ciclos del servidor y del cliente cuando ε es negativo.

Siendo t_{sync}^{client} el tiempo del reloj del planificador en el que el P_C realiza la modificación del mismo, el nuevo tiempo t_{new}^{client} de dicho reloj del nodo cliente se calcula con la Ecuación 4.

$$t_{new}^{client} = \begin{cases} max\left(t_{sync}^{client} - \varepsilon,\ t_{init}^{client}\right), & \varepsilon > 0 \\ min\left(t_{sync}^{client} - \varepsilon,\ t_{init}^{client} + T_{sync}\right), & \varepsilon < 0 \end{cases} \quad (4)$$

Típicamente, los relojes sufren derivas debidas a múltiples factores como, por ejemplo, la imperfección de los osciladores o los cambios de temperatura, entre otras causas. Por ello, el procedimiento de sincronización se realiza periódicamente al inicio de cada ciclo. Además, también en cada ciclo, el P_C informa al software de aplicación sobre si el nodo está sincronizado o no.

4. Seguridad funcional en el método de sincronización

4.1. Definiciones

La seguridad (*safety*) se define como la ausencia de riesgos inaceptables de daños a la salud de las personas, al medio ambiente y a los equipamientos, ya sea directa o indirectamente. La seguridad funcional (*functional safety*) es la parte de la seguridad general enfocada a que un sistema o equipamiento funcione al tiempo que se controlan los riesgos que un mal funcionamiento puede provocar. El objetivo de la seguridad funcional es la reducción de los riesgos y de su impacto negativo hasta un nivel tolerable, puesto que el riesgo cero no existe. El *Safety Integrity Level* (SIL) es el nivel relativo de reducción de los riesgos que provee una función de seguridad. El SIL también puede especificar el nivel objetivo de reducción de riesgo que requiere alcanzar una función de seguridad. Los riesgos se cuantifican en base a la probabilidad de que se produzca un evento determinado y a la severidad que éste tendría; es decir, en base al daño que podría causar.

La seguridad funcional contempla la detección de condiciones potencialmente peligrosas que resultan en la activación de dispositivos o mecanismos de protección o corrección para evitar que se produzcan eventos peligrosos o para mitigar las consecuencias de estos. La seguridad funcional se basa en sistemas activos como, por ejemplo, la detección de humo mediante sensores y la consiguiente activación de un sistema de extinción de incendios. No se considera seguridad funcional la seguridad lograda por medidas que dependen de sistemas pasivos como, por ejemplo, una puerta resistente al fuego o un aislamiento para soportar altas temperaturas.

Los eventos o condiciones potencialmente peligrosos suelen ser provocados por fallos del sistema. Se distinguen dos tipos de fallos: aleatorios y sistemáticos.

Los fallos aleatorios son provocados por efectos físicos como la corrosión, el estrés térmico, el desgaste, las ondas electromagnéticas, la radiación, etc. Afectan únicamente a los componentes hardware simples de un sistema. Basándose en resultados de pruebas y datos históricos se puede producir información estadística sobre este tipo de fallos. En consecuencia, es posible calcular la probabilidad media y por tanto el riesgo asociado a la ocurrencia de un fallo aleatorio.

Los fallos sistemáticos son provocados por errores humanos durante el desarrollo y la operación del sistema. Un fallo sistemático se produce siempre, cuando se dan las circunstancias adecuadas, hasta que es eliminado. Afectan al software y a los componentes hardware complejos de un sistema. El riesgo de la introducción de este tipo de fallos se puede controlar mediante la aplicación de determinadas metodologías de desarrollo y operación.

La IEC 61508 (IEC, 2010) es el estándar de seguridad funcional principal y cubre sistemas eléctricos, electrónicos y de electrónica programable. De la IEC 61508 se derivan estándares sectoriales adaptados a las características específicas del dominio en cuestión como la IEC 62304 (IEC, 2015) para dispositivos médicos, la IEC 62061 (IEC, 2005) para la industria de la maquinaria, la ISO 26262 (ISO, 2018) para automoción y la EN 50128 (EN, 2012) para la industria ferroviaria.

