Resumo das idéias do livro

**Capítulo 2 – Protocolo de barramento**

Cada fabricante de veículo escolhem protocolos de barramento que mais fazem sentido para seus automóveis. O protocolo conhecido como barramento CAN (CAN bus) está disponível no conector OBD-II. Entretanto os pacotes que trafegam por meio do barramento CAN em um automóvel não são padronizados.

A comunicação crítica (gestão de RPM e freios) ocorre nas linhas de barramento de alta velocidade, enquanto a comunicação não crítica (controle das travas da porta) ocorre nas linhas de barramento de média ou baixa velocidade.

CAN é um simples protocolo utilizado na fabricação e na indústria automobilística. Veículos modernos integram sistemas embutidos e unidades de controle eletrônico (ECUs) que se comunicam através do protocolo CAN. O protocolo CAN tem sido um padrão nos Estados Unidos para carros e caminhões leves desde 1996, mas não foi tornada obrigatória até 2008 (2001 para veículos europeus). Alguns carros que foram fabricados antes de 1996 pode ter implantado o protocolo CAN, mas é necessário verificar antes.

O CAN roda em dois fios: CAN high (CANH) and CAN low (CANL). O protocolo CAN utiliza diferentes sinais (com exceção da CAN de baixa velocidade – ‘low-speed’ – discutida em “The GMLAN Bus” na página 20), o que significa que ao entrar um sinal, o CAN eleva a tensão em uma linha e solta a outra em uma quantidade igual. Sinais diferentes é utilizado em ambientes que devem ser tolerantes a falhas, como em sistemas automotivos e de fabricação.

Quando é transmitido um bit no barramento CAN, o sinal transmite simultaneamente tanto 1V mais alto, quanto mais baixo. Os sensores e a ECU tem um transceptor que checa para garantir que ambos os sinais serão disparados; se não forem, o transceptor rejeita o pacote como ruído.

Os dois fios de par trançado compõem o barramento e exigem que o barramento seja terminado em cada extremidade. Existe um resistor de 120 ohms em ambos os fios em suas extremidades de terminação. Se o módulo não está no final do barramento, ele não precisa se preocupar com a terminação. (As someone who may tap into the lines, the only time you’ll need to worry about termination is if you remove a terminating device in order to sniff the wires.)-> Como alguem que pode desconectar os fios, o unico momento que você precisa se preocupar com o final é se você remover o terminating device para monitorar os fios.

**O conector OBDII**

Diversos veículos possuem um conector OBD-II, também conhecido como o conector de link de diagnóstico (Diagnostic Link Conector – DLC), que se comunica com a rede interna do veículo.

**Encontrando conexões CAN**

Os fios CAN passam pelo veículo e se conectam entre os ECUs e outros sensores, e eles estão sempre em pares de fios duplos. As conexões CANH e CANL são encontrados nos pinos 6 e 14 do conector OBD-II, respectivamente. Alguns carros utilizam a CAN para média velocidade (mid-speed MS-CAN) e baixa velocidade (low-speed LS-CAN), entretanto, muitos veículos utilizam protocolos diferentes para essas comunicações. Nem todos os barramentos estão expostos por meio do conector OBD-II.

**Layout do pacote do barramento CAN**

Há dois tipos de pacotes CAN: padrão e estendido. Pacotes estendidos são como os padrões, mas com um espaço maior para armazenar IDs.

**Pacotes Padrão**

Cada pacote do barramento CAN contém quatro elementos-chave:

**ID de Arbitragem (Arbitration ID):** O ID de arbitragem é uma mensagem de broadcast que identifica o ID do dispositivo que tenta comunicar, embora qualquer dispositivo possa enviar múltiplos IDs de arbitragem. Caso dois pacotes CAN sejam enviados ao longo do barramento ao mesmo tempo, aquele com o ID de arbitragem inferior ganha.

**Extensão do Identificador (IDE) - Identifier extension (IDE):** Este bit é sempre 0 para CAN padrão

**Código do tamanho dos dados (DLC) - Data length code (DLC):** Este é o tamanho dos dados, que variam de 0 a 8 bytes.

**Dados (Data):** Este é o próprio dado. O tamanho máximo dos dados transportados por um pacote de barramento CAN padrão pode ser de até 8 bytes, mas alguns sistemas forçam 8 bytes preenchendo o pacote.

