Capítulo 2

In-Vehicle Networking

En los últimos años, la comunicación entre los dispositivos electrónicos ha sido tratado como un tema de gran importancia en el diseño de sistemas electrónicos embebidos. Bucles de control software y monitorización asistida por computador han sido implementados con componentes inteligentes distribuidos físicamente en el interior del entorno bajo control (por ejemplo, un vehículo). En esos casos, se impone la necesidad de utilizar una estructura de interconexión de relativo bajo coste y suficiente capacidad para soportar los requisitos de rendimiento y tiempo. Debido a la gran hetereogeneidad de estos dispositivos electrónicos, la comunicación a través de un bus compartido se ha considerado como la solución más razonable en casi todos los casos.

Actualmente, hay un gran interés en la industria automovilística por los vehículos controlables de forma digital, donde los tradicionales componentes hidráulicos y mecánicos se reemplazan por actuadores y sensores (sistemas *x-by-wire*). Estas aplicaciones son consideradas de seguridad crítica porque involucran la seguridad de personas o equipamientos de alto coste.

Para los requisitos de comunicación de estos sistemas de control digitales, se han establecido dos alternativas de planificación conceptualmente distintas, que pueden tener una influencia fundamental en sus propiedades. Estas dos alternativas son:

- mensajes provocados por eventos (*event-triggered*)
- mensajes provocados según algun patrón temporal (*time-triggered*)

En sistemas *time-triggered* los mensajes son periódicos por naturaleza. Todas las transiciones internas del sistema están asociadas a un punto predeterminado del dominio de tiempo discreto y circular. Estos sistemas se caracterizan por su comportamiento regular en el tiempo. El controlador de comunicaciones es responsable de todas las comunicaciones y opera de forma independiente al microprocesador del dispositivo en cuestión.

El mayor problema del esquema *time-triggered* es la falta de flexibilidad. Para añadir un nodo que no se hubiera tenido en cuenta durante la fase de diseño, todo el sistema debe ser reprogramado. La mayor ventaja es que ofrecen latencias fijas en la comunicación de datos. El ancho de banda se divide en *slots* de tiempo dedicados en los cuales sólo uno de los nodos puede transmitir, evitando por tanto que ocurra cualquier colisión.

Diversos estudios han reconocido a las comunicaciones *time-triggered* como las más adecuadas para as aplicaciones que requieren una seguridad crítica. Estas arquitecturas de bus presentan algunas propiedades altamente deseables como pequeños *jitters* (retrasos irregulares en señales generadas periódicamente), retrasos de transmisión predecibles y soporte para técnicas de tolerancia a fallos.

Actualmente las dos soluciones más prometedoras dentro de esta tecnología son el protocolo TTP/C y FlexRay.

En sistemas *event-triggered*, las transiciones no tienen ninguna planificación temporal; en el peor escenario posible, se entraría en una fase de arbritraje que resultaría en latencias no deseables. Todas las acciones son provocadas debido a la ocurrencia de cualquier evento o cambio de estado. La mayor ventaja es la rapidez de respuesta a eventos externos cuando la carga del bus es baja. La reacción del sistema es rápida ya que los tiempos de transmisión de los mensajes no están bajo control de ninguna arquitectura; un nodo puede intentar el acceso al medio cuando lo desee. El problema es que, dependiendo del mecanismo de arbitraje del bus, las colisiones tendrán un impacto directo en esta rapidez de respuesta.

Además, los sistemas *event-triggered* son más flexibles. El ancho de banda del medio de transmisión se puede compartir entre todos los nodos, lo que significa que se readapta automáticamente cuando se añade o elimina uno de ellos. Por contra, añadir nuevos nodos, podría empeorar los tiempos de respuesta de forma incontrolada. De hecho, incluso en situaciones de baja carga, se experimentan variaciones en los distintos tiempos, lo que tiene un impacto negativo en los sistemas de control. En la Tabla 2.1 se puede ver una breve comparación entre ambos tipos de sistemas.

	Event-triggered	Time-triggered
Reacción a eventos externos	rápido	lento
Latencia de datos	variable	fijo
Ancho de banda	compartido	dedicado
Complejidad de diseño	baja	media

Tabla 2.1: Comparativa entre sistemas *event-triggered* y *time-triggered*

CAN, una red cuya mayor parte de implementaciones en la práctica son del tipo *event-triggered*, se ha convertido en la red de comunicación de mayor éxito en automóviles.

Hay que destacar que ambas soluciones no son mutuamente exclusivas en la práctica, aunque siempre una prevalece. Por ejemplo, Flexray proporciona servicios *event-triggered* como complemento a sus estrictas comunicaciones *time-triggered*.

La definición de todos los posibles fallos que pueden afectar al sistema de comunicaciones es un paso importante previo al diseño de la red. El sistema debe proporcionar servicios en los que se pueda tener un cierto grado de confianza justificadamente. Esto es lo que se entiende por fiabilidad. Para conseguirla es necesario que las arquitecturas de bus proporcionen soporte para enmascarar fallos.

Se debe asegurar que un determinado fallo no se propague de forma descontrolada. En un bus, los errores se pueden propagar por todo el medio de forma que desbaraten la comunicación entre los nodos que funcionan correctamente. Por tanto, todos los componentes deben ser lógicamente aislados en caso de que ocurra un fallo. Existe el peligro de que un nodo defectuoso asuma un comportamiento incorrecto y monopolice el medio de transmisión mandado mensajes espurios en cualquier momento. Este problema es difícil de solucionar en sistemas *event-triggered* porque los tiempos de transmisión de los distintos mensajes son impredecibles. Por el contrario, en los sistemas *time-triggered* estos fallos se pueden aislar con facilidad gracias al concepto de guardianes de bus (BG, *Bus Guardian*).

Los BGs son componentes de la red (separados del controlador CAN) que permiten desactivar la salida de un nodo cuando no le está explícitamente permitido transmitir. Se les carga el patrón de comunicaciones de todo el sistema, y se comportan como interruptores inteligentes. Deben ser capaces de desconectar a un nodo defectuoso de manera que no interfiera con el resto, e idealmente debería ser implementado en un chip distinto al del controlador, consiguiéndose así una mayor seguridad. En

teoría, también deberían contar con relojes independientes de los del resto del sistema, lo cual no es siempre posible conseguir.

2.1 X-by-Wire

X-by-wire hace referencia a un conjunto de tecnologías [1] para la industria automovilística que reemplazan los sistemas de control hidráulicos y mecánicos tradicionales por sistemas de control electrónico que usan actuadores electromecánicos e interfaces usuario-máquina como emuladores de pedal, eliminando una serie de componentes tradicionales del vehículo.