4.2. Análisis y metodología aplicada

Este método de sincronización está diseñado para aplicaciones que ejecutan alguna función con un nivel de riesgo potencial que depende de si el sistema está sincronizado o no. La función se ejecuta cuando el método de sincronización dice que el sistema está sincronizado. Si la función se ejecuta cuando el sistema dice estar sincronizado y realmente lo está, el riesgo potencial se reduce a un nivel aceptable. Sin embargo, si la función se ejecuta cuando el sistema dice estar sincronizado pero realmente no lo está, el riesgo potencial se eleva a un nivel inaceptable. Por lo tanto, el método de sincronización no puede decir que el sistema está sincronizado si realmente no lo está. El nivel de riesgo potencial mencionado es SIL2, lo que significa que el método de sincronización puede dar un falso positivo como máximo una vez cada aproximadamente 115 años de funcionamiento.

Para hacer un análisis de la seguridad funcional de un sistema se requiere toda la información sobre su diseño e implementación. Este sistema está compuesto por tres tipos de elementos: el software de sincronización, los nodos de ejecución y la red de comunicaciones.

El software de sincronización únicamente puede verse afectado por fallos sistemáticos y para evitarlos está desarrollado de acuerdo con la metodología que exige el estándar para SIL2.

Los nodos de ejecución son un computador industrial propietario con un procesador PowerPC y el sistema operativo de tiempo real INTEGRITY (Green Hills Software, a), que ofrece particionado espacial y temporal. Este conjunto implementa todos los mecanismos necesarios para SIL2 y ha sido desarrollado de acuerdo con la metodología que exige el estándar para dicho nivel de integridad.

La red de comunicaciones es una Ethernet conmutada. El controlador de la tarjeta de red de los nodos de ejecución es un software desarrollado de acuerdo con la metodología que exige el estándar para SIL2. Sin embargo, el hardware de la red de comunicaciones no implementa ningún mecanismo de detección de los fallos aleatorios que pueden producirse durante la comunicación entre los nodos. Este tipo de red se denomina *black channel* y requiere implementar una serie de mecanismos a nivel de software. El análisis de los fallos y de los mecanismos de detección necesarios se realiza de acuerdo con el estándar de seguridad funcional de las comunicaciones de sistemas ferroviarios EN 50159 (EN, 2011). Según el estándar, la red de comunicación bajo análisis es de Categoría 1, que se define como "un sistema de transmisión cerrado cuyas propiedades esenciales están bajo el control del diseñador y para el que se puede definir un conjunto simplificado de requisitos de seguridad". Para esta categoría de sistemas se tienen que analizar los siguientes fallos: eliminación, retraso, inserción, reordenación, repetición y corrupción de mensajes.

La eliminación significa que un mensaje se elimine accidentalmente. Si se elimina el mensaje de petición de sincronización, el servidor no envía el mensaje de respuesta, el cliente se mantiene indefinidamente a la espera de recibirlo y en consecuencia no se actualiza el estado de sincronización. Si se elimina el mensaje de respuesta, el cliente no recibe el mensaje antes de que finalice el tiempo que tiene reservado para ejecutarse dentro del ciclo, no se sincroniza y publica que no está sincronizado. Por lo tanto, la eliminación no es un fallo potencialmente peligroso porque no da un falso positivo y en consecuencia no requiere ningún mecanismo de detección.

El retraso significa que un mensaje llegue a su destino más tarde de los esperado. Si se retrasa tanto el mensaje de petición de sincronización como el de respuesta, existe el riesgo de que éste no llegue al cliente en el ciclo esperado. En este caso el cliente no recibe el mensaje antes de que finalice el ciclo, y por tanto no se sincroniza y publica que no está sincronizado. Esta situación no es potencialmente peligrosa porque no tiene como consecuencia un falso positivo. Sin embargo, el mensaje de respuesta retrasado se procesa en el siguiente ciclo, donde los tiempos de ese mensaje no son consistentes con los registrados por el cliente en ese ciclo. Así, el cliente se sincroniza con datos erróneos y publica que sí está sincronizado cuando en realidad puede que no lo esté. Esta situación sí es potencialmente peligrosa y para detectar el fallo que la origina se utiliza un identificador de sesión que permite asociar los datos recibidos con los registrados. En el método de sincronización propuesto se utiliza como identificador de sesión un número de un byte de longitud que cambia en cada ciclo.