Como os pacotes do barramento CAN estão em broadcast, todos os controladores da mesma rede veem todos os pacotes, tipo de UDP em redes Ethernet. Os pacotes não carregam informações sobre qual controlador (ou atacante) enviou o quê. Porque qualquer dispositivo pode ver e transmitir pacotes, é trivial para qualquer dispositivo no barramento que simule qualquer outro dispositivo.

**Pacotes Extendidos (Extended Packets):**

Pacotes estendidos são como padrões, exceto que eles podem ser encadeados juntos para criar mais IDs. Pacotes estendidos são projetados para caber dentro da formatação CAN padrão, a fim de manter a compatibilidade com versões anteriores. Portanto, se um sensor não tem suporte para pacotes estendidos, ele não irá falhar se (it won’t break if) outro pacote transmite pacotes CAN estendidos na mesma rede.

Os pacotes padrão também se diferem dos estendidos no uso de sinalizadores. Ao examinar os pacotes estendidos em um descarregamento da rede (network dump), você verá que, ao contrário de pacotes padrão, os pacotes estendidos usam substituição de solicitação remota (substitute remote request – SRR) no lugar da solicitação de transmissão remota (remote transmission request – RTR) com SSR definido como 1. Eles também terão o IDE definido como 1, e seus pacotes terão um identificador de 18 bits, que é a segunda parte do identificador padrão de 11 bits. Existem protocolos adicionais do tipo CAN que são específicos para alguns fabricantes e também são compatíveis com CAN padrão de forma muito semelhante à CAN extendida.

**O Protocolo ISO-TP (ISO-TP Protocol)**

A ISO 15765-2, também conhecida como ISO-TP, é um padrão para o envio de pacotes pelo barramento CAN (CAN bus) que estende o limite CAN de 8 bytes para suportar até 4095 bytes encadeando pacotes CAN juntos. O uso mais comum de ISO-TP é para diagnósticos (consulte “Serviços de Diagnóstico Unificado – Unified Diagnostic Services” na página 54) e mensagens KWP (um protocolo alternativo para CAN), mas também pode ser usado sempre que grandes quantidades de dados precisam ser transferidos por CAN. O programa CAN-utils inclui isotptun, uma ferramenta que encapsula uma prova de conceito (proof-of-concept) para SocketCAN que permite que dois dispositivos emcapsulem IP através do CAN (that allows two devices to tunnel IP over CAN). (Para obter uma explicação mais detalhada sobre como instalar e usar o can-utils, consulte “Configurando o can-utils para conectar-se a dispositivos CAN – Setting Up can-utils to Connect to CAN Devices” na página 36.).

Para encapsular ISO-TP em CAN, o primeiro byte é usado para endereçamento estendido, deixando apenas 7 bytes para dados por pacote. Enviar muita informação sobre ISO-TP pode facilmente inundar o barramento, portanto tenha cuidado ao usar este padrão para grandes transferências em um barramento ativo.

**O Protocolo CANopen (CANopen Protocol)**

Outro exemplo de extensão do protocolo CAN é o protocolo CANopen. CANopen divide o identificador de 11 bits para um código de função de 4 bits e ID de nó de 7 bits – uma combinação conhecida como identificador de objeto de comunicação (communication object identifier – COB-ID). Uma mensagem de broadcast neste sistema tem 0x tanto para o código de função quanto para o ID do nó. CANopen é visto mais em ambientes industriais do que em automóveis.

Se você encontrar um monte de IDs arbitrários de 0x0, você encontrou um bom indicador de que o sistema está usando CANopen para comunicações. CANopen é muito semelhante ao CAN normal, mas tem uma estrutura definida em torno dos IDs arbitrários. Por exemplo, as mensagens de pulsação (heartbeat messages) estão no formato de 0x700 + ID do nó. As redes CANopen são um pouco mais fáceis de reverter e documentar do que o barramento CAN padrão.