La "X" representa a la acción comandada, como por ejemplo acelerar o frenar. Varios vehículos modernos están equipados con este tipo de sistemas y un respaldo mecánico combinado. En el futuro es de esperar que los respaldos mecánicos sean reemplazados por sistemas puramente electrónicos. Hay dos motivos principales para este cambio:

- Los fabricantes esperan reducir costes de fabricación utilizando estos nuevos dispositivos.
- Sería posible la integración de nuevas funcionalidades. Por ejemplo, el control de velocidad inteligente (ACC, *Adaptive Cruise Control*), que acelera y frena el coche de forma autónoma para mantener la distancia de seguridad con el vehículo delantero, está disponible de momento sólo en los últimos vehículos de gama alta.

El sistema de frenado es uno de los módulos básicos para la seguridad durante la conducción, de ahí que este sistema haya estado sujeto a una permanente evolución. Hoy en día los sistemas hidráulicos, asistidos por inventos como el ABS, son comunes en cualquier vehículo. Actualmente los departamentos de desarrollo de todos los fabricantes están trabajando en nuevos sistemas *brake-by-wire*, y las primeras versiones han sido incluidas en varios vehículos de última generación. La idea es que cables reemplacen a los sistemas hidráulicos y las órdenes sean transmitidas electrónicamente. Se pueden distinguir dos soluciones dentro de esta tecnología:

- Frenado electro-hidráulico (EHB, *Electro-Hydraulic Brake*): este sistema está mecánicamente desacoplado de la entrada. Un simulador se encarga de dar la sensación de frenado al conductor a través de distintos sensores y actuadores. La señal de los distintos sensores es enviada a la unidad de control electrónico (ECU) que procesa estos datos junto a otros como la velocidad o la aceleración. Usando estos datos la ECU calcula una señal de presión óptima para el frenado, que es transmitida a la unidad de control hidráulico. Ésta se encargará de crear la presión hidráulica que activará el frenado de acuerdo a la entrada. En caso de fallo en la alimentación eléctrica, se habilitaría una conexión hidráulica entre el pedal y las unidades de control. Usando datos que describen el estado del vehículo, funciones como el control de velocidad inteligente son fáciles de implementar. Además, se pueden alcanzar distancias de frenado más cortas, debido a la mejora en la dinámica del sistema.
- Frenado electro-mecánico (EMB, *Electro-Mechanical Brake*): en este caso se habla de un sistema *dry* (seco) *brake-by-wire*, porque no es necesario ningún tipo de fluido. La energía y las señales se transmiten sólo electrónicamente. Los componentes de entrada son similares a los del sistema anterior, simulándose un pedal cuya información llega hasta la ECU del vehículo. A partir de ahí, la información de frenado se transmitiría a los módulos de frenado de cada rueda, donde la fuerza se realizaría por actuadores electromecánicos. Un obstáculo en este caso lo suponen las potencias eléctricas necesarias para aplicar esta fuerza de frenado.

Un bus es utilizado para transmitir estas informaciones. Por tanto, para cumplir con los requisitos de seguridad es necesario que esta estructura de bus sea aplicada de forma redundante. La fuente de energía eléctrica también ha de ser redundante.

Por otro lado, se encuentran los sistemas *steer-by-wire*. En este caso no existe transmisión directa de potencia entre el volante del vehículo y los neumáticos. En sistemas intermedios como la dirección asistida, un motor ajusta la dirección de las ruedas en proporción a la velocidad del vehículo, consiguiéndose una conducción más confortable.

Los sistemas *steer-by-wire* reales consisten en varios sensores, motores eléctricos y controladores. Los sensores conectados al volante capturan la entrada del conductor y diversos actuadores proporcionan la sensación adecuada a éste. Otros actuadores (motores eléctricos) son los encargados de generar los cambios en la dirección del coche. Estas operaciones son coordinadas por una unidad de control electrónico central, que está conectada por un sistema de bus en tiempo real (CAN, TTP o FlexRay) a los demás dispositivos. Estos protocolos se introducen en los siguientes apartados.

2.2 Controller Area Network (CAN)

La denominada *Controller Area Network* (CAN) [2] consiste en un bus serie que fue desarrollado durante los años 80 para aplicaciones de automoción por Robert Bosch GbmH, Alemania. CAN fue desarrollado para soportar sistemas de control distribuidos en automóviles y también ha sido adoptado con éxito en sistemas de control industriales.

CAN usa un protocolo de transmisión serie con el que se pueden alcanzar tasas de hasta 1 Mbps en cables de par trenzado. Los mensajes se transmiten entre los distintos nodos del bus utilizando un identificador. Este identificador no identifica al nodo transmisor o receptor, sino el contenido del mensaje (por ejemplo, temperatura o posición de un determinado eje). Todos los nodos en el bus CAN reciben todos los mensajes y comprueban si alguno de ellos es de su interés. El identificador también determina la prioridad del mensaje. Los valores numéricos más bajos tienen una prioridad más alta dentro del bus CAN.

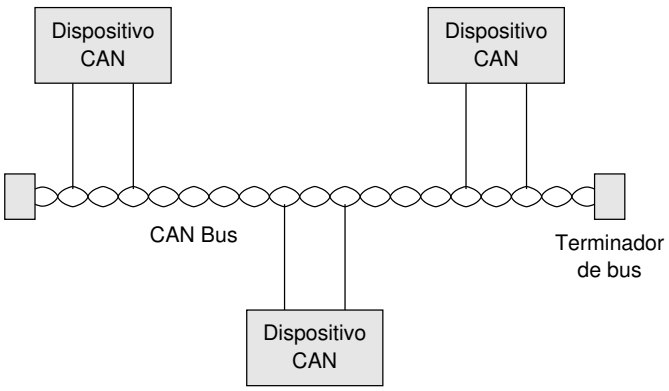

Figura 2.1: Estructura de un sistema CAN

Los primeros controladores CAN aparecieron en el mercado en 1987. Las versiones 1.0 y 1.2 de CAN definían un identificador de mensaje de 11 bits. En 1991 apareció la versión 2.0, permitiendo una extensión de 18 bits, con lo que se conseguía una longitud efectiva de 29 bits. Para mantener la compatibilidades de los nuevos dispositivos CAN con los anteriores, la especificación 2.0 se definió en 2 partes, 2.0A y 2.0B. En CAN 2.0A, el formato de mensaje es consistente con las versiones anteriores que usaban un identificador de 11 bits. En can 2.0B la extensión de 18 bits está permitida. Can 2.0B

se puede implementar en un modo pasivo, en el que el controlador sólo transmitirá identificadores de 11 bits y aceptará los de 11 bits y 29 bits; y en un modo activo, en el que podrá recibir y transmitir ambos tipos de mensajes. Las versiones 1.0, 1.2 y 2.0A se suelen denominar *standard CAN*, mientras que la versión 2.0B se denomina *extended CAN*.