La inserción significa que un mensaje sea introducido accidentalmente. Si se inserta un mensaje de petición de sincronización con destino el servidor, éste envía un mensaje de respuesta a la dirección de origen del mensaje de petición. Si se inserta un mensaje de respuesta con destino el cliente, éste se sincroniza con los datos del mensaje, que son potencialmente erróneos porque no son respuesta a una petición suya. Así, publica que sí está sincronizado cuando en realidad puede que no lo esté. Para detectar este fallo se utiliza el identificador de sesión.

La reordenación significa que el orden de los mensajes sea alterado accidentalmente. Para que en este método de sincronización ocurra una reordenación, en un ciclo el cliente no recibe el mensaje de respuesta, en el siguiente recibe el mensaje esperado y en el siguiente recibe el mensaje que no recibió dos ciclos antes. En el ciclo en el que no recibe el mensaje, el cliente no se sincroniza y publica que no está sincronizado. En el ciclo en el que recibe el mensaje esperado, el cliente se sincroniza y publica que sí está sincronizado. Y en el ciclo en el que recibe el mensaje correspondiente a dos ciclos antes, los tiempos recibidos no son consistentes con los registrados por el cliente, por lo que se sincroniza con datos erróneos y publica que sí está sincronizado cuando en realidad puede que no lo esté. Para detectar este fallo se utiliza el identificador de sesión.

La repetición significa que se envíe un mensaje repetido accidentalmente. Si se repite tanto el mensaje de petición de sincronización como el de respuesta, el cliente recibe dos mensajes de respuesta idénticos. En un ciclo se sincroniza según los datos del primer mensaje, que son consistentes con los tiempos registrados en dicho ciclo, y publica que sí está sincronizado. En el siguiente ciclo se sincroniza según los datos del segundo mensaje, que no son consistentes con los tiempos registrados en dicho ciclo ya que pertenecen a uno anterior, y publica que sí está sincronizado cuando en realidad puede que no lo esté. Para detectar este fallo se utiliza el identificador de sesión.

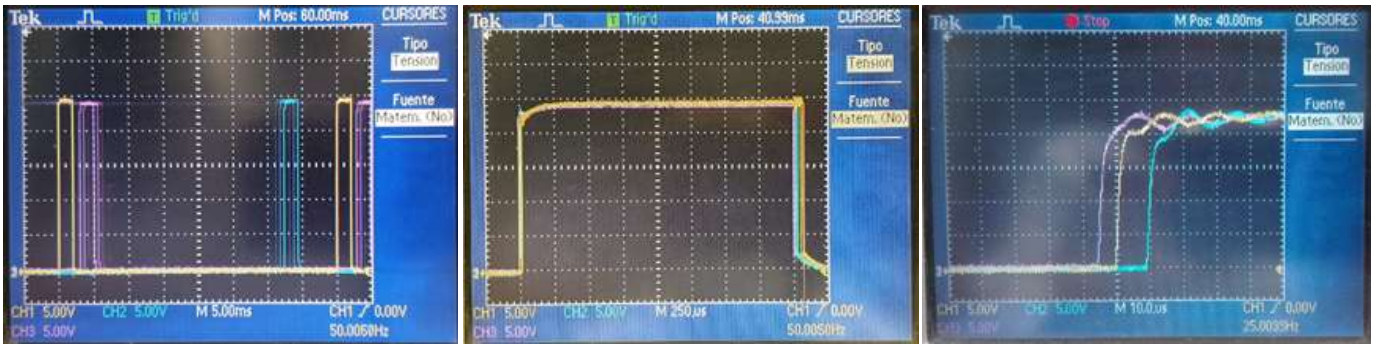
La corrupción significa que un mensaje se transforme accidentalmente en otro mensaje formalmente correcto. Si en el mensaje de petición de sincronización se corrompe el identificador de sesión, el servidor le envía el mensaje de respuesta al cliente, éste lo descarta porque el identificador de sesión es incorrecto, no se sincroniza y publica que no está sincronizado. Esto no es un fallo potencialmente peligroso porque no da un falso positivo. Si en el mensaje de respuesta se corrompe el identificador de sesión, el cliente descarta el mensaje porque es incorrecto, no se sincroniza y publica que no está sincronizado. Esto tampoco es un fallo potencialmente peligroso porque no da un falso positivo. Sin embargo, si se corrompe el tiempo de inicio del ciclo, el tiempo de recepción del mensaje de petición o el tiempo de transmisión del mensaje de respuesta, el cliente se sincroniza con datos erróneos sin saberlo y publica que sí está sincronizado cuando en realidad puede que no lo esté. Esta situación es potencialmente peligrosa y para detectar el fallo que la origina se utiliza un CRC que permite detectar modificaciones en los datos.