**O Barramento GMLAN (GMLAN Bus)**

GMLAN é uma implementação de barramento CAN pela General Motors. Ele é baseado na norma ISO 15765-2 ISO-TP, assim como o UDS (consulte “**Serviços de Diagnóstico Unificado – Unified Diagnostic Services**” na página 54). O barramento GMLAN consiste em um barramento de alta velocidade com um único fio de baixa velocidade e dois fios de alta velocidade no barramento. O barramento de baixa velocidade, o único fio do barramento CAN que opera em 33.33Kbps e com um máximo de 32 nós, foi adotado na tentativa de reduzir o custo de comunicação e fiação. Ele é utilizado para transportar informações não críticas para coisas como a central de info-entretenimento (central multimídia? “infotainment center”), controles HVAC, travas das portas, imobilizadores e assim por diante. Por outro lado, o barramento de alta velocidade funciona em 500Kbps com um máximo de 16 nós. Os nós em uma rede GMLAN estão relacionados aos sensores nesse barramento.

**O Protocolo SAE J1850 (SAE J1850 Protocol)**

O protocolo SAE J1850 foi originalmente adotado em 1994 e ainda pode ser encontrado em alguns dos veículos de hoje, por exemplo, alguns automóveis da General Motors e Chrysler. Estes sistemas de barramentos são mais antigos e mais lentos do que o CAN, mas é mais barato de se implementar. Existem dois tipos de protocolos J1850: modulação de largura de pulso (pulse width modulation – PWM) e largura de pulso variável (variable pulse width – VPW).

A velocidade é agrupada em três classes: A, B e C. As velocidades de 10.4Kbps de PWM e VPW são consideradas classe A, o que significa que são dispositivos comercializados exclusivamente para uso em empresas, indústrias e ambientes comerciais. (O barramento J1850 VPW de 10.4Kbps atende aos requisitos da indústria automotiva para emissões de baixa radiação). Os dispositivos de classe B são comercializados para uso em qualquer lugar, incluindo ambientes residenciais e tem uma segunda implementação de padrão SAE que pode se comunicar a 100Kbps, mas é um pouco mais caro. A implementação final pode operar em até 1Mbps, e é usada em dispositivos de classe C. Como você poderia esperar, esta terceira implementação é a mais cara, e é usada em sistemas críticos em tempo real e redes de mídia.

**O Protocolo PWM (PWM Protocol)**

PWM usa sinalização diferencial nos pinos 2 e 10 e é usado principalmente pela Ford. Ele opera com uma alta tensão de 5V e em 41.6Kbps, e usa sinalização diferencial de dois fios, como no CAN.

PWM tem um sinal de bit fixo, então um 1 é sempre sinal alto e um 0 é sempre um sinal baixo. Fora isso, o protocolo de comunicação é idêntico ao da VPW. As diferenças são a velocidade, tensão e número de fios usados para compor o barramento.

**O Protocolo VPW (VPW Protocol)**

VPW, um sistema de barramento de um único fio, usa apenas o pino 2 e é tipicamente usado pela General Motors e Chrysler. VPW tem uma alta tensão de 7V e uma velocidade de 10.4Kbps.

Quando comparado com a CAN, existem algumas diferenças fundamentais na forma como a VPW interpreta os dados. Por um lado, como o VPW usa a sinalização dependente do tempo, receber 1 bit não é determinado apenas por um alto potencial no barramento. O bit deve permanecer alto ou baixo durante um determinado período de tempo para ser considerado um único bit 1 ou 0. Puxar o barramento para uma posição alta o colocará em torno de 7V, enquanto o envio de um sinal baixo o colocará em níveis de terra ou perto do terra. Este barramento também está em uma fase de repouso, ou não-transmissão, em um nível próximo ao terra (até 3V).

A seção de dados é um tamanho definido – sempre 11 bits seguido por uma verificação de validade de CRC de 1 bit.

Os dados de resposta em quadro (In-Frame Response – IFR) podem seguir imediatamente após esta mensagem. Normalmente, um sinal de fim de dados (end-of-data – EOD) consistindo em 200us de longo sinal de baixo potencial seria ocorrer logo após o CRC, e se os dados IFR estão incluídos, ele vai começar imediatamente após o EOD. Se o IFR não estiver sendo usado, o EOD estenderá a 280us, causando um sinal de fim-de-quadro (end-of-frame – EOF).