CAN ha sido estandarizado por la Organización Internacional para la Estandarización (ISO, *International Organization for Standarization*). Se definen dos tipos de redes CAN: una red de alta velocidad (hasta 1 Mbps), bajo el estándar ISO 11898-2, destinada para controlar el motor e interconectar la unidades de control electrónico (ECU); y una red de baja velocidad tolerante a fallos (menor o igual a 125 Kbps), bajo el estándar ISO 11519-2/ISO 11898-3, dedicada a la comunicación de los dispositivos electrónicos internos de un automóvil como son control de puertas, techo corredizo, luces y asientos.

Como ya se ha mencionado, los mensajes CAN contienen un identificador para indicar los contenidos del mensaje en lugar de la dirección de algún dispositivo. Esto convierte a CAN en un bus multimaestro que usa comunicaciones multidifusión. Al ser un bus multimaestro, cualquier dispositivo puede determinar si este bus está disponible para la transmisión de cualquier mensaje. Si varios dispositivos empiezan a transmitir al mismo tiempo, entonces la prioridad del mensaje se utiliza para determinar que dispositivo completa la transmisión. Al ser multidifusión, cualquier dispositivo puede determinar si el mensaje recibido es de interés y actuar según la información recibido. El beneficio de ser multimaestro y multidifusión es que nuevos dispositivos pueden ser añadidos sin reconfigurar los ya existentes.

Cada nodo de un bus CAN consta de los siguientes elementos:

- Un microprocesador que controla los mensajes recibidos y decide qué se debe transmitir. Sensores, actuadores y dispositivos de control también podrían estar conectados a este microprocesador.
- Un controlador CAN (hardware con un reloj síncrono) que almacena los bits recibidos del bus hasta que un mensaje completo está disponible, de forma que pueda ser extraído por el microprocesador. Normalmente el controlador lanza una interrupción en este caso. Para el envío de mensajes, el microprocesador colocará el mensaje a transmitir en el controlador.
- Un transceptor que se encarga de adaptar los niveles de señal que el controlador espera a los del bus.

Un bus CAN tiene dos estados, uno que representa un 1 lógico, llamado el estado recesivo; y otro que representa un 0 lógico, llamado dominante. Cuando no hay tráfico en el bus, éste se encuentra en estado recesivo. Un dispositivo utilizando el bus pondrá a éste en estado dominante. Cuando no hay tráfico en el bus, éste se encuentra en estado recesivo. Un dispositivo utilizando el bus pondrá a éste en estado dominante. En el caso de un cableado consistente en un par trenzado, el transceptor convierte los niveles del controlador CAN en tensiones diferenciales. Un tranceptor CAN típico para par trenzado mantendrá las líneas CAN_L (CAN Low) y CAN_H (CAN High) a 2.5 voltios en el estado recesivo. Para señalizar el estado dominante, pondrá CAN_H a 3.5 voltios y CAN_L a 1.5 voltios.

Cuando dos o más dispositivos CAN intentan transmitir al mismo tiempo, la colisión será detectada y resuelta utilizando acceso múltiple por detección de portador con arbitraje basado en la prioridad del mensaje (CSMA/AMP, *Carrier Sense Multiple Access with Arbitration on Message Priority*). Esto significa que el mensaje de más alta prioridad será transmitido, mientras que se detendrán los demás. La prioridad de un mensaje viene dada por su identificador, teniendo los números más bajos las prioridades más altas. Cada nodo comenzará a transmitir su identificador en el bus comenzando por el más significativo; al mismo tiempo, monitorizará el bus para comprobar que ha sido colocado en

el valor adecuado. Si un dispositivo intenta escribir un 1 lógico (recesivo) mientras que otro intenta escribir un 0 lógico (dominante), el estado dominante consigue la prioridad. Los dispositivos que han escrito un 1 lógico, abandonarán la transmisión hasta que el bus pase a estar desocupado nuevamente. Este proceso se remite hasta que en el bus sólo que el mensaje con el identificador más bajo (más alta prioridad).

Por ejemplo, sean los siguientes dispositivos que intentan enviar un mensaje al mismo tiempo:

- dispositivo 1 - dirección 432 (00110110000)
- dispositivo 2 - dirección 155 (00010011011)
- dispositivo 3 - dirección 185 (00010111001)

Los tres dispositivos cambiarán el bus a estado dominante para el comienzo de la trama (SOF, *Start-Of-Frame*) y los dos primeros bits de cada identificador. Cada dispositivo monitorizará el bus y determinarán que ha habido éxito. Al llegar al tercer bit, el dispositivo 1 detectará que el bus sigue en estado dominante cuando estaba intentando que fuese recesivo, así que detectará la colisión y se retirará. Los otros dos dispositivos continuarán hasta llegar al sexto bit, momento en el que el dispositivo 3 detectará la colisión y también se retirará. De este modo sólo el dispositivo con el identificador más bajo continuará transmitiendo.

Los dispositivos CAN incorporan además múltiples esquemas de detección de errores. Cada mensaje contiene un CRC para verificar la integridad de los datos recibidos. Cuando un receptor determina un error como resultado de examinar este CRC, no mandará la trama ACK (acuse de recibo). Si ningún dispositivo lo hace, el transmisor sabrá que se produjo algún error. Por otro lado, una vez transmitido el campo de arbitraje (SOF + identificador), también se monitorizarán determinados bits para comprobar que el bus no está a un estado diferente del que se desea. Si se detectase el bit incorrecto, se generaría una trama de error y se volvería a intentar transmitir de nuevo la trama.

Cada nodo de una red CAN tiene su propio reloj, el cual no se envía durante la transmisión. La sincronización se lleva a cabo dividiendo cada bit en varios segmentos: *sincronización*, *propagación*, *fase 1* y *fase 2*. La longitud de cada fase se puede ajustar según las condiciones de la red y del nodo. Puesto que se utiliza un código NRZ (*Non Return to Zero*), cada 5 bits consecutivos con un mismo valor, se intercala el bit opuesto. Así se consigue que haya suficientes transiciones para mantener el sincronismo.