El estándar (EN, 2011) explica en su Anexo C.4 la forma de comprobar si un CRC de una longitud dada es suficiente para cumplir los requisitos de seguridad de un sistema SIL2. En este caso se ha seleccionado un CRC de 32 bits y se ha comprobado que es suficiente siempre que la frecuencia de recepción de mensajes sea menor o igual que $4.597\text{x}10^{14}$ mensajes por milisegundo. Como en este sistema el cliente recibe un mensaje por ciclo, para que un CRC de 32 bits sea suficiente, el tiempo de ciclo tiene que ser mayor o igual que el inverso de la frecuencia: $T_{cycle} \geq 2.175\text{x}10^{-15}$ milisegundos.

5. Implementación y validación

El computador de los nodos tiene tres relojes. Por una parte, la CPU tiene un reloj horario, cuya precisión del orden de milisegundos es insuficiente como referencia para el método de sincronización, y el reloj del planificador. Por otra parte, la tarjeta Ethernet tiene un reloj independiente y suficientemente preciso. La tarjeta adjunta a cada trama recibida su instante de recepción en base al reloj de la propia tarjeta y ya desde el software se asigna ese instante a t_{req}^{rx} o t_{res}^{rx}, según el caso. Sin embargo, la tarjeta no registra el instante de las tramas enviadas ni existe un API para acceder al reloj. Por ello, para obtener el tiempo de envío de un mensaje, primero se envía una trama de control a *loopback*, después se procesa la trama de control para obtener su instante de recepción y asignarlo a t_{req}^{tx} o t_{res}^{tx}, y finalmente se envía el mensaje. Como este proceso se realiza lo más rápido posible, los valores de t_{req}^{tx} y t_{res}^{tx} son suficientemente precisos, aunque no son exactos.

Por otra parte, la API no permite restar de forma directa y correcta el error de sincronización al reloj del planificador. Por ello, se obtiene el tiempo t_{sync}^{client} de este reloj, se resta el error de sincronización a ese tiempo y se asigna el nuevo tiempo t_{new}^{client} al reloj, con la pérdida de precisión que conlleva este proceso.



(a) (b) (c)

Figura 4: (a) Ondas cuando los clientes están desincronizados (arranque). (b) Ondas cuando los clientes están sincronizados. (c) Vista ampliada del flanco de subida de las ondas sincronizadas.

El método de sincronización se ha implementado en C++ y compilado con MULTI IDE (Green Hills Software, b). Se ha creado una aplicación con un ciclo de ejecución compuesto de tres particiones y un T_{cycle} de 40 milisegundos. La primera partición contiene el P_S o P_C dependiendo del nodo y un T_{sync} de 1 milisegundo. La segunda partición contiene los P_A y tiene una duración de 4 milisegundos. La tercera partición contiene unos procesos auxiliares requeridos por el nodo, como por ejemplo un servidor FTP para cargar ficheros, y tiene una duración de 35 milisegundos.

El ciclo se ejecuta en tres nodos, uno de los cuales contiene un P_S y los otros dos P_C. Uno de los P_A activa y desactiva una salida digital durante su ejecución, formando una onda cuadrada periódica de 2 milisegundos de anchura. Las salidas digitales de los tres nodos están conectadas a sendos canales de un osciloscopio.