**O Protocolo de Palavras-chave e ISO 9141-2(Keyword Protocol and ISO 9141-2)**

O protocolo de palavras-chave 2000 (ISO 14230), também conhecido como KWP2000, usa o pino 7 e é comum em veículos dos EUA feito após 2003. As mensagens enviadas usando KWP2000 podem conter até 255 bytes.

O protocolo KWP2000 tem duas variações que diferem principalmente na inicialização baud. As variações são:

ISO 14230-4 KWP (5 bauds init, 10,4 Kbaud)

ISO 14230-4 KWP (entrada rápida, 10,4 Kbaud)

ISO 9141-2, ou K-line, é uma variação do KWP2000 observada mais frequentemente em veículos europeus. K-Line usa o pino 7 e, opcionalmente, o pino 15. K-Line é um protocolo UART semelhante ao serial. Os UARTs usam bits de início e podem incluir um bit de paridade e um bit de parada. (Se você já configurou um modem, você deve reconhecer esta terminologia).

Ao contrário dos pacotes CAN, os pacotes K-Line têm um endereço de origem (transmissor) e o destino (receptor). O K-Line pode usar a mesma ou uma estrutura de solicitação de ID de parâmetro similar (PID) como CAN. (Para obter mais informações sobre PIDs, consulte “Serviços de Diagnóstico Unificado – Unified Diagnostic Services” na página 54).

**O Protocolo de Interconexão de Rede Local (Local Interconnect Network Protocol)**

O protocolo de Rede Local de Interconexão (LIN) é o mais barato dos protocolos do veículo. Foi concebido para complementar a CAN. Não tem arbitragem ou código de prioridade; em vez disso, um único nó mestre faz toda a transmissão.

O LIN pode suportar até 16 nós escravos que basicamente apenas escuta o nó mestre. Eles precisam responder na ocasião, mas isso não é sua principal função. Muitas vezes, o nó mestre LIN está conectado a um barramento CAN.

A velocidade máxima do LIN é 20Kbps. LIN é um barramento de um único fio que opera em 12V. Você não verá LIN quebrado (estourado/desobstruído – broken out) para fora ao conector de OBD, mas é usado frequentemente em vez dos pacotes diretos de CAN para segurar controles aos dispositivos simples, assim que esteja ciente de sua existência. Um quadro de mensagem LIN inclui um cabeçalho, que é sempre enviado pelo mestre, e uma seção de resposta, que pode ser enviada pelo mestre ou escravo.

O campo SYNC é usado para sincronização de clock. O ID representa o conteúdo da mensagem – ou seja, o tipo de dados que estão sendo transmitidos. O ID pode conter até 64 possibilidades. ID 60 e 61 são utilizados para transportar informações de diagnóstico.

Ao ler as informações de diagnóstico, o mestre envia com ID 60 e o escravo responde com ID 61. Todos os 8 bytes são usados em diagnósticos. O primeiro byte é chamado de endereço de nó para diagnósticos (node address for diagnostics – NAD). A primeira metade da faixa de bytes (ou seja, 1-127) é definida para diagnósticos compatíveis com ISSO, enquanto 128-255 pode ser específica para esse dispositivo.

**O Protocolo MOST (MOST Protocol)**

O protocolo MOST (Media Oriented Systems Transport) é projetado para dispositivos multimídia. Normalmente, MOST é apresentado em uma topologia em anel, ou estrela virtual, que suporta um máximo de 64 dispositivos MOST. Um dispositivo MOST atua como o mestre de cronometragem (mestre de tempo), que alimenta continuamente quadros no anel.

MOST é executado em aproximadamente 23 Mbaud e suporta até 15 áudio de qualidade de CD descompactado ou canais de áudio/vídeo MPEG1. Um canal de controle separado é executado em 768 Kbaud e envia mensagens de configuração para os dispositivos MOST.