CAN contempla cuatro tipos de tramas:

- Trama de datos (*data frame*): es una trama que contiene los datos transmitidos por un nodo.
- Trama remota (*remote frame*): es una trama que solicita la transmisión de un determinado identificador.
- Trama de error (*error frame*): se transmite por cualquier nodo al detectar un error.
- Trama de sobrecarga (*overload frame*): se utiliza para provocar un retraso entre datos y/o tramas remotas.

En la Tabla 2.2 se pueden observar los distintos campos de una trama de datos (formato base). El bit de acuse de recibo debe ser recesivo (1) para el transmisor, y dominante (0) cuando se usa por el receptor. En caso de que el bit RTR fuese 1, la trama de datos pasaría a ser una trama remota, desapareciendo también el campo de datos. En general, la transmisión se inicia de forma autónoma por la propia fuente de datos (por ejemplo un sensor), pero es posible que cualquier dispositivo los

solicite a través de esta trama remota. La trama de datos extendida (identificadores de 29 bits) se basa en la trama base, haciendo uso de los bits IDE y r0 para diferenciarse de ésta y añadiendo los 18 bits adicionales para el identificador y otro campo con un par de bits reservados. Los detalles de las tramas de error y sobrecarga se pueden encontrar en la especificación CAN de Bosch.

Campo	Longitud (bits)	Función
Start-of-frame	1	Indica el comienzo de una trama
Identifier	11	Identificador único para los datos
Remote transmission request (RTR)	1	Dominante (0)
Identifier extension bit (IDE)	1	Debe ser dominante (0)
Reserved bit (r0)	1	Bit reservado (debe ser dominante)
Data length code (DLC)	4	Número de bytes de datos (0-8 bytes)
Data field	0-8 bytes	Datos a ser transmitidos
CRC	15	Código de redundancia cíclica
CRC delimiter	1	Debe ser recesivo (1)
ACK *slot*	1	Bit de acuse de recibo
ACK delimiter	1	Debe ser recesivo (1)
End-of-frame (EOF)	7	Todos deben ser recesivos (1)

Tabla 2.2: Trama de datos CAN

La especificación CAN de Bosch no incluye detalles sobre la capa física. Es posible implementar el protocolo CAN en distintos medios tales como par trenzado o fibra óptica. Por el contrario, la especificación ISO proporciona los detalles para implementar CAN en par trenzado.

El número de dispositivos que se pueden conectar a una red CAN es en teoría ilimitado (el identificador hace referencia a un tipo de datos, no a un dispositivo), por tanto diversos dispositivos pueden utilizar el mismo identificador. La longitud del bus depende de la tasa de transmisión y del medio; 1 Mbps está garantizado si se utiliza par trenzado y menos de 40 metros de longitud. Algunas longitudes máximas típicas del bus CAN se pueden ver en la Tabla 2.3.

Máxima longitud	Tasa de transmisión
1 Mbps	40 m
500 kbps	100 m
250 kbps	200 m
125 kbps	500 m
10 kbps	6 km

Tabla 2.3: Longitud del bus CAN

La norma ISO también especifica el uso de impedancias de 120 ohm en las terminaciones del bus para evitar reflexiones que podrían incrementar la tasa de errores, pero éstas son innecesarias a bajas tasas de transmisión (menores de 125 kbps).

Ya que CAN no contempla capas superiores del modelo OSI, carece de características tales como control de flujo, direccionamiento de dispositivos o transporte de bloques de datos mayores que un mensaje. Diversas implementaciones de protocolos para estas capas superiores se han realizado. Algunas de ellas son las siguientes:

- Smart Distributed System (SDS): desarrollado por Honeywell para sistemas de automatización industrial.

- DeviceNet: desarrollado por Allan Bradley para dispositivos de control industrial. Define una capa de aplicación para cubrir un amplio rango de perfiles. Sus aplicaciones típicas incluyen el intercambio de información, sistemas de seguridad o grandes redes de control.
- CAN Application Layer (CAL): es una capa de aplicación desarrollada por el grupo de usuarios CAN-in-Automation.
- CANOpen: implementa todas los niveles del modelo OSI por encima del nivel de red (incluido éste). Consiste en un esquema de direccionamiento, varios protocolos de comunicación y una capa de aplicación definida por un perfil de dispositivo. Los protocolos de comunicación tienen soporte de gestión de red, monitorización de dispositivos y comunicación entre nodos, incluyendo un protocolo de transporte simple. Los niveles físico y de enlace son CAN, aunque se han implementado dispositivos soportando otros medios de comunicación. Cada dispositivo CANopen contiene un diccionario de objetos que se puede usar para configurar el dispositivo, recordando en cierta forma a los TEDS de IEEE 1451.
- J1939: usado para comunicación y diagnósticos entre los diferentes componentes de un vehículo.

2.3 Problemas de CAN

Actualmente CAN es la tecnología dominante en el mercado de las redes automovilísticas, pero no es la más adecuada para los avances más recientes, como pueden ser los sistemas *x-by-wire*. En este caso un estricto comportamiento determinista es necesario. Además, un nivel más alto de seguridad es necesario.

Por un lado, se puede afirmar que CAN es un protocolo determinista. Esto es sólo cierto para los mensajes de más alta prioridad, debido a la forma no destructiva en la que se resuelven las colisiones. Por otro lado, para los demás mensajes, ya que a los distintos nodos se les permite generarlos por sí mismos, no se puede saber de antemano el momento exacto en el que serán enviados, puesto que no es posible predecir el número de colisiones que sufrirán con otros de mayor prioridad. Este comportamiento puede conducir a *jitters* potencialmente peligrosos que podrían empeorar determinados algoritmos de control y empeorar su precisión. Incluso podría ocurrir que algunos mensajes llegasen fuera del límite de tiempo.

Los tiempos de respuesta en una red CAN se pueden evaluar en caso de que se conozca el periodo de los datos intercambiados cíclicamente y el intervalo mínimo entre llegadas (MIT, *Minimum Inter-arrival Time*) para los datos que se generen sin un patrón temporal determinado. En [3] se presenta una técnica para calcular las latencias de transmisión en los peores casos, lo que permite asegurar durante la fase de diseño que no se excederán determinados límites de tiempo en la entrega de mensajes. En cualquier caso, no se pueden prevenir *jitters*.

Es importante destacar que esta técnica sólo obtiene resultados significativos cuando no hay erores de transmisión en el bus (debido a la retransmisión automática de tramas en CAN, lo que podría incrementar la carga del bus de forma impredecible). Para disminuir este problema algunos controladores modernos proporcionan modos de transmisión de un sólo envío (no hay retransmisión en caso de error o pérdida del bus por colisión con mensajes de mayor prioridad). Esto supone una solución para los casos en que los datos se actualicen a una tasa adecuado (la pérdida de una muestra en un bucle de control no supone un problema habitualmente).