La Figura 4(a) muestra las ondas en el arranque de los nodos. La onda de color amarillo corresponde al servidor y las otras dos a los clientes. Los nodos arrancan de forma independiente y desincronizados de manera aleatoria, como demuestra el desalineamiento de las ondas. Sea cual sea la desincronización de los relojes de los nodos, el mayor desalineamiento o error de sincronización que puede haber entre las ondas es de más-menos la mitad de su periodo, ±20 milisegundos en este caso, tal y como se expone en el Apartado 3.3. En esta ejecución, los clientes magenta y cian arrancan con un error de sincronización de -2.5 y 7.5 milisegundos, respectivamente.

En cuanto comienza a actuar el método de sincronización, las ondas de los clientes van acercándose a la del servidor hasta quedar alineadas, como puede observarse en la Figura 4(b). El tiempo que tarda un cliente en sincronizarse completamente con el servidor depende de su propia implementación, de T_{cycle} y de T_{sync}. En este caso los clientes tardan de media del orden de decenas de milisegundos.

La Figura 4(c) muestra una vista ampliada del flanco de subida de las ondas cuando los clientes están sincronizados con el servidor. La onda situada en el centro corresponde al servidor. Puede observarse que en este instante concreto los clientes presentan un error de sincronización de -10 y 5 microsegundos. El error de sincronización de los clientes varía en cada ciclo, pero el rango en el que oscila, y por lo tanto la precisión que consigue esta implementación del método de sincronización, es de ±15 microsegundos. Aunque en teoría es posible alcanzar una precisión mayor, en la práctica lo impide la inexactitud de t_{req}^{tx}, t_{res}^{tx} y t_{new}^{client} debida a las razones expuestas al inicio del Apartado 5.

6. Conclusiones y trabajo futuro

El método de sincronización presentado en este trabajo supera los inconvenientes de los protocolos de sincronización actuales de una forma sencilla y efectiva porque: no requiere hardware específico, permite sincronizar la ejecución del software y puede ser utilizado en aplicaciones con seguridad funcional SIL2.

Como trabajo futuro se plantean tres líneas de exploración. Por una parte, se plantea evaluar el comportamiento del método para diferentes valores de T_{cycle} y T_{sync}. Por otra parte, se plantea que un proceso de sincronización pueda adoptar el rol tanto de servidor como de cliente y que un cliente pueda convertirse en el servidor ante la caída de éste. Y finalmente, se plantea realizar el análisis del sistema para su uso en aplicaciones con seguridad funcional superior a SIL2.

Referencias

Eidson, J. C., 2006. Measurement, Control, and Communication Using IEEE 1588 (Advances in Industrial Control). Springer.

EN, 2011. EN 50159: Railway applications - Communication, signalling and processing systems - Safety-related communication in transmission systems.

EN, 2012. EN 50128: Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems.

Hofmann-Wellenhof, B., Lichtenegger, H., Wasle, E., 2007. GNSS–global navigation satellite systems: GPS, GLONASS, Galileo, and more. Springer Science & Business Media.

IEC, 2005. IEC 62061: Safety of machinery - Functional safety of safety-related electrical, electronic and programmable electronic control systems.

IEC, 2010. IEC 61508: Functional safety of electrical/electronic/ programmable electronic safety-related systems.

IEC, 2015. IEC 62304: Medical device software - Software life cycle processes.

ISO, 2018. ISO 26262: Road vehicles - Functional safety.

Lévesque, M., Tipper, D., 2016. A survey of clock synchronization over packet-switched networks. IEEE Communications Surveys & Tutorials 18.4, 2926-2947.

Mills, D. L., 2010. Computer Network Time Synchronization: The Network Time Protocol on Earth and in Space, Second Edition. CRC Press.

Green Hills Software. INTEGRITY Real-Time Operating System, https://www.ghs.com/products/rtos/integrity.html.

Green Hills Software. MULTI Integrated Development Environment, https://www.ghs.com/products/MULTI_IDE.html.

Weiss, M., Eidson, J., Barry, C., Broman, D., 2015. Time-aware applications, computers, and communication systems (TAACCS). NIS.