MOST vem em três velocidades: MOST25, MOST50 e MOST150. Padrão MOST ou MOST25 corre em fibra óptica de plástico (plastic optical fiber – POF). A transmissão é feita através do comprimento de onda da luz vermelha a 650 nm usando um LED. Um protocolo semelhante, MOST50, duplica a largura de banda e aumenta o comprimento do quadro para 1025 bits. O tráfego MOST50 é geralmente transportado em cabos de par trançado não blindado (UTP) em vez de fibra óptica. Finalmente, MOST150 implementa Ethernet e aumenta a taxa de quadros para 3072 bits ou 150Mbps – aproximadamente seis vezes a largura de banda do MOST25.

Cada quadro de MOST tem três canais:

Dados transmitidos sincronizados (áudio/vídeo) – (**Synchronous** Streamed Data)

Dados distribuídos de pacotes assícronos (TCP/IP) – (**Asynchronous** Packet distributed data)

Controle de controle e dados de baixa velocidade (HMI) – (**Control Control** and low-speed data)

Além de um mestre de cronometragem (mestre de tempo – ‘timing master’), um mestre de rede MOST (a MOST network master) atribui automaticamente endereços a dispositivos, o que permite um tipo de estrutura plug-and-play. Outra característica única do MOST é que, ao conrtário de outros barramentos, roteia pacotes através de portas de entrada e saída independentes.

**Camadas da Rede MOST (MOST Network Layers)**

A menos que seu objetivo seja cortar um vídeo do carro ou fluxo de áudio, o protocolo MOST pode não ser tudo o que interessa para você. Dito isto, a MOST permite o acesso ao microfone ou sistema de células no veículo, bem como informações de trânsito que podem ser de interesse para autores de malware.

A MOST é dividida entre as sete camadas do modelo OSI que padroniza a comunicação sobre redes. Se você estiver familiarizado com outros protocolos de rede baseados em mídia, então MOST pode parecer familiar.

**Blocos de controle MOST (MOST Control Blocks)**

Em MOST25, um bloco consiste de 16 quadros. Um quadro é de 512 bits.

Os dados síncronos contêm de 6 a 15 quadlets (cada quadrlet é de 4 bytes) e os dados assíncronos contêm de 0 a 9 quadlets. Um quadro de controle é 2 bytes, mas depois de combinar um bloco completo, ou 16 quadros, você acaba com 32 bytes de dados de controle.

A área de dados contém o FblockID, InstID, FktID, OP Tipo, Tel ID, Tel Len e 12 bytes de dados. FblockIDs são os IDs de componentes principais ou blocos de função. Por exemplo, um FblockID de 0x52 pode ser o sistema de navegação. InstID é a instância do bloco de função. Pode haver mais de uma função de núcleo, como ter duas alterações de CD. InstID diferencia qual core para falar. FktID é usado para consulta blocos de função de nível mais alto. Por exemplo, um FktID de 0x0 consulta uma lista de identificações de função suportadas pelo bloco de função. OP Type é o tipo de operação a executar, get, set, increment, decrement, e assim por diante. O Tel ID e Len são o tipo de telegrama e comprimento, respectivamente. Os tipos de telegrama representam uma única transferência ou uma transferência de pacote múltiplo e o comprimento do próprio telegrama.

MOST50 tem um layout semelhante ao MOST25, mas com uma seção de dados maiores. MOST150 fornece dois canais adicionais: Ethernet e Isochronous. Ethernet funciona como normal TCP / IP e configurações Appletalk. Isochronous tem três mecanismos: modo de explosão (burst mode), taxa constante (constant rate) e transmissão de pacotes (packet streaming).

**Hacking MOST**

MOST pode ser hackeado a partir de um dispositivo que já suporta, como através da unidade de entretenimento informativo (central multimídia) de um veículo ou através de um controlador MOST bordo. O projeto baseado em Linux most4linux fornece um driver de kernel para a maioria dos dispositivos PCI e, a partir desta redação, suporta Siemens CT SE 2 e OASIS Silicon Systems ou cartões SMSC PCI. O driver most4linux permite comunicação com o espaço do usuário através da rede MOST e links para a estrutura Advanced ALSA (Arquitetura de Som Linux Avançada) para ler e gravar dados de áudio. No momento, most4linux deve ser considerado qualidade alfa, mas inclui alguns utilitários de exemplo que você pode ser capaz de construir em cima, a saber:

Most\_aplay Reproduz um arquivo .wav

Ctrl\_tx Envia uma mensagem de controle de transmissão (broadcast control message) e verifica o status

Sync\_tx Transmite constantemente

Sync\_rx Recebe constantemente

O driver most4linux atual foi escrito para kernels Linux 2.6, então você pode ter seu trabalho cortado se você quiser fazer um sniffer genérico. MOST é bastante caro para implementar, então um sniffer genérico não será barato.