Otro problema que puede afectar fuertemente al determinismo es que bajo determinados casos de errores, un mensaje podría llegar sólo a parte de los dispositivos de la red, generando inconsistencias

en la visión que cada nodo tiene del sistema.

Todos estos aspectos dificultan la integración de subsistemas de diferentes fabricantes, puesto que puede que al integrarse dejen de cumplirse sus requisitos de tiempos, aunque por separado funcionen correctamente.

Además de los relacionados con el determinismo, otro de los inconvenientes que presentan los sistemas CAN es que cualquier nodo defectuoso podría mandar repetidamente mensajes de la más alta prioridad y bloquear toda la red. En los sistemas time-triggered este problema tiene una solución inmediata con la utilización de guardianes de bus, lo cual no resulta tan inmediato en CAN. En cualquier caso, en algunos trabajos se ha abordado el problema de usar BGs con controladores CAN estándar. Concretamente, se han introducido tipos especiales de BGs que pueden también ser utilizados en sistemas *event-triggered*.

Además de la detección de errores a través del CRC, en sistemas de seguridad crítica se puede conseguir detección de errores en el dominio del tiempo cuando se conoce previamente las acciones que deben ocurrir en cada instante de tiempo. En este contexto, incluso la presencia (o ausencia) de un mensaje aporta información valiosa. Si un nodo manda mensajes con el identificador incorrecto (tomando por tanto el papel de otro), puede ser detectado al compararlo con el tiempo en el que se espera su llegada en los nodos receptores (los BGs impiden además que se envíe en otro).

Otra buena idea para estos sistemas críticos es añadir redundancia espacial (topología con un doble canal) y redundancia temporal (envío por duplicado de mensajes críticos en un mismo ciclo). CAN puede soportar redundancia temporal y ha sido usado en arquitecturas redundantes, aunque no proporciona un soporte nativo para redundancia espacial. Incluso se podría pensar en una topología en estrella en lugar del bus convencional; pero no hay productos comerciales que soporten esta característica, aunque sí estudios sobre el tema.

Por último, como ya se ha comentado, la tasa de transmisión máxima de CAN es de 1 Mbps para un bus de 40 m. Sorprendentemente, aunque pueda parecer una tasa suficiente, supone una limitación para los fabricantes de coches actuales. Si todos los valores muestreados por casa sensor se hacen accesibles a todos los bucles de control, el rendimiento dinámico del vehículo puede mejorar notablemente. Por ejemplo, se puede demostrar que los sistemas ABS proporcionan mejores rendimientos si también disponen de los valores de la aceleración vertical (tradicionalmente necesitado sólo por el sistema de control de suspensión). Este requerimiento conduce a una alta carga en la red que no puede ser satisfecha por las redes CAN tradicionales. Depender de un conjunto de redes CAN conectadas por pasarelas no es una solución óptima, tanto por el coste como por los retrasos que introduce. La tendencia actual es reducir el número de redes utilizadas en un vehículo, aunque no es generalmente posible depender sólo de una, especialmente en vehículos de alta gama.

Este inconveniente no puede ser solucionado por medio de avances tecnológicos sólamente. Para que el mecanismo de detección de colisiones funcione correctamente es necesario que todos los nodos en el bus vean el mismo nivel al mismo tiempo. Esto implica que el tiempo que toma la señal para viajar de un nodo a otro debe ser menor que el segmento de propagación del bit (esto es, una fracción del tiempo de bit). Puesto que el el mayor retraso de propagación se da cuando dos nodos se localizan en los dos extremos del bus, en la practica la máxima tasa de datos alcanzable es inversamente proporcional a la longitud del bus.

2.4 Local Interconnect Network (LIN)

LIN es un estándar [4] de hecho para la comunicación se sensores inteligentes y actuadores en vehículos. Se utiliza como una solución económica cuando no se necesita ni la versatilidad ni el ancho de

banda de un bus CAN. La primera versión de la especificación se introdujo en 1999 como LIN 1.1. Desde entonces ha evolucionado hasta la versión 2.0 de septiembre de 2003 para expandir las capacidades de configuración y diagnóstico.

El sistema se compone de un maestro y varios esclavos. El maestro conoce el orden temporal de todos los datos a transmitir, los cuales serán transmitidos por el nodo correspondiente cuando les sea solicitado. La solicitud se hace a través de un mensaje determinado en el que aparece el identificador del mensaje solicitado.

El maestro utiliza una o más tablas de planificación para comenzar las transmisiones en el bus LIN. Estas tablas contienen al menos el momento en que el se debe iniciar el envío de mensajes. Una trama LIN consiste en una cabecera y una respuesta. La cabecera es siempre enviada por el maestro, mientras que la respuesta se envía sólo por un esclavo. En conjunto, la trama incluye un campo de sincronización, una instrucción (como identificador), una respuesta predeterminada de 2, 4 y 8 bytes (como campo de datos) y un código corrector de errores (checksum).

La especificación LIN permite el uso de nodos de muy bajo coste en la red. Se trata de una red de un solo hilo basada en ISO 9141. Actualmente se usa a través de microcontroladores con soporte UART o hardware dedicado LIN. El microcontrolador genera todos los datos necesarios por software y se conecta a la red a través de un transceptor LIN. La tasa de transmisión más alta es de 20kbps, aunque se recomienda que sea de 2400bps, 9600bps o 19200bps.

2.5 J1850

En EEUU ha sido adoptado como estándar el bus J1850 SAE, parecido a CAN en cuanto al campo de aplicación (la automoción). El J1850 permite el uso de uno o dos hilos para el bus, dos velocidades de transmisión (10.4 kbps o 41.7 kbps), dos técnicas de codificación de bit (modulación por ancho de pulso PWM o modulación variable del ancho de pulso VPW) y detección de errores por CRC o checksum dependiendo del formato del mensaje y de la técnica de modulación seleccionada.

2.6 Time-Triggered Protocol

Este protocolo [5], que se definió en 1994, se basa en TDMA para asegurar el determinismo. Además, por estar basado en TDMA, se pueden utilizar sin problemas guardianes de bus para incrementar la seguridad, aislando unos *slots* de tiempo de otros.

Para facilitar la introducción, se definieron dos versiones distintas: TTP/A y TTP/C. TTP/A se basa en un esquema maestro/esclavo que permite implementaciones menos costosas. TTP/C utiliza un mecanismo totalmente distribuido, alcanzando una gran tolerancia a fallos. En el el pasado, TTP/C no tuvo demasiado éxito comparado con CAN en la industria automovilística; sin embargo, ha estado ganando aceptación en los últimos años.

TTP/C soporta canales duplicados e incorpora sistemas de gestión de redundacia. Al contrario que en CAN, la técnica de acceso al medio no impone ninguna restricción, soportándose tasas de hasta 25 Mbps por canal. La transmisión de datos se organiza como un conjunto de series TDMA (*rounds*), las cuales tienen todas la misma duración. Un conjunto de series TDMA se repite periodicamente en toda la red, lo que se conoce como ciclo de grupo (*cluster cycle*). Cada serie TDMA se compone de una secuencia de *slots* de longitud fija. La única diferencia posible entre dos series TDMA es el contenido de estos *slots*, ya que su tamaño y su nodo asociado ha de mantenerse siempre. En el caso de que varios nodos compartan un mismo *slot* se habla de *slot* multiplexado. Cada nodo podrá transmitir en

el nodo multiplexado sólo en alguna de las series TDMA, lo que permite reducir el ancho de banda de las señales más lentas.

La estructura de las series TDMA se define a través de una lista de descriptores de mensaje (MEDL, *MEssage Descriptor List*). Esta lista se define estáticamente y se carga en cada nodo antes de que la red comience a funcionar. A cada *slot* en el ciclo de grupo se le asigna su propia entrada en la MEDL, que también especifica si ese *slot* contiene información relevante para el nodo. Además especifica el papel que juega dicho nodo respecto a dicha información (lectura/escritura).

La MEDL se usa en conjunto con el concepto de un reloj mantenido globalmente para coordinar la transmisión de mensajes y verificar la recepción de otros. Se define un algoritmo distribuido para mantener un reloj global en toda la red tolerante a fallos con una precisión en el rango de los microsegundos. Este reloj se pasa a los diferentes microprocesadores de cada nodo a través del controlador TTP, lo que permite operaciones altamente sincronizadas. En este caso se suele hablar de una arquitectura *time-triggered* (TTA, *Time-Triggered Architecture*), ya que el protocolo no sólo afecta a las comunicaciones, sino también a la forma en que se generan los datos en cada nodo.

En la Figura 2.2 se puede ver un ejemplo de ciclo de grupo con las diferentes series TDMA. Se han detallado los mensajes de uno de los *slots*, pudiéndose enviar el mismo mensaje en ambos canales para conseguir redundancia de datos.

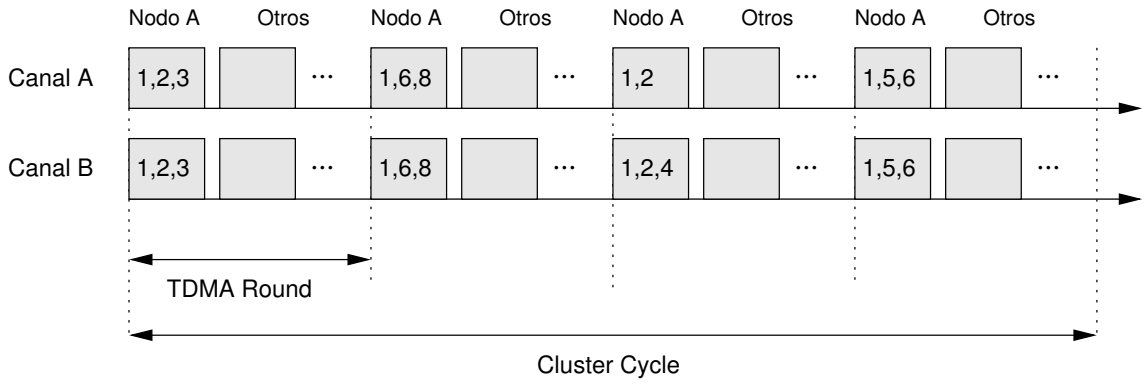

Figura 2.2: Comunicación en un sistema TTP/C

2.7 Time-Triggered CAN

TTCAN (*Time-Triggered CAN*) [6] fue introducido por Bosch en 1999 en un intento de hacer CAN adecuado para las nuevas necesidades de la industria automovilística. La especificación es actualmente estable y ha sido estandarizada por la ISO. La razón que condujo al desarrollo de TTCAN fue ofrecer un mayor grado de determinismo manteniendo la compatibilidad con los dispositivos CAN existentes.

El protocolo TTCAN se sitúa por encima del protocolo CAN, permitiendo operaciones de tipo time-triggered en una red CAN quasi-convencional. Por tanto, no es posible alcanzar tasas de transmisión mayores de 1 Mbps.

TTCAN es capaz de asegurar comunicaciones estrictamente deterministas, por lo que es adecuado para los sistemas de primera generación *drive-by-wire* (aquellos con respaldos hidráulicos o mecánicos). Sin embargo, debido a su limitado rendimiento, probablemente no será adecuado para la próxima generación *steer-by-wire*. En estos casos, las tasas de transmisión requeridas son notablemente más altas.

El protocolo se basa en un enfoque centralizado, donde un nodo especial denominado time master (TM) mantiene a toda la red sincronizada mandando periódicamente un mensaje de referencia (RM, *Reference Message*), el cual se implemente como un mensaje CAN de muy alta prioridad. La transmisión de mensajes se organiza como una secuencia repetida de ciclos básicos (BC, *Basic Cycles*). Cada ciclo básico se compone de un número de ventanas de tiempo, que pueden ser de estos 4 tipos:

- Mensaje de referencia (RM, *Reference Message*): cada ciclo básico comienza con un RM. Cuando reciben un RM, cada nodo reinicia su temporizador de forma que se mantenga todo el sincronismo en la red.

- Ventana exclusiva (*exclusive window*): cada ventana exclusiva está estáticamente reservada para un mensaje predefinido, de forma que no puedan ocurrir colisiones. Son usadas para para datos críticos que deben llegar de un modo determinista y sin ningún tipo de *jitter*.

- Ventana de arbitraje: estas ventanas no están preasignadas a un determinado mensaje, de forma que diferentes mensajes competirán por acceder al bus basándose en el sistema de arbitraje CAN para resolver las colisiones. En este caso, también se mantiene la prioridad de mensajes CAN y puede que un mensaje incluso sea enviado en el siguiente BC.

- Ventana libre: son usadas para futuras expansiones de los sistemas TTCAN.