**O Barramento FlexRay (FlexRay Bus)**

FlexRay é um barramento de alta velocidade que pode se comunicar em velocidades de até 10Mbps. Ele é voltado para a comunicação sensível ao tempo, como acionamento por fio, condução por fio, freio por fio, e assim por diante. FlexRay é mais caro implementar do que CAN, então a maioria das implementações usa FlexRay para sistemas high-end, CAN para intervalo médio (midrange) e LIN para dispositivos de baixo custo.

**Hardware**

FlexRay usa fiação de par trançado, mas também pode suportar uma configuração de canal duplo, que pode aumentar a tolerância a falhas e largura de banda. No entanto, a maioria das implementações FlexRay usa apenas um único par de fiação semelhante à implementação do barramento CAN.

**Topologia de Rede (Network Topology)**

O FlexRay suporta uma topologia de barramento padrão, como CAN bus, onde muitos ECUs funcionam sobre um barramento de par trançado. Ele também suporta topologia em estrela, como Ethernet, que pode executar segmentos mais longos. Quando implementado na topologia em estrela, um hub FlexRay é um dispositivo FlexRay central e ativo que fala com os outros nós. Em um layout de barramento, FlexRay requer terminação de resistor adequada, como em um barramento CAN padrão. As topologias de barramento e estrela podem ser combinadas para criar um layout híbrido se desejado.

**Implementação (Implementation)**

Ao criar uma rede FlexRay, o fabricante deve informar os dispositivos sobre a configuração da rede. Lembre-se que em uma rede CAN cada dispositivo só precisa saber a taxa de transmissão e que IDs se preocupa (se houver). Em um layout de barramento, apenas um dispositivo pode falar no barramento por vez. No caso do barramento CAN, a ordem de quem fala primeiro em uma colisão é determinada pela ID de arbitragem.

Em contraste, quando o FlexRay é configurado para falar em um barramento, ele usa algo chamado um esquema de acesso múltiplo por divisão de tempo (TDMA) para garantir o determinismo: a taxa é sempre a mesma (determinística) e o sistema depende dos transmissores para preencher os dados à medida que os pacotes passam pelo fio, semelhante à forma como as redes celulares como GSM operam. Os dispositivos FlexRay não detectam automaticamente a rede ou os endereços na rede, então eles devem ter essas informações programadas no momento da fabricação.

Embora essa abordagem de endereçamento estático reduza o custo durante a fabricação, pode ser complicado para um dispositivo de teste participar no barramento sem saber como a rede está configurada, como um dispositivo adicionado à sua rede FlexRay não saberá quais dados são designados para entrar em que slots. Para resolver este problema, formatos de troca de dados específicos, tais como o Formato de Troca de Barramento de Campo (Field Bus Exchange Format - FIBEX), foram concebidos durante o desenvolvimento do FlexRay.

O FIBEX é um formato XML usado para descrever o FlexRay, bem como configurações de rede CAN, LIN e MOST. Os mapas de topologia FIBEX registram os ECUs e como eles são conectados via canais, e eles podem implementar gateways para determinar o comportamento de roteamento entre barramentos. Esses mapas também podem incluir todos os sinais e como eles devem ser interpretados.

Os dados FIBEX são usados ​​durante o tempo de compilação do firmware e permitem que os desenvolvedores façam referência aos sinais de rede conhecidos em seu código; O compilador lida com todo o posicionamento e configuração. Para visualizar um FIBEX, faça o download do FIBEX Explorer em <http://sourceforge.net/projects/fibexplorer/>.