Para que no se excedan los límites de las diferentes ventanas, los controladores TTCAN deben ser capaces de desactivar la retransmisión automática en caso de que se detecte algún error o se pierda el acceso al medio por colisión. La única excepción ocurre cuando existen varias ventajas de arbitraje consecutivas.

En caso de que varios nodos intenten transmitir dentro de una ventana exclusiva por falta de sincronización, el esquema de arbitraje CAN podría resolver la situación. Esto convierte a TTCAN en un protocolo especialmente robusto.

Para incrementar la flexibilidad, se permite que no todos los ciclos básicos sean iguales. En su lugar es posible la definición de una matriz del sistema (SM, *System Matrix*) que puede contener hasta 64 ciclos básicos diferentes que se repiten periódicamente. De esta forma, el periodo efectivo de una red TTCAN viene dato por esta matriz (*ciclo de matriz*). Los diferentes BCs se distinguen por el denominado contador de ciclos (*cycle counter*), que se incluye en el primer byte de cada RM. Comienza en 0 y se repite hasta el valor máximo, después de lo cual vuelve a reiniciarse. En la Figura 2.3 se puede ver una matriz con 4 BCs.

Es importante destacar que cada BC debe estar compuesto de la misma secuencia de ventanas, esto es, ventanas con la misma duración. No obstante, la misma ventana en diferentes BCs pueden ser utilizadas para diferentes mensajes, de forma que es posible tener datos en ventanas exclusivas que se repiten sólo cada varios BCs. También es posible tener varias ventanas consecutivas del mismo tipo (unidas) o ventanas destinadas al mismo mensaje dentro del mismo BC.

Otro detalle es que en cada controlador de una red TTCAN se definen una serie de marcas de tiempo como *triggers* de transmisión o recepción, que son usadas para enviar mensajes y validar la recepción de otros. Cada nodo no conoce los detalles de todos los mensajes, sólo de aquellos que el nodo debe enviar o recibir.

Una de las principales ventajas de TTCAN es que permite la coexistencia de mensajes *event-triggered* y *time-triggered* dentro de una misma red. Además, requiere pequeños cambios a los chips CAN comunes.

Figura 2.3: Matriz del sistema TTCAN

2.8 Byteflight

El protocolo Byteflight [7] fue introducido por BMW y su desarrollo comenzó en 1996. Existen controladores de comunicación que cumplen con su especificación y ha sido usado en varios vehículos. Su tasa de transmisión es de 10 Mbps, un orden de magnitud por encima de CAN. Su introducción se debió a las nuevas necesidades del mercado automovilístico que difícilmente puede satisfacer CAN. Además se concibió de forma que pudiese proporcionar un alto grado de flexibilidad.

Byteflight permite que los nodos produzcan y envíen mensajes de forma asíncrona. La técnica de acceso asegurará que los mensajes de más alta prioridad se envíen primero, mientras que los de más baja prioridad deben esperar. El objetivo es proporcionar las ventajas de los dos esquemas: síncrono y asíncrono. En particular, se aseguran *jitters* muy bajos para los mensajes de alta prioridad y la asignación de ancho de banda para los de baja prioridad es flexible. Esta era la cuestión clave para los diseñadores del protocolo, ya que el tiempo de desarrollo para los sitemas de control en automóviles se acorta progresivamente y es necesario hacer cambios rápidamente de forma segura.

Byteflight hace uso del llamado acceso múltiple por división flexible en el tiempo (FTDMA, *Flexible Time Division Multiple Access*). Como se puede ver en la Figura 2.4, la transmisión de los datos se organiza en ciclos. Cada ciclo consta de un pulso de sincronización (*SYNC*) que es mandado periódicamente por un nodo especial conocido como *SYNC master*. Estos pulsos SYNC se usan para mantener una base común de tiempo para todos los nodos de la red.

El acceso al bus se basa en un identificador de mensaje, que sirve también para indicar la prioridad de éste. Al igual que en CAN, no puede haber dos nodos que produzcan mensajes marcados con el mismo identificador. Cada nodo mantiene un contador de *slots* que se reinicia tras cada pulso SYNC. El contador comienza en 0 y se incrementa hasta el último identificador permitido en la red (255), excepto si llega un nuevo pulso SYNC. Cuando el contador de *slots* llega al valor correspondiente al identificador del mensaje esperando a ser enviado, comienza la transmisión.

En la práctica, cada vez que el bus está desocupado por un cierto tiempo (bastante más pequeño

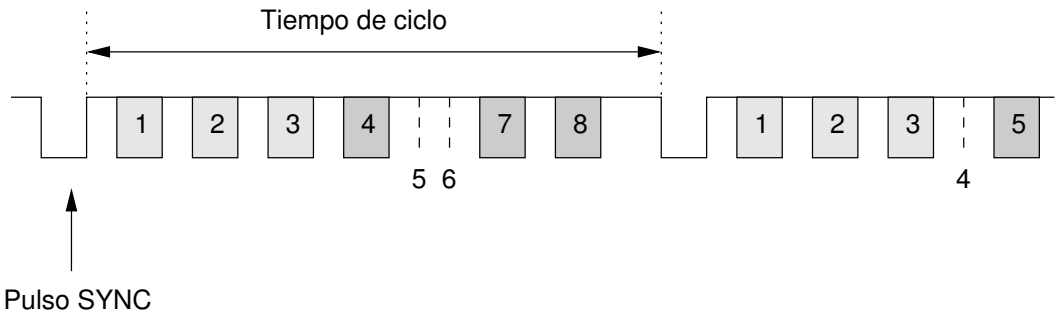

Figura 2.4: Técnica FTDMA en Byteflight

que el tiempo de transmisión de un mensaje), se incrementa este contador de *slots*. Cuando hay un mensaje en el bus, este contador se detiene hasta que termina la transmisión. Se puede decir que cada nodo tiene una oportunidad para transmitir (empezando por los de identificador más bajo) dentro de cada ciclo. Si no lo hace pasado un pequeño periodo de tiempo (denominado *mini*slot) los demás nodos entenderán que no lo va a hacer en ese ciclo y comprobarán si ha llegado su turno.

Como en CAN, se asegura la prioridad de los mensajes con los identificadores más bajos. No obstante, cada mensaje sólo puede ser enviado una vez por ciclo como máximo. Esto limita en cierta forma el problema de nodos defectuosos que podrían bloquear el bus.

Otra ventaja de FTDMA sobre CAN es que los mensajes de alta prioridad apenas sufren *jitter* si son enviados en todos los ciclos. Sin embargo, en el caso de que falten algunos mensajes (por fallor temporal), los siguientes son transmitidos un poco antes. Desde este punto de vista no es tan bueno como los sistemas *time-triggered*.

Un problema de los sistemas Byteflight es que la operación correcta de la red depende del correcto funcionamiento de un nodo específico (*SYNC master*), lo que se puede convertir en un punto de fallo para todo el sistema. En cualquier caso, se pueden utilizar nodos de respaldo.

2.9 FlexRay

El consorcio FlexRay nació en el año 2000 e incluye a muchos de los mayores fabricantes de automóviles. Este protocolo [8] cuenta con las ventajas de protocolos *time-triggered*, como TTP/C, y de soluciones asíncronas, como Byteflight. Concretamente, se diseñó teniendo en cuenta la compatibilidad con dispositivos y redes Byteflight (del que FlexRay se puede considerar una evolución).

Como ya se ha comentado, el principal problema con Byteflight está relacionado con la seguridad: aunque los mensajes de alta prioridad se intercambian siguiendo un esquema que asegura bajos *jitters*, los nodos pueden acceder a la red de forma asíncrona, de forma que no se puede utilizar ningún guardián de bus. FlexRay corrige este problema y permite el uso de guardianes. Además proporciona soporte nativo para medios de transmisión por duplicado (tanto en topologías en estrella como en bus). Se pueden alcanzar hasta 10 Mbps por canal.

Las comunicaciones en FlexRay están estructuradas en en ciclos de transmisión que se repiten periodicamentes, tal como se ha representado en la Figura 2.5. Cada ciclo de transmisión se divide en cuatro partes separadas:

- Segmento estático (*static segment*): se gestiona según una arquitectura *time-triggered*. En este caso, el momento de enviar un determinado mensaje se conoce de antemano. Esto es posible porque en FlexRay todos los *slots* para los mensajes síncronos tienen el mismo tamaño y se

asignan en orden ascendente de identificador. Como en cualquier sistema *time-triggered*, se pueden aislar a los nodos transmisores del resto a través de un guardián de bus.

- Segmento dinámico (*dynamic segment*): este segmento utiliza la técnica FTDMA de Byteflight que ya se ha comentado. Cada nodo hace uso de un contador que se incrementa cada cierto tiempo, para comenzar a transmitir cuando le llegue su turno. Si la red está ocupada, los nodos esperarán para incrementar el contador a la finalización de la transmisión en curso. Este segmento dinámico se aísla del segmento estático a través de guardianes de bus, de forma que no puedan interferir con mensajes que requieren una alta seguridad.
- Ventana de símbolo (*symbol window*): se introdujo para evaluaciones del sistema.
- Tiempo de parada en la red (*network idle time*)

Todos los segmentos excepto el estático pueden omitirse, pudiéndose tener configuraciones estáticas o estáticas y dinámicas en la misma red.

Cuando se activa la red, todos los nodos deben estar sincronizados antes de comenzar a operar. El procedimiento de arranque utiliza un esquema completamente distribuido y puede estar operativo tan pronto como dos nodos estén listos para comunicarse a través de la red. A partir de ese momento, otros controladores pueden incorporarse a la red sin causar ningún tipo de problema.

En el trascurso del intercambio de mensajes, cada nodo usa alguna de las tramas intercambiadas en el segmento estático para conseguir la sincronización necesaria (mantener un reloj global común). Este reloj se especifica a través de un contador de ciclo y un tiempo de ciclo. El contrador de ciclo es un entero de 6 bits que se incrementa por todos los nodos después de cada ciclo, pudiéndose usar para multiplexar *slots*. El tiempo de ciclo se pone a 0 al comienzo de cada ciclo de transmisión y se incrementa con el paso del tiempo. Se usa para decidir cuándo se debe enviar un mensaje síncrono.

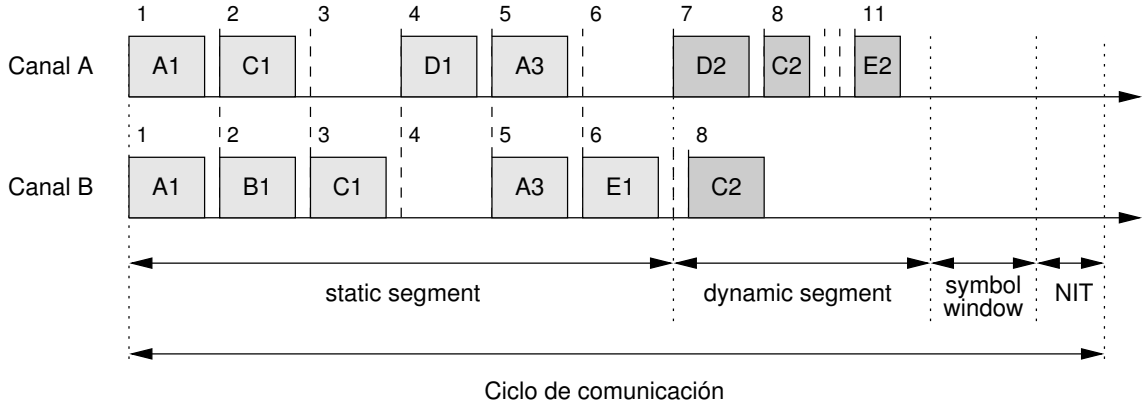

Figura 2.5: Ciclo de comunicación en un sistema FlexRay redundante

Referencias

[1] E.A. Bretz. By-wire cars turn the corner. *Spectrum, IEEE*, 38(4):68--73, Apr 2001.

[2] International Organization for Standardization. Road vehicles—Controller area network (CAN)—Part 1: Data link layer and physical signalling. *ISO IS 11898-1*, 2003.

[3] K. Tindell, A. Burns, and A. J. Wellings. Calculating controller area network (can) message response times. *Control Engineering Practice*, 3(8):1163 -- 1169, 1995.

[4] LIN Consortium. Local Interconnect Network-LIN Specification Package, Revision 2.0. September 2003.

[5] TTTech. Time-Triggered Protocol TTP/C High-Level Specification Document, Protocol Version 1.1, Specification edition 1.4.3. November 2003.

[6] International Organization for Standardization. Road vehicles—Controller area network (CAN)—Part 4: Time-triggered communication. *ISO IS 11898-4, 2004*, 2004.

[7] R. Grießbach M. Peller, J. Berwanger. Byteflight specification, draft, Version 0.5, BMW AG. October 1999.

[8] FlexRay Consortium. FlexRay Communications System—Protocol Specification, Version 2.0. June 2004.