**Ciclos FlexRay (FlexRay Cycles)**

Um ciclo FlexRay pode ser visto como um pacote. O comprimento de cada ciclo é determinado em tempo de projeto e deve consistir em quatro partes: Static, Dynamic, Symbol Window, Idle.

O segmento estático (Static) contém slots reservados para dados que sempre representam o mesmo significado. Os slots de segmento dinâmicos (Dynamic) contêm dados que podem ter representações diferentes. A janela de símbolos (Symbol Window) é usada pela rede para sinalização, e o segmento ocioso (idle) (tempo silencioso) é usado para a sincronização.

A menor unidade de tempo no FlexRay é chamada de macro, que é tipicamente um milissegundo. Todos os nós são sincronizados no tempo, e eles acionam seus macro dados (macrotick data) ao mesmo tempo.

A seção estática de um ciclo FlexRay contém uma quantidade definida de slots para armazenar dados, tipo carros de trem vazios (empty train cars). Quando uma ECU precisa atualizar uma unidade de dados estáticos, ela preenche seu slot ou carro definido; Cada ECU sabe qual carro é definido para ele. Este sistema funciona porque todos os participantes em um barramento FlexRay são sincronizados no tempo.

A seção dinâmica é dividida em minislots, geralmente um macro longo. A seção dinâmica é geralmente usada para dados intermitentes menos importantes, como a temperatura interna do ar. Como um minislot passa, um ECU pode optar por preencher os minislots com dados. Se todos os minislots estiverem cheios, o ECU deve esperar pelo próximo ciclo.

Os ciclos FlexRay são representados como carros de trem. Os transmissores responsáveis ​​pelo preenchimento de informações para os slots estáticos fazem isso quando o ciclo passa, mas os slots dinâmicos são preenchidos com base no primeiro a chegar, primeiro a ser servido. Todos os vagões são do mesmo tamanho e representam as propriedades determinísticas do tempo do FlexRay.

A janela de símbolo não é normalmente usada diretamente pela maioria dos dispositivos FlexRay, o que significa que quando se pensa como um hacker, você deve definitivamente mexer com esta seção. Os clusters FlexRay funcionam em estados que são controlados pelo gerente de estado FlexRay. De acordo com o padrão AUTOSAR 4.2.1, esses estados são os seguintes: ready, wake-up, start-up, halt-req, on-line, online-passive, keyslot-only e low-number-of-coldstarters.

Enquanto a maioria dos estados são óbvios, alguns precisam de mais explicações. Especificamente, online é o estado de comunicação normal, enquanto online-passive só deve ocorrer quando há erros de sincronização. No modo online-passive, nenhum dado é enviado ou recebido. Keyslot-only significa que os dados só podem ser transmitidos nos slots de chave. Low-number-of-coldstarters significa que o barramento ainda está operando no modo de comunicação cheio, mas está confiando apenas nos quadros de sincronização. Há também estados operacionais adicionais, como config, sleep, receive only e standby.

**Layout de Pacote (Packet Layout)**

O pacote real que o FlexRay usa contém vários campos e se encaixa no ciclo no slot estático ou dinâmico. Os bits de status são:

Bit reservado (Reserved bit)

Indicador de preâmbulo de carga útil (Payload preamble indicator)

Indicador de quadro NULL (NULL frame indicator)

Indicador de quadro de sincronização (Sync frame indicator)

Indicador de quadro de inicialização (Startup frame indicator)

O ID do quadro é o slot no qual o pacote deve ser transmitido quando usado para slots estáticos. Quando o pacote é destinado a um slot dinâmico (1-2047), o ID de quadro representa a prioridade deste pacote. Se dois pacotes tiverem o mesmo sinal, então aquele com a prioridade mais alta ganha. O comprimento da carga útil é o número em palavras (2 bytes) e pode ter até 127 palavras de comprimento, o que significa que um pacote FlexRay pode transportar 254 bytes de dados - mais de 30 vezes o de um pacote CAN. Cabeçalho CRC deve ser óbvio, ea contagem de ciclo é usado como um contador de comunicação que incrementa cada vez que um ciclo de comunicação começa.

Uma coisa muito boa sobre slots estáticos é que uma ECU pode ler slots estáticos anteriores e produzir um valor com base nessas entradas no mesmo ciclo. Por exemplo, digamos que você tenha um componente que precisa saber a posição de cada roda antes que ela possa produzir os ajustes necessários. Se os primeiros quatro intervalos de um ciclo estático contêm cada posição de roda, a ECU de calibração pode lê-los e ainda tem tempo para preencher um slot posterior com quaisquer ajustes.

**Monitorando uma Rede FlexRay**

A partir desta redação, o Linux não tem suporte oficial para FlexRay, mas existem alguns patches de vários fabricantes que adicionam suporte a determinados kernels e arquiteturas. (O Linux tem suporte FlexCAN, mas FlexCAN é uma rede de barramento CAN inspirada pelo FlexRay.)

Neste momento, não há ferramentas de código aberto padrão para monitorar uma rede FlexRay. Se você precisa de uma ferramenta genérica para monitorar o tráfego FlexRay, você atualmente tem que ir com um produto proprietário que vai custar muito. Se você quiser monitorar uma rede FlexRay sem um arquivo FIBEX, você precisará, pelo menos, saber a taxa de transmissão do barramento. Idealmente, você também conhecerá o comprimento do ciclo (em milissegundos) e, se possível, o tamanho do particionamento do cluster (relação estática-dinâmica). Tecnicamente, um cluster FlexRay pode ter até 1048 configurações com 74 parâmetros. Você encontrará a abordagem para identificar esses parâmetros detalhados no documento "Identificação Automática de Parâmetros em Redes de Comunicação Automotiva Baseadas em FlexRay - Automatic Parameter Identification in FlexRay based Automotive Communication Networks" (IEEE, 2006), de Eric Armengaud, Andreas Steininger e Martin Horauer.

Quando falsificam pacotes em uma rede FlexRay com dois canais, você precisa simultaneamente burlar ambos. Além disso, você encontrará implementações FlexRay chamado Bus Guardian que são projetados para evitar inundações ou monopolização do barramento por qualquer dispositivo. Bus Guardian funciona no nível de hardware através de um pino no chip FlexRay normalmente chamado Bus Guardian Enable (BGE). Este pino é freqüentemente marcado como opcional, mas o Bus Guardian pode conduzir este pino muito alto para desativar um dispositivo de mau comportamento.

**Ethernet Automotiva (Automotive Ethernet)**

Porque MOST e FlexRay são caros e perdendo apoio (o consórcio FlexRay parece ter se dissolvido), a maioria dos veículos mais recentes estão se movendo para Ethernet. As implementações de Ethernet variam, mas são basicamente as mesmas que você encontraria em uma rede de computadores padrão. Muitas vezes, os pacotes CAN são encapsulados como UDP, e o áudio é transportado como voz sobre IP (VoIP). Ethernet pode transmitir dados em velocidades de até 10Gbps, usando protocolos não proprietários e qualquer topologia escolhida.

Embora não haja um padrão comum para o tráfego CAN, os fabricantes estão começando a usar o padrão IEEE 802.1AS Audio Video Bridging (AVB). Este padrão suporta qualidade de serviço (QoS) e modelagem de tráfego, e usa pacotes UDP sincronizados no tempo. Para alcançar essa sincronização, os nós seguem um melhor algoritmo de clock mestre para determinar qual nó deve ser o mestre de tempo. O nó mestre normalmente sincronizará com uma fonte de temporização externa, como GPS ou (pior caso) um oscilador on-board. O mestre sincroniza com os outros nós enviando pacotes cronometrados (10 milissegundos), o escravo responde com um pedido de atraso e o deslocamento de tempo é calculado a partir dessa troca.

Do ponto de vista do pesquisador, o único desafio com o veículo Ethernet reside em descobrir como falar com a Ethernet. Você pode precisar fazer ou comprar um cabo personalizado para se comunicar com os cabos de Ethernet do veículo porque eles não se parecerão com os cabos de par trançado padrão que você encontrará em um hack de rede. Normalmente, um conector será apenas fios como os que você encontra conectado a uma ECU. Não espere que os conectores tenham seu próprio plugue, mas se o fizerem, ele não se parecerá com um conector RJ-45. Alguns conectores expostos são realmente redondos.